Generalization from Observed to Unobserved Features

Thesis submitted in partial fulfillment of the degree of
Doctor of Philosophy
by

Eyal Krupka

Submitted to the Senate of the Hebrew University

March 2008

This work was carried out under the supervision of Prof. Naftali Tishby.

Acknowledgments

TODO

We thank Amir Navot, Ran Gilad-Bachrach, Idan Segev and Shai Shalev-Shwartz for helpful discussions. We also thank Koby Crammer for the use of code for Multiclass SVM [21]. We also thank the GroupLens Research Group at the University of Minnesota for use of the MovieLens data set.

Abstract

The thesis discusses various aspects of learning from a subset of features to another feature subset in the framework of machine learning. We propose *feature to feature* learning and generalization as a possible model for clustering, as a method to improve feature selection and to incorporate prior knowledge in supervised learning. We also discuss the possible relation of our models to learning in biological neural systems.

The most common machine learning model is learning from examples. In this model, the algorithm learns structures from a finite set of observations, called a training set, and uses the learning results to perform some task. In supervised learning each observation is a pair made up of an instance and a label, and the algorithm learns to map between instances to labels. We expect a good learning algorithm to generalize from the finite training set to new unseen instances, and to be able to accurately predict their labels. The instances are typically represented by vectors of measurements, where each measurement is called a feature. For example, in an optical handwritten recognition task, an instance can be an image of the character and the label the name of the character. The features can be the gray level values of each pixel, or functions of these pixels such as strokes.

Another prominent learning model is clustering, where each observation contains an instance without a label. Clustering deals with grouping observations based on their similarity. In general, it is not clear how to measure the quality of clustering, since we do not have a well-defined prediction task as in supervised learning. In some cases, the goal of clustering is to predict some label which is unknown to the clustering algorithm. However, it is unclear how we can optimize the quality of the clustering to an unknown label. Even worse, the clustering quality depends on the specific choice of the labels. For example, good documents

clustering with respect to topics is very different from clustering with respect to authors.

In Chapter 2 we propose a predictive clustering model, where the goal is to predict unobserved properties (features) of objects. We assume that all objects are observed, but there are features of the objects that are unknown to the learner at the time of clustering. The goal of the learner is to learn and generalize from the set of observed features to new unobserved features. For example, when clustering fruits based on their observed features, such as shape, color and size, the target of clustering is to match unobserved features such as nutritional value or toxicity. When clustering users based on their movie ratings, the target of clustering is to match ratings of movies that were not rated, or even not created as yet. The basic assumption that we use is that the set of observed features is a subset of features that were selected randomly and independently from some huge set of features. We show that under mild technical conditions, clustering the objects on the basis of such a random subset performs almost as well as clustering with the full attribute set. We prove finite sample generalization theorems for this novel learning scheme that extends analogous results from the supervised learning setting. We use our framework to analyze generalization to unobserved features of two well-known clustering algorithms: k-means and the maximum likelihood multinomial mixture model. The scheme is demonstrated for collaborative filtering of users with movie ratings as features and document clustering with words as features.

In Chapter 3 we extend this framework beyond clustering, and analyze feature generalization properties of the nearest neighbor rule. We present generalization bounds on the distance between unobserved features of instances which are nearest neighbors based on the distance between the observed features.

Learning and generalization from one set of features to another set is also useful in supervised learning. In most learning models, instances are represented by vectors of features. However, in most real-life problems, we have additional information about each of the features beyond the observations. For example, in a handwritten recognition task, where the features are pixels, we know the position (row, col) of each of the pixels. In collaborating filtering, where each user is represented as a vector of movie ratings, we have additional information on each movie (e.g., genre, actors). This additional information on the features can be represented as a set of properties, and is referred to as *meta-features*. We use

meta-features to extend the formulation and improve performance of supervised learning by two different methods. The first is to improve the feature selection process by predicting the quality of unobserved features based on their meta-features and previously observed features. The second method improves generalization of supervised learning by using the meta-features as prior knowledge on the learning problem.

Feature selection is the task of choosing a small subset of features that is sufficient to predict the target labels well. In Chapter 4, instead of trying to directly determine which features are better, we attempt to learn the properties of good features. This approach uses meta-features to predict the quality of features without measuring their value on the training instances. We use this ability to devise new selection algorithms that can efficiently search for new good features in the presence of a huge number of features, and reduce dramatically the number of feature measurements needed. We demonstrate the application of our algorithms on a visual object category recognition problem. In addition, we show how this novel viewpoint enables derivation of better generalization bounds for the joint learning problem of selection and classification, and how it contributes to a better understanding of the problem. Specifically, in the context of object recognition, previous works have shown that it is possible to find one set of features (aka *universal dictionary*) which can be used for recognition of most object categories. Here we use our framework to analyze one such universal dictionary and find that the quality of features in this dictionary can be predicted accurately by its meta-features.

Meta-features can also be used for selecting which features to extract in the process of feature extraction. An extracted feature is a function of the raw features which is used as input to the classifier. This function is typically parametric, and the parameters of this function can be used as meta-features. Alternatively, the meta-features can be functions of these parameters. In principle, we could extract all possible features and then use feature selection by meta-features. However, the possible number of extracted features is typically huge or even infinite. Thus, we need an efficient method to search for good features without extracting all of them. In Chapter 5 we present and analyze such a method, and demonstrate it on a handwritten digit recognition problem.

In Chapter 6 we propose a new supervised learning framework that incorporates meta-

features. In this framework we assume that a weight is assigned to each feature, as in linear discrimination, and we use the meta-features to define a prior on the weights. This prior assumes that the weight is a smooth *function* of the meta-features. For example, in a movie rank prediction application we expect that movies with similar meta-features (genres) will tend to have similar weights. In handwritten digit recognition, we expect weights of features with similar meta-features (that is, representing adjacent pixels) to be similar. This prior enables us to learn from the weight of one feature to the weights of its neighbor features. This also means that if, by chance, one of the features is constant (or even missing) in the training set, we can still predict its weight by the weight of its adjacent features.

Using this framework we derive a practical learning algorithm that improves generalization by using meta-features. In this algorithm, we use a Gaussian Process as a smoothness prior of the meta-feature to weight mapping. We apply the algorithm to design a new kernel for the Support Vector Machine (SVM) in a handwritten digit recognition task. This kernel is based on explicit calculation of selected monomial features and can be solved in the primal representation; that is, by using a linear classifier on the calculated monomial features. We obtain higher accuracy with lower computational complexity relative to a standard polynomial kernel. In a second experiment, we use meta-features to improve movie-rating predictions. We also discuss the theoretical advantages of incorporating meta-features into learning.

In Chapter 7 we discuss the applicability of our framework of using meta-features as prior for learning in biological neural networks. More specifically, we point to a possible relation between meta-feature based regularization and a few learning mechanisms in neurons which are affected by the geometry of the neurons. We use the following simplifying assumptions. First, feature values are represented by neuron activity. Second, neurons in the cortex are spatially organized by the value of *meta-features* related to each neuron; that is, neurons which represent similar features are spatially proximate. Similarity of features here means proximity in meta-feature space. This assumption is supported by the existence of organization maps in the cortex; e.g., in the organization of the primary visual cortex the meta-features can be position, orientation and scale. The third assumption is that the learning rule of updating weights of a neuron depends not only on the activity of the input

neurons, but also on their relative spatial position, such that the weight update to proximate input neurons tends to be correlated. There is evidence for this type of learning in the brain: for example Engert and Bonhoeffer [28] found that weights (the synaptic strength) to input neurons in close spatial proximity tend to be potentiated together, even if some of the input neurons are not active. This evidence for a weight update through the activity of spatially adjacent synapses of input neurons, together with the above assumptions, can be related to the prior of smoothness of weights in meta-feature space we present in Chapter 6.

Contents

Chapter 1

Introduction

Machine learning is concerned with the development of algorithms, theory and applications that learn from data. In general, the goal of learning is to extract useful information from data. There are several machine learning models, which differ as regards the information they extract from data and the type of data they deal with. In this chapter, we review the main machine learning models and notions that constitute the context and background for this thesis.

1.1 Supervised Learning

In the supervised learning setup the goal is to learn a function from an input pattern to a label; e.g., given an image of a digit, the output of the function might be the digit value. The input to the learning algorithm is a *training set* S of m instances. Each instance in the training set includes a pair $\{\mathbf{x}, y\}$ where $\mathbf{x}$ is the input pattern and y is the label. Typically, the input pattern is a vector in $\mathbb{R}^n$, where each of the n dimensions is called a feature. We assume that there is some unknown target concept c that maps the inputs to the labels, $y = c(\mathbf{x})$. The output of the algorithm is a function, or a hypothesis h that maps between the input pattern $\mathbf{x}$ and a label y. In order to formalize and analyze this setup, it is commonly assumed that there is some unknown fixed distribution $\mathcal{D}$, from which the instances are independently drawn. The performance of the learning algorithm is then

measured by the *generalization error*, which is the probability of error in predicting the label on a new instance that was drawn from $\mathcal{D}$:

$$err_{\mathcal{D}}(h) = \Pr_{\mathbf{x}\sim\mathcal{D}}\left(h\left(\mathbf{x}\right) \neq c\left(\mathbf{x}\right)\right).$$

Since concept c and distribution $\mathcal{D}$ are unknown to the learning algorithm, it cannot be minimized directly. However, we can directly measure the *training error* of any concept c

$$err_S(h) = \frac{1}{m}\sum_{i=1}^{m} I_{\{h(\mathbf{x}_i)\neq y_i\}}.$$

where $I_{\{b\}}$ is 1 if the boolean expression, b, is true, and 0 otherwise.

Based on the training set, the learning algorithm selects h from a given class of hypotheses $\mathcal{H}$. Since we cannot minimize the generalization error directly, one commonly used strategy is the *Empirical Risk Minimization* (ERM) (see e.g., [112]), where we find $h \in \mathcal{H}$ that minimizes the empirical risk, which in the above case is the training error[1]. However, in many cases the selection of the hypothesis may overfit the training set, and hence a low training error does not guarantee a low generalization error. A low training error is valuable only when we have good generalization, in other words, when, the difference between training and generalization error is small. In the following section we briefly present some of the approaches in statistical learning theory used to analyze the relation between the training and generalization error.

1.1.1 Generalization

One of the key issues in statistical learning theory is to analyze our ability to generalize from a finite training set. Valiant [110] defined the PAC (Probably Approximately Correct) framework, which includes an analysis of the difference between training and generalization error of hypothesis (concept) classes. PAC deals with the relationship between the size of the training set (m), the difference between the training and the generalization error (ϵ) and the properties of the hypothesis class $\mathcal{H}$. The output of the algorithm (hypothesis h) depends on the randomly drawn training set. Hence, in the PAC framework we analyze the

[1] In the general case, the empirical risk is the average of any loss function $L\left(c(\mathbf{x}), h(\mathbf{x})\right)$. In our special case the loss is one if $h\left(\mathbf{x}\right) \neq c\left(\mathbf{x}\right)$, and zero otherwise.

probability (δ) of drawing a "bad" training set:

$$\Pr_{S\sim\mathcal{D}^m}\{|err_S(h) - err_{\mathcal{D}}(h)| > \epsilon\} \leq \delta,$$

where δ depends on the hypothesis class $\mathcal{H}$ and the size of training set.

In the case where the learning algorithm select one hypothesis from a finite set of $|\mathcal{H}|$ hypotheses, we get the cardinality bound below:

$$\delta = 2\,|\mathcal{H}|\,e^{-2m\epsilon^2}.$$

Intuitively, if the hypothesis set is large and the training set is small, we may get a hypothesis with low training error just by chance. Hence, in order to guarantee small difference between the training and the generalization error it is better to use a small hypothesis class. However, the hypothesis class we use should contain the correct hypothesis, otherwise both the training and generalization error might be large. This results is related to the Occam's Razor principle "*All things being equal, the simplest solution tends to be the right one*"[2]. In the cardinality bound, a simple solution is one from a small hypothesis space.

In many learning algorithms the hypotheses are parametric; that is, each hypothesis in $\mathcal{H}$ is defined by a set of parameters. When the parameters are continuous the size of the hypothesis class is infinite. Vapnik and Chervonenkis defined the VC-dimension [113] as a measure of the capacity, or complexity, of the hypothesis class. As before, the bound on the difference between the training and the generalization error is tighter when the VC-dimension is smaller, in other words, when the hypothesis class is simpler.

A classic algorithm with continuous parameters that minimizes the empirical risk is the Perceptron [86]. The Perceptron algorithm finds a hyperplane that separates between all points in the training set such that all instances with positive labels are in one side of the hyperplane, and all the negative labels are on the other.

In practice, we typically do not know in advance how complex our hypothesis class should be to find a hypothesis consistent with the training set, or with a low training error. To address this problem Vapnik and Chervonekis proposed the principle of Structural Risk Minimization (SRM). They divided the class of hypotheses into a hierarchy of nested subset

[2]The translation of original citation is "entities should not be multiplied beyond necessity". The above is one of its common rephrasings.

of hypotheses with increasing complexity, e.g., $|\mathcal{H}_1| < |\mathcal{H}_2| < |\mathcal{H}_3| < ....$ Once we have the training set we can use ERM to select the best hypothesis from each hypothesis class. Then we can control the balance between complexity and training error, for instance by selecting the hypothesis from the smallest set which is consistent[3] Unlike the cardinality bound described above, the generalization bound in SRM depends not only on the size of training set, but on the data themselves. These types of bounds are called *data dependent* bounds [3, 95].

One of the most successful algorithms that uses the SRM principle is the Support Vector Machine (SVM) [111]. This algorithm finds the hyper-plane that separates between all points in the training set with the maximum margin, i.e., the plane as far from the training points as possible. To achieve non-linear separation between the training points, the input vectors can be mapped into a high dimensional space. This mapping can be done implicitly by using kernel functions. The generalization bound of SVM depends on the actual margin of the result hyperplane, and hence is a data dependent bound.

There are many other methods available to work with complex hypothesis sets, typically by using some bias (give higher priority) to simpler hypotheses in order to avoid overfitting. These methods are called regularization methods. In SVM the regularization term is an additional cost to the L2-norm of the weights, which is equivalent to preferring large margin classification. Other regularization methods, which are used for model selection, include the Minimum Description Length (MDL) [85], and the Bayesian information criterion (BIC) [89].

1.1.2 Prior Knowledge

In the previous section we discussed the need to limit the complexity of the hypothesis space in order to achieve good generalization. Arbitrary reduction of the hypothesis space can reduce the difference between training and generalization error, but this may cause an increase in the training error - since we might exclude good hypotheses. Loosely, as manifested by the *no free lunch* theorems [118], without some bias or limitation on the hypothesis space, i.e., a preference for some hypothesis over others, we cannot learn from a finite training set.

[3] For other alternatives that control the balance between complexity and empirical risk, see e.g., [111]

Hence, to achieve good generalization we must incorporate prior knowledge on the learning problem we want to solve.

Prior knowledge is defined as all information about the problem available in addition to the training data [88]. Many learning algorithms incorporate some general assumptions on the hypotheses which may be correct for wide variety of learning problems. The most common assumption is smoothness of the label in the input space; in other words two input points which are similar (adjacent) are likely to have the same label. The simplest learning algorithm that uses this assumption is the nearest neighbor algorithm [29]. This algorithm assigns a label to a test point based on the label of the nearest point in the training set. Another common assumption is that the probability density $P(\mathbf{x})$ is low between points of different classes. Hence the decision boundary between classes is expected to be in low-density regions. This assumption explains why in many problems it is possible to find large-margin separation between different classes, as is the case in SVM.

The existence of such general assumptions allows us to use general-purpose learning algorithms and get good results for classification problems from different domains. Most of these general purpose algorithms assume that each of the instances is given as vector $\mathbb{R}^n$, and deals with all features regardless of their origin. However, in most practical problems additional domain knowledge is required to achieve good performance. Actually, even when using one of the standard general-purpose learning algorithms, prior knowledge is incorporated in the phase of choosing the correct representation of the input to the algorithm. The vector of n features can be composed manually, and selected by a human expert, typically by defining n functions of the raw input features. For example, in many visual object-recognition algorithms [91, 74, 120] SVM is used, but only after modifying the representation from raw-pixels to high-level, more complex features (e.g., local shapes, edges etc.). Finding a good representation for this problem is a challenging goal. Working directly with SVM on the raw-pixels typically yields poor results in most non-trivial visual object recognition tasks. As in the last example, in many applications one of the most crucial steps for a successful classifier is to use a good representation, typically as a vector with a fixed number of elements (features). Only then we can successfully use general-purpose classifiers. This means that the existence of general-purpose classifiers usually does not solve the need for addi-

tional domain knowledge, but allows the researcher to use standard tools after the domain knowledge is implemented by creating a good representation.

Recently, Guyon *et al.* [38] organized a challenge to compare performance of agnostic learning vs. prior knowledge methods. In this challenge, participants designed classifiers for five different datasets. The participants were allowed to compete in two tracks: The "prior knowledge" (PK) track, for which they had access to the original raw data representation and as much knowledge as possible about the data, and the "agnostic learning" (AL) track for which they were forced to use data pre-formatted as a table with dummy features. For the first few months of the challenge, AL was in the lead, showing that the development of good AL classifiers is considerably faster. After several months, however PK was ahead of AL on four out of five datasets [38].

In addition to representation, prior knowledge can be incorporated by modifying off-the-shelf classifiers, learning algorithms, or even by building special classifiers for specific problems. For example, for handwritten digit recognition tasks good performance can be achieved by using various heuristics such as specially engineered architectures or distance measures (see e.g., [67, 97]). In the context of SVM a variety of successful prior knowledge incorporation methods have been published over the last ten years (see e.g., [23]). In a recent review [65], these methods were classified into four categories: (i) sample methods based on modification of training data, e.g. creating virtual samples; (ii) regularization methods based on modification of the cost function; (iii) kernel design techniques and (iv) constrained methods based on adding constraints to the optimization problem.

In chapter 6 we propose a new method to incorporate the prior knowledge we have on features into a general-purpose classifier such as SVM. The rationale is based on the fact that in many cases even after we represent instances as vector of features, we have additional knowledge about each of the features and on relationships between features. This knowledge can be represented by vectors that describe each of the features, i.e., vectors of *meta-features* (see Section 1.3.3). Then, we can use the meta-features to learn from the weight of one feature to weights of other features, similar to the way that we learn from one instance to another in supervised learning.

1.1.3 Feature Selection

In many supervised learning tasks the input is represented by a very large number of features, many of which are not needed for predicting the labels. *Feature selection* is the task of choosing a small subset of features that is sufficient to predict the target labels well. The main motivations for feature selection are computational complexity, reducing the cost of measuring features, improving classification accuracy and problem understanding. Feature selection is also a crucial component in the context of *feature extraction*. In feature extraction the original input features (e.g., pixels) are used for generating new, more complicated features by a set of functions on the original features (e.g., multiplications of a few raw features). Feature extraction is a very useful tool for producing sophisticated classification rules using simple classifiers. One main problem here is that the potential number of additional features we can extract is huge, and the learner needs to decide which of them to include in the model.

In the most common selection paradigms an evaluation function is used to assign scores to subsets of features and a search algorithm is used to search for a subset with a high score. The evaluation function can be based on the performance of a specific predictor (*wrapper* model) or on some general (typically cheaper to compute) relevance measure of the features to the prediction (*filter* model) [57]. In any case, an exhaustive search over all feature sets is generally intractable due to the exponentially large number of possible sets. Therefore, search methods apply a variety of heuristics, such as hill climbing and genetic algorithms. Other methods simply rank individual features, assigning a score to each feature independently. These methods ignore redundancy and inevitably fail in situations where only a combined set of features is predictive of the target function. However, they are usually very fast, and are very useful in most real-world problems, at least in the initial stage of filtering out useless features. One very common such method is *Infogain* [82], which ranks features according to the mutual information[4] between each feature and the labels. Another selection method is Recursive Feature Elimination (RFE, [39]). SVM-RFE is a wrapper selection method for linear Support Vector Machine (SVM). In each round it measures the

[4]Recall that the mutual information between two random variables X and Y is $I(X;Y) = \sum_{\{x,y\}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$.

quality of the candidate features by training SVM and eliminates the features with the lowest weights. See [37] for a comprehensive overview of feature selection.

1.2 Unsupervised Learning

In unsupervised learning the input to the learning algorithm is a set of m input instances $\mathbf{x}_1, ..., \mathbf{x}_m$ without any label. The task in unsupervised learning is to find structures in the observed data. Unlike supervised learning, the goal is not always well-defined, and hence it is harder to evaluate the results or theoretically analyze performance. The field of unsupervised learning is very broad, and includes many models such as *clustering*, *dimensionality reduction,* and *probabilistic generative models.* In clustering [48, 119], the input instances are clustered into groups based on their similarity. In dimensionality reduction we transform the data into a lower dimensional space. A simple example of a dimensionality reduction algorithm is *Principal Component Analysis* (PCA) [49] that finds a linear transformation into orthogonal dimensions that has the maximum variance. This is achieved by eigenvalue decomposition. Graph based methods for dimensionality reduction (see e.g., [105], [9]) embed the data points into low-dimensional space while trying to preserve pairwise distances or neighborhoods between the points. Generative model methods try to find a probabilistic model that created the data, for example the existence of a discrete hidden variable such that all observed variables are independent given the hidden variable. The most common algorithm for finding the parameters of the model is the EM algorithm [24].A generative approach to finding interesting projections of the data is Independent Component Analysis (ICA) [10], where it is assumed that each element (feature) in the observation is a linear combination of non-Gaussian independent random variables.

1.2.1 Clustering

Clustering deals with grouping objects (i.e., instances, data points) based on their similarity. One standard approach to clustering is to characterize the source of data by mixture density. The parameters of the distributions are typically estimated by maximum likelihood, and the task is solved by the EM algorithm. In this process, each point in the distributions is

assigned to one (or more) clusters. In *hard clustering*, a single cluster assigned to each point. In *soft clustering* a probability per cluster is assigned to each point. A simple example of a clustering algorithm is the *k-means* [69, 70, 41], which is well suited when we assume that the points are generated from a mixture of k Gaussians.

A notable alternative to generative clustering is the use of *spectral methods* for clustering (see e.g., [76, 115]). These methods are based on the top eigenvectors of the matrix derived from the pairwise distance between data points. The assumption of these methods is that the clusters are volumes with a high density of points separated by volumes of low density.

A different approach to clustering, proposed by Slonim and Tishby [98, 107], is based on the *information bottleneck* method. Unlike distance-based clustering, this method is based on distributional clustering and the optimization goal of clustering is based on Shannon's mutual information [94, 20]. The information between two random variables measures the amount of information that can be obtained about one random variable by observing another. The mutual information between X and Y is denoted by $I(X;Y)$ and is defined as

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}.$$

The basic idea of the information bottleneck [107] is to find a compressed representation, T, of a random variable X, that preserves the maximum possible information on a second random variable Y. The input to this algorithm is the joint probability distribution $p(x,y)$. Slonim and Tishby [98] used this method for document clustering. On one side of the trade-off the variable T "compresses" the documents into a few clusters. On the other side, it tries to preserve maximum information between T and the distribution of words in documents for example. Intuitively, this is done by putting documents with similar distributions of words in the same cluster.

In most cases, unlike supervised learning, the goal of clustering is not predictive. This is the case in particular if all instances are given, and we do not try to say anything about new instances. Hence, the quality of clustering cannot be measured by some prediction error, and its not clear how to define generalization in clustering. In general, we do not have an absolute judgment for the quality of clustering [48], as pointed out by Backer and Jain [2, 119], "in cluster analysis a group of objects is split up into a number of more or less

homogeneous subgroups on the basis of an often *subjectively* chosen measure of similarity (i.e., chosen subjectively based on its ability to create "interesting" clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups".

An interesting work which addresses the difficulty of defining a good clustering was presented by Kleinberg [56]. In this work the author enumerates the desired properties a distance-based clustering algorithm in $\mathbb{R}^n$ should satisfy, such as scale invariances and richness of possible clusterings. Then he proves that it is impossible to construct a clustering that satisfies all the requirements.

In a recent paper [114] Luxburg and Ben-David discuss the goal of clustering in two very different cases. The first is when we have complete knowledge about our data generating process, and the second is how to approximate an optimal clustering when we have incomplete knowledge about our data. In most current analyses of clustering methods, incomplete knowledge refers to getting a finite sample of instances rather than the distribution itself. Thus, we can define the desired properties of a good clustering. An example of such a property is the stability of the clustering with respect to the sampling process; e.g., the clusters do not change significantly if we add some data points to our sample.

In chapter 2 of this thesis we propose a predictive goal for clustering and a possible definition for generalization in clustering. The prediction refers to unobserved properties (features) of the objects, rather than unobserved instances. We assume that all objects are observed, but there are features of the objects that we might not be aware of at the time of clustering. We analyze our ability to predict them, and to generalize from the observed to the unobserved features.

1.3 Labels, Instances and Features

In the previous sections we defined instances, features and labels. In supervised learning we have training instances, where each instance includes a vector of features and a label. The goal is to predict labels of new instances based on their feature vector. In unsupervised learning we do not have labels, and we try to find structures from the relationships between

features. However, in many cases, the setup is more complex.

Sometimes in clustering we assume that there is a label, which is unknown to the learning algorithm. Thus, it is desirable to have the clusters match this unknown label. For example the performance of Slonim and Tishby's unsupervised document clustering algorithm [98, 99] is evaluated by comparing the resultant clustering to the topic of each document. The topic can be considered as a "hidden" label, since it is unknown to the clustering algorithm. Practically, the performance of the clustering algorithm is easy to measure in this case, for instance by the mutual information between the clusters and the label. However, since the clustering algorithm has no access to the labels, it is unclear how it can optimize the quality of the clustering. Even worse, the clustering quality depends on the specific choice of the unobserved labels. For example, good document clustering with respect to topics is very different from clustering with respect to authors. Hence, to design a clustering algorithm that successfully predicts the label, we need to make assumptions about the relationships between the distribution of instances $P(\mathbf{x})$ and the distribution of the label given the instances $P(y|\mathbf{x})$. Only then can we learn from the empirical distribution $P(\mathbf{x})$ about the label.

Some of the common assumptions used in supervised learning are also applicable to clustering. This includes the assumption that the classes are separated by low-density volumes in the feature space. Another assumption which is sometimes used is that the features are generally statistically dependent, but statistically independent given the label, i.e.,

$$P(x_1, ..., x_n|y) = \prod P(x_i|y).$$

This assumption is directly used by supervised algorithms like Naive Bayes (see e.g., [30]), and in clustering algorithms that find a hidden separating variable. However, some of the assumptions that are very useful in supervised learning are not sufficient for unsupervised clustering that tries to predict the label. For example the smoothness of the label in the feature-space is not informative in itself about the relation between $P(\mathbf{x})$ and $P(y|\mathbf{x})$.

Becker and Hinton [7, 6] proposed an unsupervised learning model where two different perceptual inputs have a common cause in the outside world. In the framework of the above discussion, this common cause can be considered as the unobserved label, which is

the target of learning. They proposed an algorithm called IMAX where there are two neural network modules, each of which receives input from a different view of the same instance. The learning algorithm finds the common cause in the world by maximizing the mutual information between the outputs of these modules[5]

In chapter 2 of this thesis we propose a novel assumption on the relation between the distribution of observed features $P(\mathbf{x})$ and the hidden label in the context of clustering. We assume that the hidden label is yet another feature that "happened to be" unobserved. More formally, we assume that each feature and the label is selected randomly and independently from some fixed distribution. This assumption enables us to theoretically analyze generalization from the observed features $\mathbf{x}$ to the hidden label (feature) y.

1.3.1 Learning from Labeled and Unlabeled Data

In recent years, models for learning both from labeled and unlabeled data have received increasing attention. In these models it is typically assumed that we have a small amount of labeled data and a large amount of unlabeled data. One class of models is transductive learning [112], where it is assumed that all data (labeled and unlabeled) are given at the time of learning. Unlike inductive learning, we do not need to generate a classification rule, but only label the given unlabeled data. One of the baseline methods for transductive learning is first to cluster the instances without using the labels, and then assign a label to each cluster by the majority of labeled instances in the cluster. Another example of transductive algorithm is the transductive SVM [112].

A second category of models for learning from labeled and unlabeled data is called semi-supervised learning [16, 121]. In this category the learning algorithm needs to generate a classification rule in order to classify new instances that were not part of the training. An example of a semi-supervised algorithm is co-training [13]. This algorithm, proposed by Blum and Mitchell, is based on a principle similar to IMAX. It includes two classifiers, each of which receives input from a different view of the same instance. First, each classifier is trained on the labeled data, using standard supervised learning. Then, each classifier is used

[5] From the data processing inequality [20] the maximum information can be found simply by assigning the input of each module to its output. Hence, using IMAX makes sense only if we limit the complexity of the output of the modules, e.g., by constraining the outputs to be a binary variable.

to generate labels (predictions) on the unlabeled data. The output labels of each classifier are used to train the other classifier. The algorithm is used iteratively on the unlabeled data, where in general the prediction improves during the iterations.

1.3.2 Other Models

In some cases there is no clear distinction between features and labels. Let's consider the case of predicting movie ratings in the framework of collaborative filtering [35]. Collaborative filtering refers to methods of making predictions about a user's preferences, by collecting preferences of many users. In collaborative filtering for movie ratings the goal is to predict movie ratings by a user given a partial list of ratings from this user and many other users. Clustering methods are used for collaborative filtering by clustering users based on the similarity of their ratings (see e.g., [72, 109]). Consider Table 1.1. Suppose that our goal is to predict the movie rating of Tommy for the movie K-Pax. Let's try to define this problem in the standard supervised learning setup. One way to view it is to regard each user as an instance and each movie except K-Pax as a feature. The feature value is the movie rating, though some of the feature values may be missing (unknown). The rating of the movie K-Pax is the label. The training set contains all users who rated K-Pax. However, we can view this in a "transposed" way. In this case each movie is an instance, and each user except Tommy is a feature and Tommy's rating is the label. The training set in this case is all movies that were rated by Tommy.

This is one example where the role of instance, features and labels is not unique. Indeed, many algorithms for collaborative filtering are not formulated in the standard supervised learning of instances and labels, and are viewed as a task of filling in missing values of the rating matrix (see e.g., [101]). Examples of other cases where the matrix of features-instances can be viewed symmetrically is in gene expression (see e.g., [39]) and the symmetric information bottleneck applied to document-word clustering [99].

Some domains in machine learning are significantly richer than the feature-label framework. This includes *reinforcement learning* [51] where we have an agent the performs actions in an environment. Based on the action of the agent, he is rewarded (or penalized). The

	Matrix	Shrek	Star Wars	K-Pax
John	Liked	Disliked	Liked	Disliked
David	Liked	Liked	Disliked	Liked
Jodie	Disliked	?	Disliked	Liked
Tommy	Disliked	Liked	Disliked	?

Table 1.1: Example of a movie rating dataset. Each user is a vector of movie ratings (liked/disliked), with some values that may be missing (unrated movies). The goal of the learning algorithm is to predict the missing values.

goal of learning is to select actions that maximize long-term reward.

1.3.3 Meta-features - the Properties of Features

In the standard formulation of supervised learning the input is represented as a vector of features. However, in many learning problems, we have additional information about each of the features. This information can be represented as a set of properties, referred to as *meta-features*. For instance, in an image recognition task, where the features are pixels, the meta-features can be the (x, y) position of each pixel. In collaborative filtering, where the users are represented as vectors of movie ratings, the meta-features can be the additional information we have on each movie. In text classification, where the features are words, the meta-features can represent any additional information we have on the words such as stem, semantic meaning, synonym, part of speech etc. Another potential application that can use meta-features is gene-expression. In this case the gene-expression is the feature, and the meta-features represent our knowledge about each gene. In Table 1.2 we summarize examples of candidate meta-features for some learning tasks.

Using meta-features can enrich learning in several ways. Taskar *et. al.* [104] used meta-features of words for text classification where there are features (words) that are unseen in the training set, but appear in the test set. In their work the features are words and the meta-features are words that appear in the neighborhood of that word. They used the meta-features to predict the role of words that are unseen in the training set. Raina *et. al.* [84] use a property of pairs of words which indicates whether they are synonyms or not for the task of estimating the words' covariance matrix. Kadous and Sammut [50] used property-

Application	Example of possible features	Meta-features	Notes
Handwritten recognition	pixels, monomials, curves	Position, orientation, geometric shape descriptors	See examples in Chapters 5, 6
Collaborative filtering (movies)	Vector of movie ratings per use	Movie genres, year, actors	A detailed experiment is available in Section 6.3
Text classification	Words (bag of words)	Stem, semantic meaning, synonym, co-occurrence, frequency, part of speech, is stop word	See e.g., [84]

Table 1.2: Examples of meta-features for various learning tasks.

based clustering of features for handwritten Chinese recognition and other applications. Very recently, Lee *et. al.* [68] used meta-features for feature selection in related tasks. They assume that the meta-features are informative on the relevance of the features. Using a related task they model feature relevance as a function of the meta-features.

In this thesis, we formulate the use of meta-features as a method to learn and generalize from a subset of features to other features. For example, in feature selection task, we assume that we have a *training set of features*, which includes the quality of each feature and its meta-features. Based on this training set, we learn a function that maps from meta-features to quality. Then we use this function to predict the quality of new features. In this case, good generalization refers to our ability to predict this quality of new features. In Chapters 4 and 5 we show that this can be used to efficiently search for good features when we have huge set of candidate features.

In Chapter 6 we propose a learning framework that incorporates meta-features. In this framework we assume that a weight is assigned to each feature, as in linear discrimination. Then, instead of learning the weights directly, we learn a function that maps from the meta-feature space to the weights. Using this framework we analyze the theoretical advantages of incorporating meta-features into the learning, and derive practical algorithms that improve generalization by using meta-features.

1.4 Machine Learning and Neuroscience

The brain is the best known learner. The brain processes incoming data, extracts relevant patterns, learns to classify without supervision and learns to perform complicated tasks. The study of the brain as a computational machine is carried out at several levels. Cognitive psychology models the mental processes in the brain at an abstract level, and based on this model attempts to predict observable behavior. The physiology approach tries to understand how the brain works by looking inside the brain (e.g., by recording activity of neurons). Machine learning is related to the study of the brain in several ways. Some of the machine learning approaches are inspired, or even try to model, learning in the brain. Specifically, Artificial Neural Networks [42] is a field that try to solve learning problems using models that are based on abstractions of biological neural networks. In addition, dealing with the learning tasks that are similar to what the brain solves may teach us about how the brain works. On the theoretical level, machine learning can help us understand the expected mechanisms that are required for learning.

The human brain is composed of 10^{11} neurons. A neuron is composed of a cell body (soma) a dendrite and an axon. A typical neuron get inputs from about $10,000$ neurons through synaptic connections, most of which are located on the dendritic tree. For each synapse, the related input neuron is called the pre-synaptic neuron, and the output neuron is call post-synaptic neuron. The output of a neuron is an action potential, which is a short spike of voltage that propagates through its axon to other neurons. One of the simplest artificial models of the neuron is a computational unit that calculates the weighted sum of all of the inputs. The output is one if the resulting sum is above threshold, and zero otherwise. The weights in the artificial neurons are analogous to synaptic efficacy in biological neurons. One of the basic forms of learning at the neuronal level is synaptic plasticity, that is, a change in the synaptic efficacy between (biological) neurons. One of the most important neural learning mechanisms was introduced by Donald Hebb in 1949 [43]. Hebb stated that when two neurons are repeatedly activated together, the synaptic efficacy between them increases. This type of associative learning is known as Hebbian learning.

The exact nature of the Hebb rule in biological neurons changes as a function of the type

of neurons, and many variations of the basic rule have been discovered. One interesting variation is known as Spike Timing Dependent Plasticity (STDP), which states that change in synaptic efficacy can be positive or negative, depending on the exact timing difference between the pre-synaptic and the post-synaptic spike [71]. Evidence for another interesting modification of the basic Hebb rule was found by Engert and Bonhoeffer [28]. They found that weights (synaptic strength) to input synapses in close spatial proximity (up to $70\mu m$) tend to be potentiated together, even if some of the input neurons are not active at all. The authors called this a breakdown of synaptic specificity of long term potentiation (synaptic weight increase).

Many learning rules in machine learning and artificial neural networks are variations of Hebb rule; i.e., the change of weight is proportional to the correlation between pre- and post-synaptic signals. An example of such a rule is Oja's rule [77], which finds the largest eigenvector of the observations' covariance matrix. Other variations of the Hebb rule, when applied to networks can perform Principal Component Analysis (PCA) (see e.g., [78]). Hyvarinen and Oja [47] showed that Hebb-like learning rules can also carry out independent component analysis (ICA).

In Chapter 7, we point to a possible relation between using meta-features as prior knowledge as proposed in this thesis and learning mechanisms in the cortex. Specifically, we relate it to the breakdown of synaptic plasticity found by Engert and Bonhoeffer [28].

Chapter 2

Generalization in Clustering with Unobserved Features[1]

Data clustering can be defined as unsupervised classification of objects into groups based on their similarity (see Section 1.2.1). Often, it is desirable to have the clusters match some labels that are unknown to the clustering algorithm. In this context, good data clustering is expected to have homogeneous labels in each cluster, under some constraints on the number or complexity of the clusters. This can be quantified by mutual information (see e.g., [20]) between the objects' cluster identity and their (unknown) labels, for a given complexity of clusters. However, since the clustering algorithm has no access to the labels, it is unclear how it can optimize the quality of the clustering. Even worse, the clustering quality depends on the specific choice of the unobserved labels. For example, good document clustering with respect to topics is very different from clustering with respect to authors.

In our setting, instead of attempting to cluster by some arbitrary labels, we try to predict unobserved features from observed ones. In this sense our target labels are simply other features that happened to be unobserved. For example, when clustering fruits based on their observed features such as shape, color and size, the target of clustering is to match unobserved features such as nutritional value or toxicity. When clustering users based on

[1] Part of the results presented in this chapter were first presented as a talk in NIPS 2005 [60]. An extended version of the results was published in Journal of Machine Learning Research [62].

their movie ratings, the target of clustering is to match ratings of movies that were not rated, or not even created as yet.

In order to theoretically analyze and quantify this new learning scheme, we make the following assumptions. Consider a very large set of features, and assume that we observe only a *random* subset of n features, called *observed features*. The other features are called *unobserved features*. We assume that the random selection of observed features is made from some unknown distribution $\mathcal{D}$ and each feature is selected independently.[2]

The clustering algorithm has access only to the observed features of m instances. After clustering, one of the *unobserved* features is randomly selected to be the target label. This selection is done using the same distribution, $\mathcal{D}$, of the observed feature selection. Clustering performance is measured with respect to this feature. Obviously, the clustering algorithm cannot be directly optimized for this specific feature.

The question is whether we can optimize the *expected* performance on the unobserved features, based only on the observed features. The expectation is over the *random* selection of the unobserved target features. In other words, can we find the clustering that is most likely to match a randomly selected unobserved feature? Perhaps surprisingly, for a large enough number of observed features, the answer is yes. We show that for any clustering algorithm, the average performance of the clustering with respect to the observed and unobserved features is similar. Hence we can indirectly optimize clustering performance with respect to unobserved features by analogy with generalization in supervised learning. These results are universal and do not require any additional assumptions such as an underlying model or a distribution that created the instances.

In order to quantify these results, we define two terms: the average observed information and the expected unobserved information. Let T be the variable which represents the cluster for each instance, and $\{X_1, ..., X_L\}$ $(L \to \infty)$ the set of discrete random variables which denotes the features. The average observed information, denoted by I_{ob}, is the average mutual information between T and each of the observed features. In other words, if the observed features are $\{X_1, ..., X_n\}$ then $I_{ob} = \frac{1}{n}\sum_{j=1}^{n} I(T; X_j)$. The expected unobserved

[2]For simplicity, we also assume that the probability of selecting the same feature more than once is near zero.

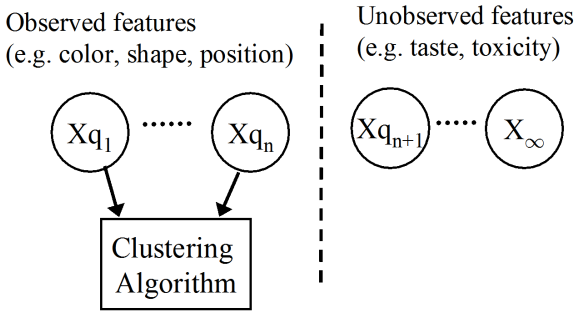


Figure 2.1: The learning scheme. The clustering algorithm has access to a random subset of features $(X_{q_1}, ..., X_{q_n})$ of m instances. The goal of the clustering algorithm is to assign a class label t_i to each instance, such that the expected mutual information between the class labels and a randomly selected *unobserved* feature is maximized.

information, denoted by I_{un}, is the *expected* value of the mutual information between T and a new *randomly* selected feature, i.e., $E_{q\sim\mathcal{D}}\{I(T;X_q)\}$. We are interested in cases where this new selected feature is most likely to be one of the unobserved features, and therefore we use the term unobserved information. Note that whereas I_{ob} can be measured directly, here we deal with the question of how to infer and maximize I_{un}.

Our main results consist of two theorems. The first is a generalization theorem. It gives an upper bound on the probability of a large difference between I_{ob} and I_{un} for all possible partitions. It also states a *uniform convergence in probability* of $|I_{ob} - I_{un}|$ as the number of observed features increases. Conceptually, the average observed information, I_{ob}, is analogous to the training error in standard supervised learning [112], whereas the unobserved information, I_{un}, is similar to the generalization error.

The second theorem states that under constraints on the number of clusters, and a large enough number of observed features, one can achieve nearly the best possible performance, in terms of I_{un}. Analogous to the principle of Empirical Risk Minimization (ERM) in statistical learning theory [112], this is done by maximizing I_{ob}.

We use our framework to analyze clustering by the maximum likelihood of the multinomial mixture model (also called Naive Bayes Mixture Model, see Figure 2.2 and Section 2.1.2). This clustering assumes a generative model of the data, where the *instances* are assumed to be sampled independently from a mixture of distributions, and for each such

distribution all features are independent. These assumptions are quite different from our assumptions of fixed instances and randomly observed features[3]. Nevertheless, in Section 2.1.2 we show that this clustering achieves nearly the best possible clustering in terms of information on unobserved features.

In Section 2.2 we extend our framework to distance-based clustering. In this case the measure of the quality of clustering is based on some distance function instead of mutual information. We show that the k-means clustering algorithm [69, 70] not only minimizes the observed intra-cluster variance, but also minimizes the unobserved intra-cluster variance; i.e., the variance of unobserved features within each cluster.

Table 2.1 summarizes the similarities and differences of our setting to that of supervised learning. The key difference is that in supervised learning, the set of features is fixed and the training instances are assumed to be randomly drawn from some distribution. Hence, the generalization is to new instances. In our setting, the set of instances is fixed, but the set of observed features is assumed to be randomly selected. Thus, the generalization is to new features.

Our new theorems are evaluated empirically in Section 2.3, on two different datasets. The first is a movie ratings dataset, where we cluster users based on their movie ratings. The second is a document dataset, with words as features. Our main point, however, is the new conceptual framework and not a specific algorithm or experimental performance.

Section 2.4 discusses related work and Section 2.5 presents conclusions and ideas for future research. Proofs of the theorems used in this chapter are provided in Section 2.6. A notation table is available in Section 2.7.

2.1 Feature Generalization of Information

In this section we analyze feature generalization in terms of mutual information between the clusters and the features. Consider a fixed set of m instances denoted by $\{\mathbf{x}[1], \ldots, \mathbf{x}[m]\}$. Each instance is represented by a vector of L features $\{x_1, \ldots, x_L\}$. The value of the qth feature of the jth instance is denoted by $x_q[j]$. Out of this set of features, n features are

[3]Note that in our framework, random independence refers to the *selection* of observed features, not to the feature values.

		Prediction of unobserved features	
	Supervised learning	Information	Distance-based
Training set	Randomly selected *instances*	n randomly selected features (observed features)	
Test set	Randomly selected *unlabeled instances*	Randomly selected *unobserved features*	
Hypothesis class	Class of functions from instances to labels	All possible partitions of m instances into k clusters	
Output of learning algorithm	Select hypothesis function	Cluster the *instances* into k clusters	
Goal	Minimize *expected* error on test set	Maximize *expected* information on *unobserved* features	Minimize *expected* intra-cluster variance of *unobserved* features
Assumption	Training and test instances are randomly and independently drawn from the same distribution	Observed and unobserved features are randomly and independently selected using the same distribution	
Strategy	Empirical Risk Minimization (ERM)	Observed Information Maximization (OIM)	Minimize observed intra-cluster variance
Related clustering algorithm		Maximum likelihood multinomial mixture model (Figure 2.2)	k-means
Good generalization	The training and test errors are similar	The observed and unobserved information are similar	The observed and unobserved intra-cluster variance are similar

Table 2.1: Analogy with supervised learning

randomly and independently selected according to some distribution $\mathcal{D}$. The n randomly selected features are the *observed features* (variables) and their indices are denoted by $\tilde{\mathbf{q}} = (q_1, ..., q_n)$, where $q_i \sim \mathcal{D}$. The ith observed feature of the jth instance is denoted by $x_{q_i}[j]$. After selecting the observed features, we also select *unobserved features* according to the same distribution $\mathcal{D}$. For simplicity, we assume that the total number of features, L, is large and the probability of selecting the same feature more than once is near zero (as in the case where $L \gg n^2$, where $\mathcal{D}$ is uniform distribution). This means that we can assume that a randomly selected unobserved feature is not one of the n observed features. It is important to emphasize that we have a fixed and finite set of instances; i.e., we do not need to assume that the m instances were drawn from any distribution. Only the features are randomly selected.

We further assume that each of the features is a discrete variable with no more than s different values[4]. The clustering algorithm clusters the instances into k clusters. The clustering is denoted by the function $\mathbf{t} : [m] \to [k]$ that maps each of the m instances to one of the k clusters. The cluster label of the jth instance is denoted by $\mathbf{t}(j)$. Our measures for the quality of clustering are based on Shannon's mutual information. Let random variable Z denote a number chosen uniformly at random from $\{1, \ldots, m\}$. We define the quality of clustering with respect to a single feature, q, as $I(\mathbf{t}(Z); x_q[Z])$, i.e., the empirical mutual information between the cluster labels and the feature.

Our measure of performance assumes that the number of clusters is predetermined. There is an obvious tradeoff between the preserved mutual information and the number of clusters. For example, one could put each instance in a different cluster, and thus get the maximum possible mutual information for all features. Obviously all clusters will be homogeneous with respect to all features but this clustering is pointless. Therefore, we need to have some constraints on the number of clusters.

Definition 2.1 *The **average observed information** of a clustering* $\mathbf{t}$ *and the observed*

[4]Since we are exploring an empirical distribution of a finite set of instances, dealing with continuous features is not meaningful.

features is denoted by $I_{ob}(\mathbf{t}, \tilde{\mathbf{q}})$ *and defined by*

$$I_{ob}(\mathbf{t}, \tilde{\mathbf{q}}) = \frac{1}{n} \sum_{i=1}^{n} I\left(\mathbf{t}(Z); x_{q_i}[Z]\right).$$

The ***expected unobserved information*** *of a clustering is denoted by* $I_{un}(t)$ *and defined by*

$$I_{un}(\mathbf{t}) = \mathbf{E}_{q \sim \mathcal{D}} \left\{ I\left(\mathbf{t}(Z); x_q[Z]\right) \right\}.$$

In general, I_{ob} is higher when clusters are more coherent; i.e., elements within each cluster have many identical observed features. I_{un} is high if there is a high probability that the clusters are informative on a randomly selected feature q (where $q \sim \mathcal{D}$). In the special case where the distribution $\mathcal{D}$ is uniform and $L \gg n^2$, I_{un} can also be written as the average mutual information between the cluster label and the unobserved features set; i.e., $I_{un} \approx \frac{1}{L-n} \sum_{q \notin \{q_1, \ldots, q_n\}} I\left(\mathbf{t}(Z); x_q[Z]\right)$. Recall that L is the total number of features, both observed and unobserved.

The goal of the clustering algorithm is to cluster the instances into k clusters that maximize the *unobserved* information, I_{un}. Before discussing how to maximize I_{un}, we first consider the problem of estimating it. Similar to the generalization error in supervised learning, I_{un} cannot be calculated directly in the learning algorithm, but we may be able to bound the difference between the observed information I_{ob} — our "training error" — and the unobserved information I_{un} — our "generalization error". To obtain generalization, this bound should be *uniform over all possible clusterings* with a high probability over the randomly selected features. The following lemma argues that *uniform convergence in probability* of I_{ob} to I_{un} always occurs.

Lemma 2.1 *With the definitions above,*

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)} \left\{ \sup_{\mathbf{t}:[m]\to[k]} \left| I_{ob}(\mathbf{t}, \tilde{\mathbf{q}}) - I_{un}(\mathbf{t}) \right| > \epsilon \right\} \leq 2e^{-2n\epsilon^2/(\log k)^2 + m \log k}, \quad \forall \epsilon > 0.$$

Proof: For any q,

$$0 \leq I\left(\mathbf{t}(Z); x_q[Z]\right) \leq H\left(\mathbf{t}(Z)\right) \leq \log k.$$

Using Hoeffding's inequality, for any specific (predetermined) clustering

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\left\{\left|I_{ob}\left(\mathbf{t},\tilde{\mathbf{q}}\right)-I_{un}\left(\mathbf{t}\right)\right|>\epsilon\right\}\leq 2e^{-2n\epsilon^2/(\log k)^2}.$$

Since there are at most k^m possible partitions, the union bound is sufficient to prove Lemma 2.1. □

Note that for any $\epsilon > 0$, the probability that $|I_{ob} - I_{un}| > \epsilon$ goes to zero, as $n \to \infty$. The convergence rate of I_{ob} to I_{un} is bounded by $O\left((\log k)/\sqrt{n}\right)$. As expected, this upper bound decreases as the number of clusters, k, decreases.

Unlike the standard bounds in supervised learning, this bound increases with the number of instances (m), and decreases with increasing numbers of observed features (n). This is because in our scheme the training size is not the number of instances, but rather the number of observed features (see Table 2.1). However, in the next theorem we obtain an upper bound that is independent of m, and hence is tighter for large m.

Consider the case where n is fixed, and m increases infinitely. We can select a random subset of instances of size m'. For large enough m', the empirical distribution of this subset is similar to the distribution over all instances. By fixing m', we can get a bound which is independent of m. Using this observation, the next theorem gives a bound that is independent of m.

Theorem 2.1 ***(Information Generalization)*** *With the definitions above,*

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\left\{\sup_{\mathbf{t}:[m]\to[k]}|I_{ob}(\mathbf{t},\tilde{\mathbf{q}})-I_{un}(\mathbf{t})|>\epsilon\right\}\leq 8(\log k)e^{-n\epsilon^2/\left(8(\log k)^2\right)+4sk\log k/\epsilon-\log\epsilon},\quad\forall\epsilon>0.$$

The proof of this theorem is given in Section 2.6.1. In this theorem, the bound does not depend on the number of instances, but rather on s which is the maximum alphabet size of the features. The convergence rate here is bounded by $O\left((\log k)/\sqrt[3]{n}\right)$. However, for relatively large n one can use the bound in Lemma 2.1, which converges faster.

As shown in Table 2.1, Theorem 2.1 is clearly analogous to the standard uniform convergence results in supervised learning theory (see e.g., [112]), where the random sample is replaced by our randomly selected features, the hypotheses are replaced by the clustering,

and I_{ob} and I_{un} replace the empirical and expected risks, respectively. The complexity of the clustering (our hypothesis class) is controlled by the number of clusters, k.

We can now return to the problem of specifying a clustering that maximizes I_{un}, using only the observed features. For reference, we will first define I_{un} of the best possible clustering.

Definition 2.2 ***Maximally achievable unobserved information:*** *Let $I^*_{un,k}$ be the maximum value of I_{un} that can be achieved by any partition into k clusters,*

$$I^*_{un,k} = \sup_{\mathbf{t}:[m]\to[k]} I_{un}(\mathbf{t}).$$

The clustering that achieves this value is called ***the best clustering****. The average observed information of this clustering is denoted by $I^*_{ob,k}$.*

Definition 2.3 ***Observed information maximization algorithm:*** *Let* IobMax *be any clustering algorithm that, based on the values of the observed features, selects a clustering $\mathbf{t}^{opt,ob} : [m] \to [k]$ having the maximum possible value of I_{ob}, i.e.,*

$$\mathbf{t}^{opt,ob} = \arg\max_{\mathbf{t}:[m]\to[k]} I_{ob}(\mathbf{t}, \tilde{\mathbf{q}}).$$

Let $\tilde{I}_{ob,k}$ be the average observed information of this clustering and $\tilde{I}_{un,k}$ be the expected unobserved information of this clustering, i.e.

$$\begin{aligned}\tilde{I}_{ob,k}(\tilde{\mathbf{q}}) &= I_{ob}\left(\mathbf{t}^{opt,ob}, \tilde{\mathbf{q}}\right),\\ \tilde{I}_{un,k}(\tilde{\mathbf{q}}) &= I_{un}\left(\mathbf{t}^{opt,ob}\right).\end{aligned}$$

The next theorem states that *IobMax* not only maximizes I_{ob}, but also maximizes I_{un}.

Theorem 2.2 ***(Achievability)*** *With the definitions above,*

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\left\{\tilde{I}_{un,k}(\tilde{\mathbf{q}}) \le I^*_{un,k} - \epsilon\right\} \le 8(\log k)e^{-n\epsilon^2/\left(32(\log k)^2\right)+8sk\log k/\epsilon-\log(\epsilon/2)}, \quad \forall\epsilon > 0. \tag{2.1}$$

Proof: We now define a *bad clustering* as a clustering whose expected unobserved information satisfies $I_{un} \le I^*_{un,k} - \epsilon$. Using Theorem 2.1, the probability that $|I_{ob} - I_{un}| > \epsilon/2$ for any of the clusterings is upper bounded by the right term of equation 2.1. If for all

clusterings $|I_{ob} - I_{un}| \leq \epsilon/2$, then surely $I^*_{ob,k} \geq I^*_{un,k} - \epsilon/2$ (see Definition 2.2) and I_{ob} of all bad clusterings satisfies $I_{ob} \leq I^*_{un,k} - \epsilon/2$. Hence the probability that a bad clustering has a higher average observed information than the best clustering is upper bounded as in Theorem 2.2. □

For small m, a tighter bound, similar to that of Lemma 2.1 can easily be formulated.

As a result of this theorem, when n is large enough, even an algorithm that knows the value of *all* features (observed and unobserved) cannot find a clustering which is significantly better than the clustering found by the $IobMax$ algorithm. This is demonstrated empirically in Section 2.3.

Informally, this theorem means that for a large number of features we can find a clustering that is informative on unobserved features. For example, clustering users based on similar ratings of current movies are likely to match future movies as well (see Section 2.3).

In the generalization and achievability theorems (Theorems 2.1, 2.2) we assumed that we are dealing only with hard clustering. In Section 2.6.2 we show that the generalization theorem is also applicable to soft clustering; i.e., assigning a probability distribution among the clusters to each instance. Moreover, we show that soft clustering is not required to maximize I_{ob}, since its maximum value can be achieved by hard clustering.

2.1.1 Toy Examples

In the first two examples below, we assume that the instances are drawn from a given distribution (although this assumption is not necessary for the theorems above). We also assume that the number of instances is large, so the empirical and the actual distributions of the instances are about the same.

Example 2.1 *Let $X_1, ..., X_\infty$ be Bernoulli($\frac{1}{2}$) random variables, such that all variables with an even index are equal to each other ($x_2 = x_4 = x_6 = ...$), and all variables with an odd index are independent of each other and of all other variables. If the number of randomly observed features is large enough we can find a clustering rule with two clusters such that $I_{ob} \cong \frac{1}{2}$. This is done by assigning the cluster labels based on the set of features that are correlated, e.g., $\mathbf{t}(i) = x_2[i] + 1 \quad \forall i$, assuming that x_2 is one of the observed features.*

$I(\mathbf{t}(Z); x_i(Z))$ is one for even i, and zero for odd i. For large n, the number of randomly selected features with odd indices and even indices[5] *is about the same (with high probability), and hence $I_{ob} \cong \frac{1}{2}$. For this clustering rule $I_{un} \cong \frac{1}{2}$, since half of the unobserved features match this clustering (all features with an even index).*

Example 2.2 *When $X_1, ..., X_\infty$ are i.i.d. (independent and identically distributed) Bernoulli($\frac{1}{2}$) random variables, $I_{un} = 0$ for any clustering rule, regardless of the number of observed features. For a finite number of clusters, I_{ob} will necessarily approach zero as the number of observed features increases. More specifically, if we use two clusters, where the clustering is determined by one of the observed features (i.e., $\mathbf{t}(i) = x_j(i)$, where x_j is an observed feature), then $I_{ob} = \frac{1}{n}$ (because $I(\mathbf{t}(Z); x_j(Z)) = 1$ and $I(\mathbf{t}(Z); x_l(Z)) = 0$ for $l \neq j$).*

Example 2.3 *Clustering fruits based on the observed features (color, size, shape etc.) also matches many unobserved features. Indeed, people clustered fruits into oranges, bananas and others (by giving names in the language) long before vitamin C was discovered. Nevertheless, this clustering was very informative about the amount of vitamin C in fruits, i.e., most oranges have similar amounts of vitamin C, which is different from the amount in bananas.*

Based on the generalization theorem, we now suggest a qualitative explanation of why clustering into bananas and oranges provides relatively high information on unobserved features, while clustering based on position (e.g., right/left in the field of view) does not. Clustering into bananas and oranges contains information on many observed features (size, shape, color, texture), and thus has relatively large I_{ob}. By the generalization theorem, this implies that it also has high I_{un}. By contrast, a clustering rule which puts all items that appeared in our right visual field in one cluster, and the others in a second cluster, has much smaller I_{ob} (since it does not match many observed features), and indeed it is not predictive about unobserved features.

Example 2.4 *As a negative example, if the type of observed features and the target unobserved features are very different, our assumptions do not hold. For example, when the observations are pixels of an image, and the target variable is the label of the image, we cannot generalize from information about the pixels to information about the label.*

[5] Note that the indices are arbitrary. The learning algorithm does not use the indices of the features.

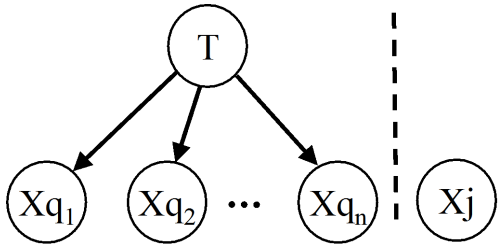


Figure 2.2: The Bayesian network [81] of the multinomial mixture model. The observed random variables $\{X_{q_1}, \ldots, X_{q_n}\}$ are statistically independent given the parent hidden random variable, T. This parent variable represents the cluster label. Although the basic assumptions of the multinomial mixture model are very different from ours, Theorem 2.3 tells us that this method of clustering generalizes well to unobserved features.

2.1.2 Feature Generalization of Maximum Likelihood Multinomial Mixture Models

In the framework of Bayesian graphical models, the multinomial mixture model is commonly used. The assumption of this model is that all features are conditionally independent, given the value of some hidden variable, i.e.

$$\Pr\left(T = t, x_{q_1}, \ldots, x_{q_n}\right) = \Pr\left(T = t\right) \prod_{r=1}^{n} \Pr\left(x_{q_r} \,|T = t\right).$$

where T denotes the hidden variable. The Bayesian network [81] of this model is given in Figure 2.2. This standard model does not assume the existence of unobserved features, so we use the notation $x_{q_1}, \ldots, x_{q_n}$ to denote the observed features which are used by the model. The set of instances are assumed to be drawn from such a distribution, with unknown parameters. Given the set of instances, the goal is to learn the distributions $\Pr\left(T = t\right)$ and $\Pr\left(x_{q_r} \,|T = t\right)$ that maximizes the probability of the observation, i.e., values of the instances. This maximum-likelihood problem is typically solved using an EM algorithm [24] with a fixed number of clusters (values of T). The output of this algorithm includes a soft clustering of all instances; i.e., $P(T|Y)$, where Y denotes the index of the instance.

In the following theorem we analyze the feature generalization properties of soft clustering by the multinomial mixture model. We show that under some technical conditions, it pursues nearly the same goal as *IobMax* algorithm (Definition 2.3), i.e., maximizing $\sum_j I\left(\mathbf{t}(Z); X_{q_j}(Z)\right)$.

Theorem 2.3 *Let $I_{ob,ML,k}$ be the observed information of clustering achieved by the maximum likelihood solution of a multinomial mixture model for k clusters. Then*

$$I_{ob,ML,k} \geq \tilde{I}_{ob,k} - \frac{2H(T)}{n},$$

where $\tilde{I}_{ob,k}$ is the observed information achieved by the IobMax clustering algorithm (Definition 2.3).

Proof:

Elidan and Friedman [27] showed that learning a hidden variable can be formulated as the multivariate information bottleneck [31]. Based on their work, in Section 2.6.3 we show that maximizing the likelihood of observed variables is equivalent to maximizing $\sum_{j=1}^{n} I(T;X_{q_j}) - I(T;Y)$. Using our notations, this is equivalent to maximizing $I_{ob} - \frac{1}{n}I(T;Y)$. Since $I(T;Y) \leq H(T)$, the difference between maximizing I_{ob} and $I_{ob} - \frac{1}{n}I(T;Y)$ is at most $2H(T)/n$. □

The meaning of Theorem 2.3 is that for large n, finding the maximum likelihood of mixture models is similar to finding the maximum unobserved information. Thus the standard EM-algorithm for maximum likelihood of mixture models can be viewed as a form of the *IobMax* algorithm[6].

The standard mixture model assumes a generative model for generating the instances from some distribution, and finds the maximum likelihood of this model. This model does not assume anything about the selection of features or the existence of unobserved features. Our setup assumes that the instances are fixed and the observed features are randomly selected and we try to maximize information on unobserved features. Interestingly, while the initial assumptions are quite different, the results are nearly equivalent. We show that finding the maximum likelihood of the mixture model indirectly predicts unobserved features as well.

The maximum likelihood mixture model was used by Breese et al. [14] to cluster users by their voting on movies. This clustering is used to predict the rating of new movies. Our analysis shows that for a large number of rated (observed) movies, it is nearly the best

[6] Ignoring the fact that achieving a global maximum is not guaranteed.

clustering method in terms of information on new movies.

The multinomial mixture model is also used for learning with labeled and unlabeled instances, and is considered a baseline method (see Section 2.3 in [90]). The idea is to cluster the instances based on their features. Then, the prediction of a label for an unlabeled instance is estimated from the labels of other instances in the same cluster. From our analysis, this is nearly the best clustering method for preserving information on the label, assuming that the label is yet another feature that happened to be unobserved in some instances. This provides another interpretation regrading the hidden assumption of this clustering scheme for labeled and unlabeled data.

2.2 Distance-Based Clustering

In this section we extend the framework and include analysis of feature generalization bounds for distance-based clustering. We apply this to analyze feature generalization of the k-means clustering algorithm (See Table 2.1). The setup in this section is the same as the setup defined in Section 2.1 except as described below. We assume that we have a distance function, denoted by f, that measures the distance for every two values of any of the features. We assume that f has the following properties:

$$0 \leq f(x_q[j], x_q[l]) \leq c. \quad \forall q, j, l, \tag{2.2}$$

$$f(a, a) = 0. \qquad \forall a, \tag{2.3}$$

$$f(a, b) = f(b, a). \qquad \forall a, b, \tag{2.4}$$

where c is some positive constant. An example of such a function is the square error, i.e., $f(a, b) = (a - b)^2$, where we assume that the value of all features is bounded as follows $|x_q[j]| \leq \sqrt{c}/2$ $(\forall q, j)$, for some constant c. The features themselves can be discrete or continuous. Although we do not directly use the function f in the definitions of the theorems in the following section, it is required for their proofs (Section 2.6.4).

2.2.1 Generalization of Distance-Based Clustering

As in Section 2.1, we have a set of m fixed instances $\{\mathbf{x}[1], ..., \mathbf{x}[m]\}$, and the clustering algorithm clusters these instances into k clusters. For better readability, in this section the partition is denoted by $\{C_1, ..., C_k\}$. $|C_r|$ denotes the size of the rth cluster.

The standard objective of the k-means algorithm is to achieve minimum intra-cluster variance, i.e., minimize the function

$$\sum_{r=1}^{k} \sum_{j \in C_r} |\mathbf{x}[j] - \mu_r|^2 ,$$

where μ_r is the mean point of all instances in the rth cluster.

In our setup, however, we assume that the clustering algorithm has access only to the *observed* features over the m instances. The goal of clustering is to achieve minimum intra-cluster variance of the *unobserved* features. To do so, we need to generalize from the observed to the unobserved intra-class variance. To formalize this type of generalization, let's first define these variances formally.

Definition 2.4 *The* ***observed intra-cluster variance*** $D_{ob}\{C_1, ..., C_k\}$ *of a clustering* $\{C_1, ..., C_k\}$ *is defined by*

$$D_{ob}\{C_1, ..., C_k\} = \frac{1}{nm} \sum_{r=1}^{k} \sum_{j \in C_r} \sum_{i=1}^{n} (x_{q_i}[j] - \mu_{q_i}[r])^2 ,$$

where $\mu_q[r]$ *is the mean of feature* q *over all instances in cluster* r*, i.e.,*

$$\mu_q[r] = \frac{1}{|C_r|} \sum_{l \in C_r} x_q[l].$$

In other words, D_{ob} is the average square distance of each observed feature from the mean of the value of the feature in its cluster. The average is over all observed features and instances. The k-means algorithm minimizes the observed intra-cluster variance.

Definition 2.5 *The* ***expected unobserved intra-cluster variance*** $D_{un}\{C_1, ..., C_k\}$ *is defined by*

$$D_{un}\{C_1, ..., C_k\} = \frac{1}{m} \sum_{r=1}^{k} \sum_{j \in C_r} \mathbf{E}_{q \sim \mathcal{D}} (x_q[j] - \mu_q[r])^2 .$$

D_{ob} and D_{un} are the distance-based variables analogous to I_{ob} and I_{un} defined in Section 2.1. In our setup, the goal of the clustering algorithm is to create clusters with minimal unobserved intra-class variance (D_{un}). As in the case of information-based clustering, we first consider the problem of estimating D_{un}. Before presenting the generalization theorem for distance-based clustering, we need the following definition.

Definition 2.6 *Let α be the ratio between the size of smallest cluster and the average cluster size, i.e.,*

$$\alpha\left(\{C_1,...,C_k\}\right) = \frac{\min_r |C_r|}{m/k}.$$

Now we are ready for the generalization theorem for distance-based clustering.

Theorem 2.4 *With the above definitions, if $|x_q[j]| \leq R$ for every q, j then for every $\alpha_c > 0$, $\epsilon > 0$,*

$$\Pr_{\{q_1,...,q_n\}}\left\{\sup_{\alpha(\{C_1,...,C_k\})\geq\alpha_c} |D_{ob}\left\{C_1,...,C_k\right\} - D_{un}\left\{C_1,...,C_k\right\}| \leq \epsilon\right\} \geq 1-\delta,$$

where

$$\delta = \frac{8k}{\alpha_c} e^{-n\epsilon^2/8R^4 + \log\left(R^2/\epsilon\right)}.$$

The proof of this theorem is given in Section 2.6.4. Theorem 2.4 is a special case of a more general theorem (Theorem 2.6) that we present in Section 2.6.4. Theorem 2.6 can be applied to other distance-based metrics, beyond the intra-cluster variance defined in Definition 2.5.

Note that for any $\epsilon > 0$, the probability that $|D_{ob} - D_{un}| \leq \epsilon$ goes to one, as $n \to \infty$. The convergence rate of D_{ob} to D_{un} is bounded by $O(1/\sqrt{n})$. As expected, for a fixed value of δ the upper bound on $|D_{ob} - D_{un}|$ decreases as the number of clusters, k, decreases.

Theorem 2.4 bounds the difference between observed and unobserved variances. We now use it to find a clustering that minimizes the expected unobserved intra-cluster variance, using only the observed features.

Theorem 2.5 *Let $\left\{C_1^{opt},...,C_k^{opt}\right\}$ be the clustering that achieves the minimum unobserved intra-cluster variance under the constraint $\alpha\left(\{C_1,...,C_k\}\right) \geq \alpha_c$ for some constant $0 < \alpha_c \leq 1$, i.e.*

$$\left\{C_1^{opt},...,C_k^{opt}\right\} = \arg\min_{\{C_1,...,C_k\}:\alpha\geq\alpha_c} D_{un}\left\{C_1,...,C_k\right\}$$

and let D_{un}^{opt} *the best unobserved intra-cluster variance, be defined by* $D_{un}^{opt} = D_{un}\left\{C_1^{opt}, ..., C_k^{opt}\right\}$.

Let $\left\{\hat{C}_1^{opt}, ..., \hat{C}_k^{opt}\right\}$ *be the clustering with the minimum observed intra-cluster variance, under the same constraint on* $\alpha\left(\{C_1, ..., C_k\}\right)$, *i.e.,*

$$\left\{\hat{C}_1^{opt}, ..., \hat{C}_k^{opt}\right\} = \arg \min_{\alpha(\{C_1,...,C_k\}) \geq \alpha_c} D_{ob}\left\{C_1, ..., C_k\right\}$$

and let $\hat{D}_{un}^{opt}$ *be the unobserved intra-cluster variance of this clustering, i.e.,* $\hat{D}_{un}^{opt} = D_{un}\left\{\hat{C}_1^{opt}, ..., \hat{C}_k^{opt}\right\}$.

For any $\epsilon > 0$,

$$\Pr_{\{q_1,...,q_n\}}\left\{\hat{D}_{un}^{opt} \leq D_{un}^{opt} + \epsilon\right\} \geq 1 - \delta, \tag{2.5}$$

where

$$\delta = \frac{16k}{\alpha_c} e^{-n\epsilon^2/32R^4 + \log\left(R^2/\epsilon\right)}. \tag{2.6}$$

Proof: We now define a *bad clustering* as a clustering whose expected unobserved intra-cluster variance satisfies $D_{un} > D_{un}^{opt} + \epsilon$. Using Theorem 2.4, the probability that $|D_{ob} - D_{un}| \leq \epsilon/2$ for all possible clusterings (under the constraint on α) is at least $1 - \delta$, where δ defined in Eq. 2.6. If for all clusterings $|D_{ob} - D_{un}| \leq \epsilon/2$, then surely $D_{ob}\left\{C_1^{opt}, ..., C_k^{opt}\right\} \leq D_{un}^{opt} + \epsilon/2$ and D_{ob} of all bad clusterings satisfies $D_{ob} > D_{un}^{opt} + \epsilon/2$. Hence the probability that any of the bad clusterings has a lower observed intra-cluster variance than the best clustering is upper bounded by δ. Therefore, with a probability of at least $1 - \delta$ none of the bad clusterings is selected by an algorithm that selects the clustering with the minimum D_{ob}. □

We cannot directly calculate the unobserved intra-cluster variance. However, Theorem 2.5 means that an algorithm that selects the clustering with the minimum observed intra-cluster variance indirectly finds the clustering with nearly minimum unobserved intra-cluster variance.

In general, minimizing observed intra-cluster variance is the optimization objective of k-means. Hence, k-means indirectly minimizes the unobserved intra-cluster variance. This means that in our context, k-means can be viewed as an analog to the empirical risk minimization (ERM) in the standard supervised learning context. We minimize the observed variance (training error) in order to indirectly minimize the expected unobserved variance (test error).

k-means is used in collaborative filtering such as movie rating predictions for grouping users based on similar ratings (see e.g., [72]). After clustering, we can predict ratings of a new movie based on the ratings of a few users for this movie. If the intra-cluster variance of a new, previously unobserved movie is small, then we can estimate the rating of one user from the average ratings of other users in the same cluster.

An experimental illustration of the behavior of the observed and unobserved intra-cluster variances for k-means is available in Section 2.3.1.

2.3 Empirical Evaluation

In this section we test experimentally the generalization properties of *IobMax* and the k-means clustering algorithm for a finite number of features. For *IobMax* we examine the difference between I_{ob} and I_{un} as a function of the number of observed features, and number of clusters used. We also compare the value of I_{un} achieved by the *IobMax* algorithm to I^*_{un}, which is the maximum achievable I_{un} (see Definition 2.2). Similarly, for distance-based clustering we use k-means to examine the behavior of the observed and unobserved intra-cluster variances (see Definitions 2.4, 2.5).

The purpose of this section is *not* to suggest new algorithms for collaborative filtering or compare it to other methods, but simply to illustrate our new theorems on empirical data.

2.3.1 Collaborative Filtering

In this section, our evaluation uses a dataset typically employed for collaborative filtering. Collaborative filtering refers to methods of making predictions about a user's preferences, by collecting the preferences of many users. For example, collaborative filtering for movie ratings can make predictions about the rating of movies by a user given a partial list of ratings from this user and many other users. Clustering methods are used for collaborative filtering by clustering users based on the similarity of their ratings (see e.g., [72, 109]).

In our setting, each user is described as a vector of movie ratings. The rating of each movie is regarded as a feature. We cluster users based on the set of observed features, i.e., rated movies. In our context, the goal of the clustering is to maximize the information

between the clusters and unobserved features, i.e., movies that have not yet been rated by any of the users. These can be movies that have not yet been made. By Theorem 2.2, given a large enough number of rated movies, we can achieve the best possible clustering of users with respect to unseen movies. In this region, no additional information (such as user age, taste, rating of more movies) beyond the observed features can improve the unobserved information, I_{un}, by more than some small ϵ.

For distance-based clustering, we cluster the users by the k-means algorithm based on a subset of features (movies). As we show in Section 2.2.1 the goal of k-means is to minimize the observed intra-cluster variance. From Theorem 2.5, this indirectly minimizes the unobserved intra-cluster variance as well. Here we empirically evaluate this type of generalization.

Dataset. We use MovieLens (www.movielens.umn.edu), which is a movie rating data set. It was collected and distributed by GroupLens Research at the University of Minnesota. It contains approximately 1 million ratings of 3900 movies by 6040 users. Ratings are on a scale of 1 to 5. We use only a subset consisting of 2400 movies rated by 4000 users (or 2000 by 2000 for distance-based clustering). In our setting, each instance is a vector of ratings $(x_1, ..., x_{2400})$ by a specific user. Each movie is viewed as a feature, where the rating is the value of the feature.

Experimental Setup. We randomly split the 2400 movies into two groups, denoted by "A" and "B", of 1200 movies (features) each. We use a subset of the movies from group "A" as observed features and all movies from group "B" as the unobserved features. The experiment was repeated with 20 random splits and the results averaged. We estimate I_{un} by the mean information between the clusters and ratings of movies from group "B". We use a uniform distribution of feature selection ($\mathcal{D}$), and hence I_{un} can be estimated as the average information on the unobserved features, i.e., $I_{un} = \frac{1}{1200} \sum_{j \in B} I(T; X_j)$. A similar setup is used for the distance-based clustering (with two groups of 1000 movies).

Handling Missing Values. In this data set, most of the values are missing (not rated). For information based-clustering, we handle this by defining the feature variable as 1,2,...,5 for the ratings and 0 for a missing value. We maximize the mutual information based on the empirical distribution of values that are present, and weight it by the probability

of presence for this feature. Hence, $I_{ob} = \sum_{j=1}^{n} P(X_j \neq 0) I(T; X_j | X_j \neq 0)$ and $I_{un} = E_j \{P(X_j \neq 0) I(T; X_j | X_j \neq 0)\}$. The weighting prevents overfitting to movies with few ratings. Since the observed features are selected at random, the statistics of missing values of the observed and unobserved features are the same. Hence, all our theorems are applicable to these definitions of I_{ob} and I_{un} as well.

In order to verify that the estimated mutual information is not just an artifact of the finite sample size, we tested the mutual information after random permutation of ratings of each movie among users. Indeed, the resulting mutual information was significantly lower in the case of random permutation.

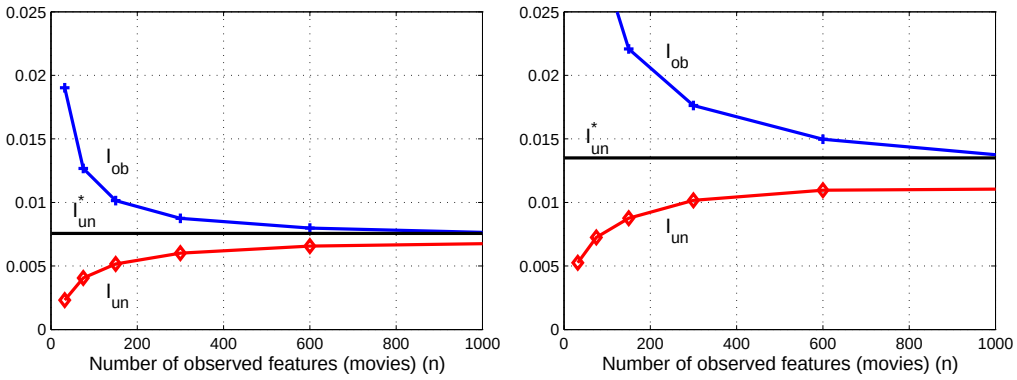
For the distance based clustering, we handle missing data by defining a default square distance between a feature and the cluster center where one (or two) of the values is missing. We select this default square distance to be the average variance of movie ratings (which is about 0.9).

Greedy *IobMax* Algorithm

For information-based clustering, we cluster the users using a simple greedy clustering algorithm (see Algorithm 1). The input to the algorithm is all users, represented solely by the observed features. Since this algorithm can only find a local maximum of I_{ob}, we ran the algorithm 10 times (each used a different random initialization) and selected the results that had a maximum value of I_{ob}.

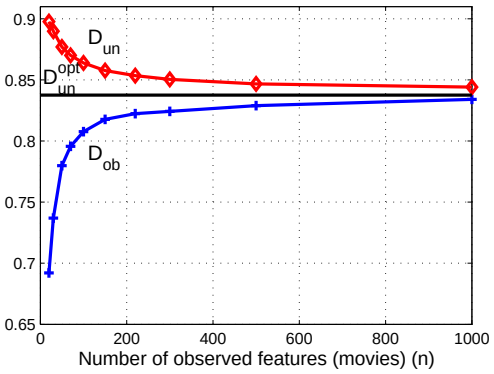
In our experiment, the number of observed features is large. Therefore, based on Theorem 2.3, the greedy *IobMax* can be replaced by the standard EM-algorithm which finds the maximum likelihood for multinomial mixture models. Although, in general, this algorithm finds soft clustering, in our case the empirical result clusterings are not soft, i.e., one cluster is assigned to each instance (see Section 2.6.2). As expected, the results of both algorithms are nearly the same.

In order to estimate $I^*_{un,D}$ (see Definition 2.2), we also ran the same algorithm when all the features were available to the algorithm (i.e., also features from group "B"). In this case the algorithm tries directly to find the clustering that maximizes the mean mutual

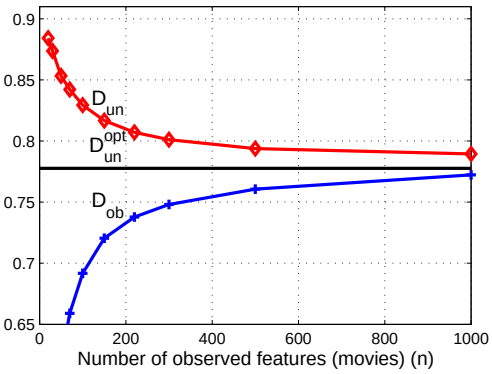


(a) 2 Clusters

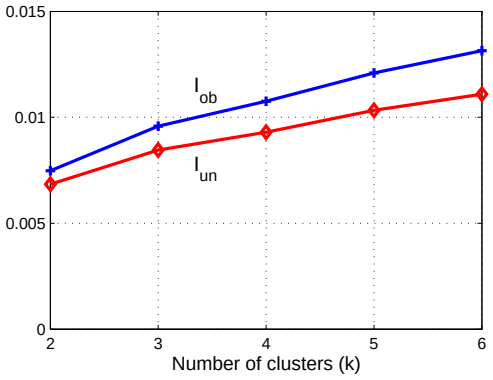
(b) 6 Clusters



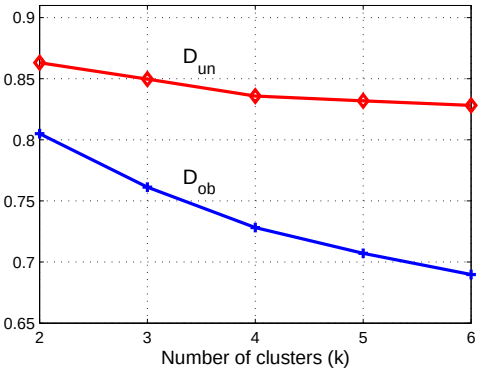
(c) 2 Clusters



(d) 6 Clusters



(e) Fixed number of features (n=1200)



(f) Fixed number of features (n=100)

Figure 2.3: Feature generalization as a function of the number of training features (movies) and the number of clusters. (a) (b) and (e) show the observed and unobserved information for various numbers of features and clusters (high is good). The overall mean information is low, since the rating matrix is sparse. (c) (d) and (f) shows the observed and unobserved intra-cluster variance (low is good). In these figures, the variance is only calculated on values which are not missing. Figures (e) and (f) show the effect of the number of clusters when the number of features is fixed.

Algorithm 1 A simple greedy *IobMax* algorithm

1. Assign a random cluster to each of the instances.
2. For $r = 1$ to R (where R is the upper limit on the number of iterations)
   (a) For each instance,
      i. Calculate I_{ob} for all possible clustering assignments of the current instance.
      ii. Choose the clustering that maximizes I_{ob}.
   (b) Exit if the clusters of all documents do not change.

information on features from group "B".

Results

The results are shown in Figure 2.3. It is clear that as the number of observed features increases, I_{ob} decreases while I_{un} increases (see Figure 2.3(a,b)). When there is only one feature, two clusters can contain all the available information on this feature (e.g., by assigning $t(j) = x_{q_1}[j]$), so I_{ob} reaches its maximum value (which is $H(X_{q_1}[Z])$). As the number of observed features increases, we cannot preserve all the information on all the features in a few clusters, so the observed mutual information (I_{ob}) decreases. On the other hand, as the number of observed features increases, the cluster variable, $T = t(Z)$, captures the structure of the distribution (users' tastes), and hence contains more information on unobserved features. The generalization theorem (Theorem 2.1) tells us that the difference between I_{un} and I_{ob} will approach zero as the number of observed features increases. This is similar to the behavior of training and test errors in supervised learning. Informally, the achievability theorem (Theorem 2.2) tells as that for a large enough number of observed features, even though our clustering algorithm is based only on observed features, it can achieve nearly the best possible clustering, in terms of I_{un}. This can be seen in Figures 2.3 (a,b), where I_{un} approaches I^*_{un}, which is the unobserved information of the best clustering (Definition 2.2). As the number of clusters increases, both I_{ob}, I_{un} increase (Figure 2.3e), but the difference between them also increases.

Similar results were obtained for distance based clustering. The goal here is to minimize the unobserved intra-cluster variance (D_{un}), and this is done by minimizing the observed

intra-cluster variance (D_{ob}). As discussed in Section 2.2.1, this can be achieved by k-means[7]. Again, for a small numbers of features (n) the clustering overfits the observed features, i.e., the D_{ob} is relatively low but D_{un} is large. However, for large n, D_{un} and D_{ob} approach each other and both of them approach the unobserved intra-cluster variance of the best possible clustering (D_{un}^{opt}) as expected from Theorem 2.5. When the number of clusters increases, both D_{ob} and D_{un} decrease, but the difference between them increases.

2.3.2 Words and Documents

In this section we repeat the information-based clustering experiment, but this time for document clustering with words as features. We show how clustering which is based on a subset of words (observed words) is also informative about the unobserved words. The obtained curves of information vs. number of features are similar to those in the previous section. However, in this section we also examine the resulting clustering (Table 2.2) to get a better intuition as to how this generalization occurs.

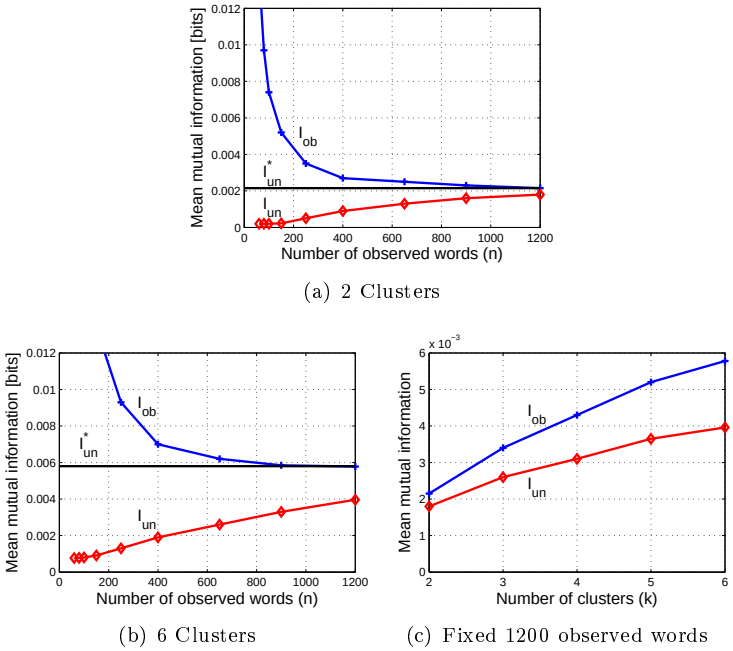
Dataset. We use the 20-newsgroups (20NG) corpus, collected by Lang [64]. This collection contains about 20,000 messages from 20 Usenet discussion groups, some of which have similar topics.

Preprocessing. In order to prevent effects caused by different document lengths, we truncate each document to 100 words (by randomly selecting 100 words), and ignore documents which consist of fewer than 100 words. We use the "bag of words" representation: namely we convert each document into a binary vector $(x_1, x_2, ...)$, where each element in the vector represents a word, and equals one if the word appears in the document and zero otherwise. We select the 2400 words whose corresponding X_i has maximum entropy[8], and remove all other words. After this preprocessing each document is represented by a vector $(x_1, ..., x_{2400})$.

Experimental setup. We randomly split the 2400 words into two groups of 1200 words (features) each. The groups were called "A" and "B". We use a variable number of words (1

[7] Since k-means does not necessarily find the global optimum, we ran it 20 times with different initialization points, and chose the results with minimal observed intra-cluster variance. This does not guarantee a global optimum, but no other tractable algorithm is available today to achieve global optima.

[8] In other words, the probability of the word appearing in a document is not near zero or near one.



(a) 2 Clusters

(b) 6 Clusters

(c) Fixed 1200 observed words

Figure 2.4: I_{ob}, I_{un} and I^*_{un} per number of training words and clusters. In (a) and (b) the number of words is variable, and the number of clusters is fixed. In (c) the number of observed words is fixed (1200), and the number of clusters is variable. The overall mean information is low, since a relatively small number of words contributes to the information (see Table 2.2)

to 1200) from group "A" as observed features. All the features from group "B" are used as the unobserved features. We repeat the test with 10 random splits and present the mean results.

Results

The results are shown in Figure 2.4 and Table 2.2. The qualitative explanation of the figure is the same as for collaborative filtering (see Section 2.3.1 and Figure 2.3). Table 2.2 presents a list of the most informative words, and their appearance in each cluster. This helps understand the way clustering learned from observed words matches unobserved words. We can see, for example, that although the word "player" is not part of the inputs to the clustering algorithm, it appears much more in the first cluster than in other clusters.

	Word	t=1	t=2	t=3
Observed words	he	0.47	0.04	0.25
	game	0.23	0.01	0
	team	0.20	0	0
	x	0.01	0.15	0.01
	hockey	0.11	0	0
	jesus	0.01	0	0.09
	christian	0	0	0.08
	use	0.04	0.21	0.09
	file	0	0.08	0.01
Unobserved words	god	0.02	0.01	0.15
	players	0.13	0	0
	baseball	0.10	0	0
	window	0	0.10	0
	server	0	0.06	0

Table 2.2: Probability of a word appearing in a document from each cluster. Each column in the table represents a cluster (total of three clusters), and the numbers are the probabilities that a document from a cluster will contain the word (e.g., The word "he" appears in 47% of the documents from cluster 1). The results presented here are for learning from 1200 observed words, but only a few of the most informative words appear in the table.

Intuitively this can be explained as follows. The algorithm finds clusters that are informative on many observed words together, and thus matches the co-occurrence of words. This clustering reveals the hidden topics of the documents (sports, computers and religious), and these topics contain information on the unobserved words. We see that generalization to unobserved features can be explained from a standpoint of a generative model (a hidden variable which represents the topics of the documents) or from a statistical point of view (relationship between observed and unobserved information). In Section 2.5 we further discuss this dual view.

2.4 Related work

In the framework of learning with labeled and unlabeled data (see Section 1.3.1), a fundamental issue is the link between the marginal distribution $P(\mathbf{x})$ over instances $\mathbf{x}$ and the conditional $P(y|\mathbf{x})$ for the label y [103]. From this point of view our approach assumes that

the label, y, is a feature in itself.

In the context of supervised learning, Ando and Zhang [1] proposed an approach that regards features in the input data as labels in order to improve prediction in the target supervised problem. Their idea is to create many auxiliary problems that are related to the target supervised problem. They do this by masking some features in the input data, i.e., making them unobserved features, and training classifiers to predict these unobserved features from the observed features. Then they transfer the knowledge acquired from these related classification tasks to the target supervised problem. A similar idea was used by Caruana and Sa [15] in supervised training of a neural net. The authors used some features as extra outputs of the neural net, rather than inputs, and show empirically that this can improve the classifier performance. In our framework, this could be interpreted as follows. We regard the label as a feature, and hence we can learn from prediction of these features to the prediction of the label. Loosely speaking, if we successfully predict many such features by the classifier, we expect to generalize better to the target feature (label).

2.5 Discussion

We introduce a new learning paradigm: clustering based on observed features that generalizes to unobserved features. Our main results include two theorems that demonstrate how, without knowing the value of the unobserved features, one can estimate and maximize information between the clusters and the unobserved features. Using this framework we analyze feature generalization in the Maximum Likelihood Multinomial Mixture Model (Figure 2.2). The multinomial mixture model is a generative probabilistic model which approximates the probability distribution of the observed data ($\mathbf{x}$), from a finite sample of instances. Our model does not assume any distribution that generated the instances, but instead assumes that the set of observed features is simply a random subset of features. Then, using statistical arguments we show that we can cluster by the unobserved features. Despite the very different assumptions of these models, we show that clustering by multinomial mixture models is nearly optimal in terms of maximizing information on unobserved features. However, to analyze and quantify this generalization our framework is required.

This dual view on the multinomial mixture model can also be applied to two different approaches that may explain our "natural clustering" of objects in the world (e.g., assigning object names in language). Let's return to our clustering of bananas and oranges (Example 2.3). From a generative point of view, we find a model with the cluster labels bananas and oranges as values of a hidden variable that created the distribution. This means that we have a mixture of two distributions, each related to one object type that is assigned to a different cluster. Since we have two types of objects (distributions), we expect that their unobserved features will correspond to these two types as well. However, the generative model does not quantify this expectation. In our framework, we view fruits in the world, and cluster them based on some kind of *IobMax* algorithm; i.e., we find a representation (clustering) that contains significant information on as many observed features as possible, while still remaining simple. From our generalization theorem (Theorem 2.1), such a representation is expected to contain information on other rarely viewed salient features as well. Moreover, we expect this unobserved information to be similar to the information we have on the clustering on the observed features.

In addition to information-based clustering, we present similar generalization theorems for distance-based clustering, and use these to analyze generalization properties of k-means. Under some assumptions, k-means is also known as a solution for the maximum likelihood Gaussian mixture model. Analogous to what we show for information based clustering and multinomial mixture models, we show that this optimization goal of k-means is also optimal in terms of generalization to unobserved features.

The key assumption that enables us to prove these theorems is the *random independent selection* of the observed features. Note that the counter claim to random selection would be that given two instances $\{\mathbf{x}[1], \mathbf{x}[2]\}$, there is a correlation between the distance of a feature $|x_q[1] - x_q[2]|$ and the probability of observing this feature; e.g., the probability of *observing* features that are similar is higher. If no such correlation exists, then the selection can be considered random in our context. Hence, we believe that in practice the random selection assumption is reasonable. However, in many cases, the assumption of complete independence in the selection of features is less natural. Therefore, we believe that further research on the effects of dependence in selection is required.

Another interpretation of the generalization theorem, without using the *random independence* assumption, might be combinatorial. The difference between the observed and unobserved information is large only for a small portion of all possible partitions into observed and unobserved features. This means that almost any arbitrary partition generalizes well.

The value of clustering which preserves information on unobserved features is that it enables us to learn new – previously unobserved – attributes from a small number of examples. Suppose that after clustering fruits based on their observed features (Example 2.3), we eat a chinaberry[9] and thus, we "observe" (by getting sick), the previously unobserved attribute of toxicity. Assuming that in each cluster, all fruits have similar unobserved attributes, we can conclude that all the fruits in the same cluster, i.e., all chinaberries are likely to be poisonous.

Clustering is often used in scientific research when collecting measurements on objects such as stars or neurons. In general, the quality of a theory in science is measured by its predictive power. Therefore, a reasonable measure of the quality of clustering, as used in scientific research, is its ability to predict unobserved features, or measurements. This is different from clustering that merely describes the observed measurements, and supports the rationale for defining the quality of clustering by its predictivity on unobserved features.

Further Research

Our clustering maximizes expected information on randomly selected features. Although on average this information may be high, there might be features the clustering has no information about. To address this problem, we could create more than one clustering, in such a way that each clustering contains information on other features. To achieve this, we want each new clustering to discover new information, i.e., not to be redundant with previously created clusterings. This can be done in the context of the Information Bottleneck [36, 17]. Another alternative is to represent each instance by a low dimensional vector, and then use this vector to predict unobserved features. Blitzer et al. [11] represented words in a model called Distributed Binary Latent Variables, and used this representation to predict

[9] Chinaberries are the fruits of the Melia azedarach tree, and are poisonous.

another word. Adopting this idea in our context, we can replace cluster labels by a vector of binary variables assigned to each instance, where each such variable encodes an independent aspect of the instance. Generalization in this case refers to the ability to predict unobserved features from these latent variables.

Our framework can also be extended beyond clustering by formulating a general question. Given the (empirical) marginal distribution of a random subset of features $P(X_{q_1}, \ldots, X_{q_n})$, what can we say about the distribution of the full set $P(X_1, \ldots, X_L)$? In this chapter we proposed a clustering based on a subset of features, and analyzed the information that the clustering yielded on features outside this subset. It would be useful to find more sophisticated representations than clustering, and analyze other theoretical aspects of the relationship between the distribution of the subset to that of the full set. This type of theoretical analysis can help in deriving prediction algorithms, where there are many instances for some of the variables (features), but other variables are rarely viewed, as in collaborative filtering. By relating the distribution of some variables to the distribution of others, we can also analyze and improve the estimation of $p(\mathbf{x})$ from a finite sample, even without assuming the existence of unobserved features. In a different context, Han [40] analyzed the relationship between the average (per variable) entropy of random subsets of variables. He showed that the average entropy of a random subset of variables monotonically decreases with the size of the subset (see also [20]). These results were developed in the context of information theory and compression, but may be applicable to learning theory as well.

In this chapter, we assumed that we do not have additional prior information on the features. In practice, we often do have such information, and this knowledge can be represented by meta-features (See Section 1.3.3). A possible extension of our framework is to use this knowledge to improve and analyze generalization as a function of the meta-features of the unobserved features. We can create several partitions of the same instances, where each partition is optimized to different type of features. The type of feature may be defined by its meta-features. Then, if we assume that we know the value of meta-features of a new unobserved feature, we can decide which of the partitions is most informative on this feature.

In this work we focused on a new feature generalization analysis. Another research direction is to combine standard instance generalization with feature generalization. In problems

like collaborative filtering or gene expression, there is an inherent symmetry between features and instances that have been used before in various ways (see e.g., [109]). In the context of supervised learning, a recent work by Globerson and Roweis [34] addresses the issue of handling differences in the set of observed features between training and test time. However, a general framework for generalization to both unobserved features and unobserved instances is still lacking.

In some clustering models, it is assumed that instances are sampled from a fixed distribution. Thus, the quality of a clustering algorithm can be measured by its stability with respect to the sampling process, e.g., the clusters do not change significantly if we add some data points to our sample. A direction for further research of our work could concentrate on analyzing the feature stability of a clustering algorithm, e.g., stability with respect to the addition of new features.

2.6 Proofs

In this section we provide proofs for the theorems that we present in Chapter 2.

2.6.1 Proof of Theorem 2.1

We start by introducing the following lemma, which is required for the proof of Theorem 2.1.

Lemma 2.2 *Consider a function g of two independent discrete random variables (U, V). We assume that $g(u, v) \leq c$, $\forall u, v$, where c is some constant. If $\Pr\{g(U,V) > \tilde{\epsilon}\} \leq \delta$, then*

$$\Pr_V \{\mathbf{E}_u (g(u, V)) \geq \epsilon\} \leq \frac{c - \tilde{\epsilon}}{\epsilon - \tilde{\epsilon}} \delta, \quad \forall \epsilon > \tilde{\epsilon}.$$

Proof of lemma 2.2: Let $\mathcal{V}_L$ be the set of values of V, such that for every $v' \in \mathcal{V}_L$, $E_u (g(y, v')) \geq \epsilon$. For every such v' we get,

$$\epsilon \leq \mathbf{E}_u (g(u, v')) \leq c \Pr\{g(U,V) > \tilde{\epsilon} | V = v'\} + \tilde{\epsilon} \Pr\{g(U,V) \leq \tilde{\epsilon} | V = v'\}.$$

Hence, $\Pr\{g(U,V) > \tilde{\epsilon} | V = v'\} \geq \frac{\epsilon - \tilde{\epsilon}}{c - \tilde{\epsilon}}$. From the complete probability formula,

$$\begin{aligned} \delta \geq \Pr\{g(U,V) > \tilde{\epsilon}\} &= \textstyle\sum_z \Pr\{g(U,V) > \tilde{\epsilon} | V = v\} P(v) \\ &\geq \frac{\epsilon - \tilde{\epsilon}}{c - \tilde{\epsilon}} \textstyle\sum_{V : V \in V_L} P(v) \\ &= \frac{\epsilon - \tilde{\epsilon}}{c - \tilde{\epsilon}} \Pr_V \{\mathbf{E}_u (g(u, V)) \geq \epsilon\}. \end{aligned}$$

Lemma 2.2 follows directly from the last inequality. □

We first provide an outline of the proof of Theorem **2.1** and then provide a detailed proof.

Theorem 2.1 – Proof outline: For the given m instances and any clustering $\mathbf{t}$, draw uniformly and independently m' instances (repeats allowed). For any feature index q, we can estimate $I(\mathbf{t}(Z); x_q[Z])$ from the empirical distribution of $(\mathbf{t}, x_q)$ over the m' instances. This empirical distribution is $p(\mathbf{t}(Z'), x_q[Z'])$ where Z' is a random variable denoting the index of instance chosen uniformly from the m' instances (defined formally below). The proof is built up from the following upper bounds, which are independent of m, but depend on the choice of m'. The first bound is on $\mathbf{E}\{|I(\mathbf{t}(Z); x_q[Z]) - I(\mathbf{t}(Z'); x_q[Z'])|\}$, where q is

fixed and the expectation is over random selection of the m' *instances*. From this bound we derive an upper bound on $|I_{ob} - \mathbf{E}(\hat{I}_{ob})|$ and $|I_{un} - \mathbf{E}(\hat{I}_{un})|$, where $\hat{I}_{ob}$, $\hat{I}_{un}$ are the estimated values of I_{ob}, I_{un} based on the subset of m' instances, i.e., the empirical distribution. The last required bound is on the probability that $\sup_{\mathbf{t}:[m]\to[k]} \left|\mathbf{E}\left(\hat{I}_{ob}\right) - \mathbf{E}\left(\hat{I}_{un}\right)\right| > \epsilon_1$, for any $\epsilon_1 > 0$. This bound is obtained from Lemmas 2.1 and 2.2. The choice of m' is independent of m. Its value should be large enough for the estimations $\hat{I}_{ob}$, $\hat{I}_{un}$ to be accurate, but not too large, so as to limit the number of possible clusterings over the m' instances.

Note that we do not assume the m instances are drawn from a distribution. The m' instances are drawn from the empirical distribution over the m instances. □

Theorem 2.1 – Detailed proof: Let $\tilde{\mathbf{l}} = (l_1, \ldots, l_{m'})$ be indices of m' instances, where each index is selected randomly, uniformly and independently from $\{1, \ldots, m\}$. Let random variable Z' denote a number chosen uniformly at random from $\{1, \ldots, m'\}$. For any feature index q, we can estimate $I\left(\mathbf{t}(Z); x_q[Z]\right)$ from $I\left(\mathbf{t}(l_{Z'}); x_q[l_{Z'}]\right)$ as follows. The maximum likelihood estimation of entropy given a discrete empirical distribution $(\hat{p}_1, ..., \hat{p}_N)$, is defined as $\hat{H}_{MLE} = -\sum_{i=1}^{N} \hat{p}_i \log \hat{p}_i$. Note that N is the alphabet size of our discrete distribution. From [80] (Proposition 1) the bias between the empirical and actual entropy $H(p)$ is bounded as follows:

$$-\log\left(1 + \frac{N-1}{m'}\right) \leq \mathbf{E}\left(\hat{H}_{MLE}(\hat{p})\right) - H(p) \leq 0,$$

where the empirical estimation $\hat{H}_{MLE}$ is based on m' instances drawn from the distribution p. The expectation is over random sampling of these m' instances. Since $I\left(\mathbf{t}\left(Z\right); x_q[Z]\right) = -H\left(\mathbf{t}(Z), x_q[Z]\right) + H\left(\mathbf{t}\left(Z\right)\right) + H\left(x_q\left(Z\right)\right)$, we can upper bound the bias between the actual and the empirical estimation of the mutual information as follows:

$$\mathbf{E}_{\tilde{\mathbf{l}}=(l_1,\ldots,l_{m'})}\left\{\left|I\left(\mathbf{t}\left(Z\right); x_q[Z]\right) - I\left(\mathbf{t}\left(l_{Z'}\right); x_q[l_{Z'}]\right)\right|\right\} \leq \log\left(1 + \frac{ks-1}{m'}\right) \leq \frac{ks}{m'}. \tag{2.7}$$

Recall that s is the upper bound on the alphabet size of x_q.

Let $\hat{I}_{ob}(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}})$ and $\hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)$ be the estimated values of $I_{ob}(\mathbf{t}, \tilde{\mathbf{q}})$, $I_{un}(\mathbf{t})$ based on $(l_1, \ldots, l_{m'})$, i.e.

$$\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) = \frac{1}{n}\sum_{i=1}^{n} I\left(t(l_{z'}); x_{q_i}[l_{z'}]\right),$$

$$\hat{I}_{un}(\mathbf{t}, \tilde{\mathbf{l}}) = \mathbf{E}_{q\sim\mathcal{D}}\left\{I\left(\mathbf{t}(l_{z'}); x_q[l_{z'}]\right)\right\}.$$

From Equation 2.7 we obtain,

$$|I_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}\right) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right)\right)|,\ |I_{un}\left(\mathbf{t}\right) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right)| \leq ks/m'. \tag{2.8}$$

and hence,

$$\begin{aligned} |I_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}\right) - I_{un}\left(\mathbf{t}\right)| &\leq \left|\mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right)\right) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right)\right| + 2ks/m' \\ &\leq \mathbf{E}_{\tilde{\mathbf{l}}}\left(\left|\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right|\right) + 2ks/m'. \end{aligned} \tag{2.9}$$

Using Lemma 2.1 we have an upper bound on the probability that

$$\sup_{\mathbf{t}:[m]\to[k]}\left|\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right| > \epsilon.$$

over the random selection of *features*, as a function of m'. However, the upper bound we need is on the probability that

$$\sup_{\mathbf{t}:[m]\to[l]}\left\{\mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right)\right) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right)\right\} > \epsilon_1.$$

Note that the expectations $\mathbf{E}_l(\hat{I}_{ob})$, $\mathbf{E}_l(\hat{I}_{un})$ are done over a random selection of the subset of m' *instances*, for a set of features that is randomly selected *once*. In order to link these two probabilities, we use Lemma 2.2.

From Lemmas 2.1 and 2.2 it is easy to show that

$$\Pr_{\tilde{\mathbf{q}}}\left\{\mathbf{E}_{\tilde{\mathbf{l}}}\left(\sup_{\mathbf{t}:[m]\to[k]}\left|\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right|\right) > \epsilon_1\right\} \leq \frac{4\log k}{\epsilon_1} e^{-n\epsilon_1^2/\left(2(\log k)^2\right) + m'\log k}. \tag{2.10}$$

Lemma 2.2 is used, where V represents the random selection of features, U represents the random selection of m' instances, $g(u, v) = \sup_{\mathbf{t}:[m]\to[k]}|\hat{I}_{ob} - \hat{I}_{un}|$, $c = \log k$, and $\tilde{\epsilon} = \epsilon_1/2$. Since

$$\mathbf{E}_{\tilde{\mathbf{l}}}\left(\sup_{\mathbf{t}:[m]\to[k]}\left|\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right|\right) \geq \sup_{\mathbf{t}:[m]\to[k]}\mathbf{E}_{\tilde{\mathbf{l}}}\left(\left|\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)\right|\right).$$

and from eq. 2.9, 2.10 we obtain

$$\Pr_{\tilde{\mathbf{q}}}\left\{\sup_{\mathbf{t}:[m]\to[k]}|I_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}\right) - I_{un}\left(\mathbf{t}\right)| > \epsilon_1 + \frac{2ks}{m'}\right\} \leq \frac{4\log k}{\epsilon_1} e^{-n\epsilon_1^2/\left(2(\log k)^2\right) + m'\log k}.$$

By selecting $\epsilon_1 = \epsilon/2$, $m' = 4ks/\epsilon$, we obtain Theorem 2.1. □

Note that the selection of m' depends on s (maximum alphabet size of the features). This reflects the fact that in order to accurately estimate $I(\mathbf{t}(Z); x_q[Z])$, we need a number of instances, m', which is much larger than the product of k and the alphabet size of x_q.

2.6.2 Information Generalization for Soft Clustering

In Section 2.1 we assumed that we are dealing with hard clustering. Here we show that the generalization theorem (Theorem 2.1) is also applicable to soft clustering. Nevertheless, we also show that soft clustering is not required, since the maximum value of I_{ob} can be achieved by hard clustering. Hence, although *IobMax,* as appears in Definition 2.3, is a hard clustering algorithm, it also achieves maximum I_{ob} (and nearly maximum I_{un}) of all possible soft clusterings.

Theorem 2.1 is applicable to soft clustering from the following arguments. In terms of the distributions $P(\mathbf{t}(Z), x_q(Z))$, assigning a soft clustering to an instance can be approximated by a second empirical distribution, $\hat{P}$, achieved by duplicating each of the instances, and then using hard clustering. Consider, for example, a case where we create a new set of instances by duplicating each of the original instances by 100 identical instances. Using hard clustering on the $\times 100$ larger set of instances, can approximate any soft clustering of the original set with quantization of $P(T|\mathbf{X})$ in steps of 1/100. Obviously, for any $\epsilon > 0$ we can create $\hat{P}$ that satisfies $max\left|P - \hat{P}\right| < \epsilon$.

Now we show that for any soft clustering of an instance, we can find a hard clustering of the same instance that has the same or a higher value of I_{ob} (without changing the cluster identity of other instances). This is enough to show that soft clustering is not required to achieve the maximum value of I_{ob}, since any soft clustering can be replaced by hard clustering instance by instance. Let $P_\lambda(T|X_{q_1}, \ldots, X_{q_n})$ define the distribution of any soft clustering. It can be written as the weighted sum of k distributions as follows

$$P_\lambda(T|X_{q_1}, \ldots, X_{q_n}) = \sum_{i=1}^{k} \lambda_i \tilde{P}_i^j(T|X_{q_1}, \ldots, X_{q_n}), \quad 0 \le \lambda_i \le 1, \sum_{i=1}^{k} \lambda_i = 1,$$

where $\tilde{P}_i^j$ is created by keeping the same soft clustering of instances $\{1, \ldots, j-1, j+1, ..., m\}$, and replacing the soft clustering of the jth instance by a hard clustering $\mathbf{t}(j) = i$. Since

$I(T;X_q)$ is a convex function of $P(T|X_q)$ for a fixed $P(X_q)$ for any q [20], we get

$$I_{P_\lambda}(T;X_q) \leq \sum_{i=1}^{k} \lambda_i I_{\tilde{P}_i^j}(T;X_q).$$

Taking the sum over all observed features $(q_1, \ldots q_n)$, we get

$$\sum_q I_{P_\lambda}(T;X_q) \leq \sum_{i=1}^{k} \lambda_i \sum_q I_{\tilde{P}_i^j}(T;X_q).$$

and hence at least one of the distributions $\tilde{P}_1^j, \ldots, \tilde{P}_k^j$ has the same or higher I_{ob} then P_λ. In other words, we can replace the soft clustering of any instance j by a hard clustering without decreasing I_{ob}.

2.6.3 Maximum Likelihood Mixture Model and IobMax

In the proof of Theorem 2.3 we claimed that maximizing the likelihood of observed variables is equivalent to maximizing $\sum_{j=1}^{n} I(T;X_j) - I(T;Y)$. In this section we show this based on the work of [27]. For the purpose of better readability in the context of their paper, we use the same notations as in their paper, and review them briefly here. Let Y be a variable that denotes the instance identity, i.e., $Y[i] = i$ where $i \in \{1, \ldots, m\}$. Let $Q(Y, \mathbf{X})$ be the empirical distribution of the features $\mathbf{X}$ in the instances, augmented by the distribution of Y. Let $P(\mathbf{X}, T)$ be the maximum likelihood mixture model of the joint distribution $Q(\mathbf{X})$, i.e., $P(\mathbf{X}, T) = P(T) \prod_j \Pr(x_j|T)$.

From Propositions 4.1, 4.3 in [27], finding local maxima of the likelihood function is equivalent to minimizing the following Lagrangian

$$\mathcal{L}_{EM} = I_Q(T;Y) - (\mathbf{E}_Q[\log P(\mathbf{X}, T)] - \mathbf{E}_Q[\log Q(T)]).$$

as a function of $Q(T|Y)$ and $P(\mathbf{X}, T)$. In the stationary point of the EM-algorithm (see Propositions 4.4 and 4.5 in [27]), $Q(x_j, T) = P(x_j, T)$. Minimizing $\mathcal{L}_{EM}$ is equivalent to

minimizing $I(T;Y) - \sum I(T;X_j)$ as shown below:

$$\begin{aligned}
\mathcal{L}_{EM} &= I_Q(T;Y) - (\mathbf{E}_Q[\log P(\mathbf{X},T)] - \mathbf{E}_Q[\log Q(T)]) \\
&= I_Q(T;Y) - \sum_{X,T} Q(\mathbf{X},T) \log \left[P(T) \prod_j P(x_j|T) \right] - H(T) \\
&= I_Q(T;Y) + H(T) - \sum_j \sum_{T,x_j} Q(x_j,T) \log \frac{P(x_j,T)}{P(T)} - H(T) \\
&= I_Q(T;Y) - \sum_j \sum_{T,x_j} Q(x_j,T) \log \frac{P(x_j,T)}{P(x_j)P(T)} + \sum_j \sum_{T,x_j} Q(x_i,T) \log P(x_j) \\
&= I_Q(T;Y) - \sum_j I(T;X_j) + \sum_j H(X_j).
\end{aligned}$$

Since $\sum_j H(X_j)$ is independent of $Q(T|Y)$, and $P(T,Y) = Q(T,Y)$ minimizing $\mathcal{L}_{EM}$ is equivalent to maximizing

$$\sum_j I(T;X_j) - I(T;Y).$$

2.6.4 Proof of Theorem 2.4

Before proving Theorem 2.4, we write generalized definitions of D_{ob}, D_{un} and prove a generalization bound for these generalized definitions (Theorem 2.6). Then we show that Theorem 2.4 is a special case of Theorem 2.6.

The quality of the clustering with respect to a single variable, X_q, is defined by a (weighted) average distance of all pairs of instances within the same cluster (large distance means lower quality). This measure is denoted by D_q which is defined by

$$D_q\{C_1,...,C_k\} = \frac{1}{m} \sum_{r=1}^{k} \frac{1}{|C_r|} \sum_{j,l \in C_r} f(x_q[j], x_q[l]). \tag{2.11}$$

Using these definitions, we define a generalized observed intra-cluster variable, denoted by $\tilde{D}_{ob}$, as the average of D_q over the observed features within the cluster, i.e.,

$$\tilde{D}_{ob}\{C_1,...,C_k\} = \frac{1}{n} \sum_{i=1}^{n} D_{q_i}\{C_1,...,C_k\}. \tag{2.12}$$

When $f(a,b) = \frac{1}{2}(a-b)^2$, we get

$$\begin{aligned} D_{ob}\{C_1,...,C_k\} &= \frac{1}{nm}\sum_{r=1}^{k}\sum_{j\in C_r}\sum_{i=1}^{n}\left(x_{q_i}[j] - \frac{1}{|C_r|}\sum_{l\in C_r} x_{q_i}[l]\right)^2 \\ &= \frac{1}{nm}\sum_{i=1}^{n}\sum_{r=1}^{k}\frac{1}{2|C_r|}\sum_{j\in C_r}\sum_{l\in C_r}\left(x_{q_i}[j] - x_{q_i}[l]\right)^2 \\ &= \frac{1}{nm}\sum_{i=1}^{n}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j\in C_r}\sum_{l\in C_r} f\left(x_{q_i}[j], x_{q_i}[l]\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} D_{q_i}\{C_1,...,C_k\} \\ &= \tilde{D}_{ob}\{C_1,...,C_k\}, \end{aligned} \tag{2.13}$$

which means that D_{ob} is a special case of $\tilde{D}_{ob}$.

Similarly we define the generalized unobserved intra-cluster variance, denoted by $\tilde{D}_{un}$, as follows:

$$\tilde{D}_{un}\{C_1,...,C_k\} = \mathbf{E}_{q\sim\mathcal{D}}\left\{D_q\{C_1,...,C_k\}\right\}. \tag{2.14}$$

Again, when $f(a,b) = \frac{1}{2}(a-b)^2$, we get

$$D_{un}\{C_1,...,C_k\} = \tilde{D}_{un}\{C_1,...,C_k\}. \tag{2.15}$$

Theorem 2.6 *With the above definitions, for every function, f satisfies eq. 2.2, 2.3 and 2.4 (see Section 2.2) and for every $\epsilon > 0$,*

$$\Pr_{\{q_1,...,q_n\}}\left\{\sup_{\alpha(\{C_1,...,C_k\})\geq\alpha_c}\left|\tilde{D}_{ob}\{C_1,...,C_k\} - \tilde{D}_{un}\{C_1,...,C_k\}\right| > \epsilon\right\} \leq \frac{2k}{\alpha_c}e^{-n\epsilon^2/2c^2+\log\frac{c}{\epsilon}}.$$

where α is defined in Definition 2.6.

Before proving Theorem 2.6, we introduce the following lemma that is required for the proof.

Lemma 2.3 *Let $\{Z_1,...,Z_S\}$ be a set of jointly S distributed random binary variables, where $z_i \in \{0,1\}$. If $\Pr(z_i = 1) \leq \delta$ for every i then for any $N_b \geq 1$*

$$\Pr\left\{\sum_{i=1}^{S} z_i \geq N_b\right\} \leq \frac{S}{N_b}\delta.$$

Lemma 2.3 follows directly from Markov's inequality.

The proof of Theorem 2.6 (given below) is based on the observation that $\tilde{D}_{ob}$ and $\tilde{D}_{un}$ are the weighted average of distance functions over a subset of the pairs of instances. This subset includes only pairs that are within the same cluster. In other words, the calculated inter-cluster variances of a clustering is based on the weighted average of $\frac{1}{2}\sum_{r=1}^{k}|C_r|\left(|C_r|-1\right)$ pairs out of the $\frac{1}{2}m(m-1)$ pairs of instances. We define "bad pairs" as pairs of instances with a large difference between the observed and unobserved distances. We use Hoeffding's inequality and Lemma 2.3 to bound the probability that we have a large number of "bad pairs". Then we show that if the number of "bad pairs" is small, all clusterings are "good", i.e., $\left|\tilde{D}_{ob}-\tilde{D}_{un}\right|\leq\epsilon$.

Proof: of Theorem 2.6

For each pair of instances j,l $(l>j)$ we define a random variable d_{jl} as the difference between the average observed distance and the expected distance,

$$d_{jl}=\left|\frac{1}{n}\sum_{i=1}^{n}f\left(x_{q_i}[j],x_{q_i}[l]\right)-\mathbf{E}_q\left\{f\left(x_q[j],x_q[l]\right)\right\}\right|.$$

We also define a binary random variable Z_{jl} $(l>j)$ by

$$z_{jl}=\begin{cases}1 & \text{if } d_{jl}>\tilde{\epsilon}\\ 0 & \text{otherwise}\end{cases}$$

where $\tilde{\epsilon}$ is a positive constant. In other words, z_{jl} is one for "bad" pairs, i.e., pairs that have a large difference between the average observed distance and the expected distance. From Hoeffding's inequality,

$$\Pr_{\{q_1,\ldots,q_n\}}\left(z_{jl}=1\right)\leq\tilde{\delta}, \tag{2.16}$$

where

$$\tilde{\delta}=2e^{-2n\tilde{\epsilon}^2/c^2}. \tag{2.17}$$

We have $\frac{1}{2}m(m-1)$ of these random binary variables. Let N_{bad} be the number of "bad" pairs, i.e., $N_{bad}=\sum_{j,l:l>j}z_{jl}$. First we calculate an upper bound on $\left|\tilde{D}_{ob}-\tilde{D}_{un}\right|$ as a function of N_{bad}, and later we prove an upper bound on the probability of large N_{bad}.

By definition of $\tilde{D}_{ob}, \tilde{D}_{un}, d_{jl}$ and the properties of f (Equations 2.3 and 2.4) we can bound the difference between $\tilde{D}_{ob}$ and $\tilde{D}_{un}$ (for any clustering) as follows

$$\left|\tilde{D}_{ob}\left\{C_1,...,C_k\right\} - \tilde{D}_{un}\left\{C_1,...,C_k\right\}\right|$$

$$\begin{aligned} &= \left|\frac{1}{mn}\sum_{i=1}^{n}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in r} f\left(x_{q_i}[j], x_{q_i}[l]\right) - \frac{1}{m}\sum_{i=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in C_r}\mathbf{E}_q\left\{f\left(x_q[j], x_q[l]\right)\right\}\right| \\ &\leq \frac{2}{m}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in C_k: l>j} d_{jl}. \end{aligned}$$

By defining ϵ_d as the following function of the clustering

$$\epsilon_d\left(\{C_1,...,C_k\}\right) = \frac{2}{m}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in C_k: l>j} d_{jl}, \tag{2.18}$$

we have

$$\left|\tilde{D}_{ob}\left\{C_1,...,C_k\right\} - \tilde{D}_{un}\left\{C_1,...,C_k\right\}\right| \leq \epsilon_d. \tag{2.19}$$

Recall that r_t is the number of instances in cluster t. Now we calculate an upper bound on ϵ_d as a function of $\tilde{\epsilon}$ and N_{bad}. The total number of pairs in the rth cluster is $\frac{1}{2}|C_r|\left(|C_r|-1\right)$. We have N_{bad} pairs with a difference above $\tilde{\epsilon}$ (but not more than c, since f is bounded). The error of each of the other pairs is upper bounded by $\tilde{\epsilon}$. Hence we get

$$\begin{aligned} \epsilon_d &\leq \frac{2}{m}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in C_r: l>j}\left(\tilde{\epsilon} + z_{jl}\left(c-\tilde{\epsilon}\right)\right) \\ &\leq \frac{2}{m}\left(\frac{1}{2}\sum_{r=1}^{k}\frac{1}{|C_r|}|C_r|\left(|C_r|-1\right)\tilde{\epsilon} + \frac{N_{bad}\left(c-\tilde{\epsilon}\right)}{\min_r|C_r|}\right) \\ &\leq \frac{2}{m}\left(\frac{1}{2}\sum_{r=1}^{k}|C_r|\,\tilde{\epsilon} + \frac{N_{bad}c}{\min_r|C_r|}\right). \end{aligned}$$

Note that $\sum_r |C_r| = m$ (the sum of the size of all clusters is m). Hence,

$$\epsilon_d \leq \tilde{\epsilon} + \frac{2N_{bad}}{m\min_r|C_r|}c. \tag{2.20}$$

Let N_b be defined as follows

$$N_b = \frac{m\min_r|C_r|\,\tilde{\epsilon}}{2c}. \tag{2.21}$$

If $N_{bad} \leq N_b$ then $\epsilon_d \leq 2\tilde{\epsilon}$ (from Equations 2.20, 2.21). Hence,

$$\Pr_{\{q_1,\ldots,q_n\}}\{\epsilon_d > 2\tilde{\epsilon}\} = \Pr_{\{q_1,\ldots,q_n\}}\{N_{bad} > N_b\}. \tag{2.22}$$

From Lemma 2.3 and eq. 2.16 for any $N_b \geq 1$

$$\Pr_{\{q_1,\ldots,q_n\}}\{N_{bad} \geq N_b\} \leq \frac{m(m-1)}{2N_b}\tilde{\delta} \leq \frac{m^2}{2N_b}\tilde{\delta}. \tag{2.23}$$

Combining Equations 2.21, 2.22, 2.23 and the definition of $\tilde{\delta}$ (eq. 2.17) we get

$$\Pr_{\{q_1,\ldots,q_n\}}\{\epsilon_d > 2\tilde{\epsilon}\} \leq \frac{mc}{\min_r |C_r|\,\tilde{\epsilon}}\tilde{\delta} = \frac{2mc}{\min_r |C_r|\,\tilde{\epsilon}}e^{-2n\tilde{\epsilon}^2/c^2} \tag{2.24}$$

By selecting $\epsilon = \tilde{\epsilon}/2$ and using eq. 2.19 and 2.24 we get

$$\Pr_{\{q_1,\ldots,q_n\}}\left\{\sup_{\alpha(\{C_1,\ldots,C_k\})\geq\alpha_c}\left|\tilde{D}_{ob}\{C_1,\ldots,C_k\} - \tilde{D}_{un}\{C_1,\ldots,C_k\}\right| > \epsilon\right\} \leq 4\frac{m}{\min_r |C_r|}e^{-n\epsilon^2/2c^2+\log\frac{c}{\epsilon}}. \tag{2.25}$$

Using the definition of α we have $m/\min_r |C_r| \leq k/\alpha_c$. Together with eq. 2.25 we get Theorem 2.6.

□

Now we are ready to prove Theorem 2.4 by showing it is a special case of Theorem 2.6.

Proof: of Theorem 2.4

Let $f(a,b) = \frac{1}{2}(a-b)^2$. In this case f satisfies eq. 2.2, 2.3 and 2.4 where $c = 2R^2$ (since $0 \leq f(x_q[j], x_q[l]) \leq 2R^2$ for all q, j, l). In addition, $\tilde{D}_{ob} = D_{ob}$ and $\tilde{D}_{un} = D_{un}$ (eq. 2.13 and 2.15). Therefore, Theorem 2.4 is a special case of Theorem 2.6.

□

2.7 Notation Table

The following table summaries the notation and definitions used in the chapter for quick reference.

Notation	Short Description
m	Number of instances
L	Number of features (both observed and unobserved)
$\{X_1, \ldots, X_L\}$	Random variables (features)
n	Number of observed features
$\tilde{\mathbf{q}} = (q_1, \ldots, q_n)$	Indices of observed features
$\mathcal{D}$	Probability distribution of selecting features
$\{\mathbf{x}[1], \ldots, \mathbf{x}[m]\}$	Instances (each instance is a vector of L elements, where n are observed)
$x_q[j]$	The qth feature of the jth instance
$x_{q_i}[j]$	The ith observed feature of the jth instance
k	Number of clusters
$\mathbf{t} : [m] \to [k]$	Function that maps instances to clusters
$\{C_1, \ldots, C_k\}$	Clusters (C_r is the set of instances in the rth cluster)
$\lvert C_r \rvert$	Size of rth cluster
T	A variable that represents the cluster label
Z	A random variable taking values uniformly from $\{1, 2, \ldots, m\}$ (Random selection of instance index)
s	Upper bound on the number of values a discrete feature can have
I_{ob}	Average observed information (Definition 2.1)
I_{un}	Expected unobserved information (Definition 2.1)
$I^*_{un,k}$	Maximum possible value of I_{un} for k clusters (Definition 2.2)
$I^*_{ob,k}$	Value of I_{ob} for clustering with maximum I_{un} (Definition 2.2)
$\tilde{I}_{un,k}$	Value of I_{un} for clustering that achieves $\tilde{I}_{ob,k}$ (Definition 2.3)
$\tilde{I}_{ob,k}$	Maximum possible value of I_{ob} for k clusters (Definition 2.3)
$f(\cdot, \cdot)$	Distance function between two feature values
c	Constant - upper bound on the distance function
$D_{ob}\{C_1, \ldots, C_k\}$	Observed intra-cluster variance (Definition 2.4)
$D_{un}\{C_1, \ldots, C_k\}$	Expected unobserved intra-cluster variance (Definition 2.5)
D^{opt}_{un}	Minimum possible intra-cluster variance (Theorem 2.5)
$\alpha(\{C_1, \ldots, C_k\})$	Ratio between smallest to average cluster size (Definition 2.6)

Chapter 3

Feature Generalization of the Nearest Neighbor Rule

In this chapter we extend the framework presented in Chapter 2 beyond clustering, and analyze the feature generalization properties of the nearest neighbor rule. We present generalization bounds on the distance between unobserved features of instances which are nearest neighbors based on the distance between the observed features. We use the same notations as in the previous chapter (see Section 2.7).

3.1 Mathematical Formulation

Using our framework, we expect that two instances that are similar in their observed features are also similar in their unobserved features. By making the assumption that the observed and unobserved features are randomly selected from a large set of features, this can be formulated directly from Hoeffding's inequality. Consider two instances $x[1]$ and $x[2]$. Let μ be the expected distance between the features of two instances, where the expectation is over a random selection of the feature, that is, $\mu = \mathbf{E}_{q\sim\mathcal{D}}\{f(x_q[1], x_q[2])\}$. For every $\epsilon > 0$,

$$\Pr_{\{q_1,\ldots,q_n\}}\left\{\left|\frac{1}{n}\sum_{i=1}^{n} f(x_{q_i}[1], x_{q_i}[2]) - \mu\right| > \epsilon\right\} \leq 2e^{-2n\epsilon^2/c^2},$$

where f is a distance function with the properties described in Section 2.2. Informally, this means that for a large number of features the observed and unobserved distances between two instances are likely to be similar. For example, two users with similar taste on all movies made up to the present time, are likely to have similar taste on future movies as well. One orange is more similar to another orange than to a banana in its observed features (color, shape etc.). Hence, it is likely that other properties, which we might not be aware of, will also be more similar between two oranges than between an orange and a banana.

This can easily be extended to analyze feature generalization properties of the nearest neighbor rule. Suppose we have m instances and an additional *target instance*, indexed by $m+1$. This instance is denoted by $\mathbf{x}[m+1]$. Our goal is to select one instance out of the m instances that is most similar to the values of *unobserved* features of the target instance. The idea is to select an instance that is most similar in its observed features to the target instance, and then estimate a generalization bound for the expected distance of this instance and the target instance in the unobserved features.

Definition 3.1 *The average observed distance of the nearest neighbor instance to the target instance is denoted by $D_{nn,ob}$ and is defined by*

$$D_{nn,ob} = \min_{j \in \{1,\ldots,m\}} \frac{1}{n} \sum_{i=1}^{n} f\left(x_{q_i}[j], x_{q_i}[m+1]\right).$$

The index of the nearest neighbor is denoted by j_{nn}, i.e.

$$j_{nn} = \arg \min_{j \in \{1,\ldots,m\}} \frac{1}{n} \sum_{i=1}^{n} f\left(x_{q_i}[j], x_{q_i}[m+1]\right).$$

Now we are interested in the *expected* distance of the nearest neighbor, where the expectation is over a random selection of a feature according to $\mathcal{D}$. Recall that we assume that this randomly selected feature is most likely to be an unobserved feature.

Definition 3.2 *The expected unobserved distance of the nearest neighbor to the target instance is denoted by $D_{nn,un}$ and is defined by*

$$D_{nn,un} = \mathbf{E}_{q \sim \mathcal{D}} \left\{ f\left(x_q[j_{nn}], x_q[m+1]\right) \right\}.$$

The next theorem states that the difference between the observed distance ($D_{nn,ob}$) and the expected unobserved distance ($D_{nn,un}$) is small when the number of observed features is large enough, with high probability.

Theorem 3.1 *With the above definitions,*

$$\Pr_{\{q_1,\ldots,q_n\}}\{|D_{nn,ob} - D_{nn,un}| > \epsilon\} < 2e^{-2n\epsilon^2/c^2+\log m}.$$

Proof: Using Hoeffding's inequality and the union bound over all possible selections of one instance out of the m instances we get

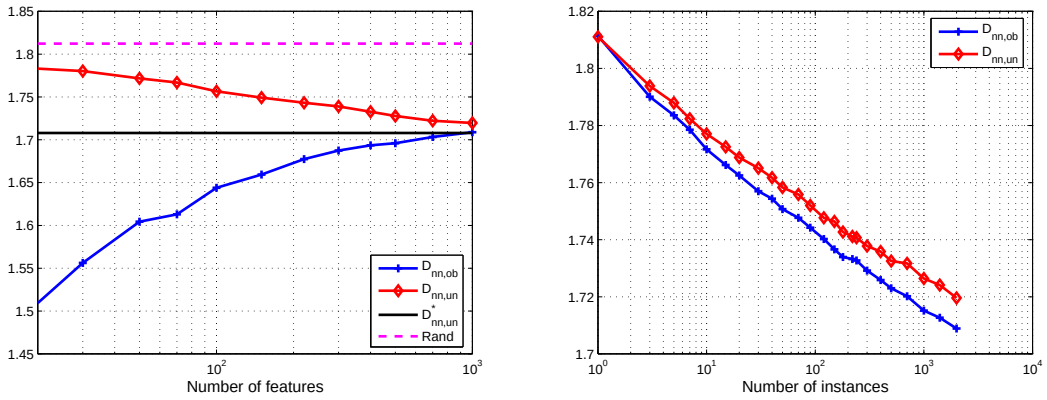
$$\Pr_{\{q_1,\ldots,q_n\}}\left\{\sup_j \left|\frac{1}{n}\sum_{i=1}^{n} f\left(x_{q_i}[j], x_{q_i}[m+1]\right) - \mathbf{E}_{q\sim\mathcal{D}}\left\{f(x_q[j], x_q[m+1]\right\}\right| > \epsilon\right\} \leq 2me^{-2n\epsilon^2/c^2}.$$

Since it holds for all j, it holds for the nearest neighbor, that is, j_{nn}.

□

Note that unlike the standard analysis of the nearest neighbor rule (see e.g., [25]), our bound on generalization error increases with the number of instances and decreases with the number of features. This occurs because we generalize to unobserved features, not instances. Consider the case where the number of instances, m, is large relative to the number of features ($\log m \gg n$). In this case, even if all features are iid (independent and identically distributed), there is a high probability of an instance that *by chance* is similar to our target instance in its observed features. Hence we cannot conclude that this instance is also similar to the target instance in the unobserved features. For a fixed number of instances, the convergence rate is bounded by $O\left(1/\sqrt{n}\right)$. For a fixed number of observed features, the bound on the difference $|D_{nn,ob} - D_{nn,un}|$ increases as $O\left(\sqrt{\log m}\right)$. This does not necessarily mean that more instances result in a larger $D_{nn,un}$, it simply means that a small difference in the observed and the unobserved variance is not guaranteed if $\log m \gg n$. However, if the number of observed features is large ($\log m \ll n$) then the similarity of the observed features of the nearest neighbor implies a similarity of the unobserved features as well. In Section 3.2 we empirically demonstrate the effects of m, n on the $D_{nn,ob}$ and $D_{nn,un}$. A simple way to avoid this dependence on the number of instances is to use the k-nearest neighbor rule, where k increases with the number of instances. However, we do not include this analysis here. Note as well that the results for clustering methods, presented in Section 2.2, are not dependent on the number of instances.

The above analysis is applicable to the standard nearest neighbor case, for the special case where the label is assumed to be a randomly selected unobserved feature. In this case,



(a) Fixed number of instances (2000)

(b) Fixed number of features (1000)

Figure 3.1: Feature generalization of the nearest neighbor rule. (a) shows the effect of the number of features for a fixed number of instances on the observed and unobserved distances (see Definitions 3.1, 3.2). $D^*_{nn,un}$ is the unobserved distance of the nearest neighbor in the *unobserved* features; that is, the best possible nearest neighbor. *Rand* refers to the average distance of features for randomly selected instances. (b) shows the effect of the number of instances on the observed and unobserved distances, where the number of features is fixed.

the value of the unobserved features is known for the labeled instances, and is unknown for the unlabeled instances. Theorem 3.1 states an upper bound (in probability) on the expected difference between an unobserved feature among the unlabeled instances and the same feature in the nearest labeled instance.

3.2 Empirical Demonstration

In this section we empirically evaluate the values of the average observed and unobserved distances of the nearest neighbor rule (Theorem 3.1) for users based on movie ratings. We test both the effects of number of features and number of instances.

Experimental setup

As in Section 2.3.1, we used the MovieLens dataset for our experiment. Our distance measure, f, is the square of the difference between the value of a feature for two users, i.e., $f(a, b) = (a - b)^2$. Our feature selection distribution ($\mathcal{D}$) is uniform. We randomly

selected one of the users to be the target instance, and out of the remaining m instances we selected the nearest neighbor in the observed distance. Then we measured the unobserved distance for this neighbor. For a reference, the unobserved distance of the nearest neighbor in the *unobserved* features was also measured and averaged over all unobserved features (denoted by $D^*_{nn,un}$). Another reference is the average distance between two randomly selected instances.The experiment was repeated for various numbers of features (n) and instances (m). The reported results were averaged over a random selection of 500 target users for each (m, n).

Results

The results are shown in Figure 3.1. When the number of features (ratings) is small, the observed distance is relatively small, since we can find users with very similar ratings for a small group of movies. However, this may occur by chance and does not necessarily reflect similar taste among these users. Indeed, the unobserved distance is relatively high for the same users; i.e., we cannot generalize from the observed to the unobserved distance. As expected from Theorem 3.1, as the number of features increases the difference between $D_{nn,ob}$ and $D_{nn,un}$ decreases. In the context of collaborative filtering, this means that when the number of observed features is large, the similarity of observed ratings reflects similarity in the users' taste.

In Figure 3.1b, we show the effect of increasing the number of users (m) for a fixed number of features ($n = 1000$). When the number of users is small, we typically do not find a user with similar observed ratings. However, the distances of the unobserved ratings and observed ratings are nearly the same. When the number of users increases, the probability of finding a user with more similar ratings increases, and hence the average observed distance decreases. However, the average unobserved distance decreases slower, and hence the difference between observed and unobserved distance increases (Theorem 3.1). When there are a large number of users, the observed similarity not only increases because we have a higher probability of finding a user with a similar taste, but also because we may find similar ratings by coincidence. This effect is significant when the number of observed features is small.

3.3 Discussion and Related Work

Using our framework, with the assumption that the label is simply an unobserved feature, we have developed a new method of analyzing the nearest neighbor rule, and obtaining finite sample generalization bounds.

A random projection of the full feature space into a subspace is commonly used for fast similarity computation in clustering or approximate nearest neighbor search (see e.g., [53, 55]). The rationale is computational: finding distances in high-dimensional space is computationally intensive. Random projection of the high dimensional space into a low dimensional space is done by multiplying the input vector of features with a random matrix. It has been shown that such random projection preserves distances up to certain level of accuracy; thus we can still find the approximate nearest neighbor or achieve reasonable clustering in the lower dimensional space. Our framework differs in two major ways. First, in our framework the random selection of observed features is part of the model that created the data, whereas in the context of random projections it is assumed that all features are observed and the subspace is randomly selected by the algorithm to reduce computational complexity. Second, in the context of random projection, data points are typically assumed to exist in some metric space, and it is irrelevant whether the axes have any meaning; in other words, only distances between points are important. In our framework, the axes are assumed to be meaningful features such as movie ratings, or represent attributes of an object. These differences lead to a different type of theoretical analysis. When analyzing a random projection, the focus is on maintaining similarities of data point distances. In our framework, we are interested in the prediction, or generalization, to unobserved features. This generalization can be in terms of mutual information (Section 2.1) or average distance per feature (Section 2.2).

Chapter 4

Learning to Select Features using Their Properties[1]

Feature selection is the task of choosing a small subset of features that is sufficient to predict the target labels well (See Section 1.1.3). In this chapter we present a novel approach to the task of feature selection. Classical methods of feature selection tell us which features are better. However, they do not tell us what *characterizes* these features or how to judge new features which were not measured in the training data. We argue that in many cases it is natural to represent each feature by a set of properties, which we call *meta-features* (See Section *1.3.3*) . As a simple example, in image related tasks where the features are gray-levels of pixels, two meta-features can be the (x, y) position of each pixel. The value of the meta-features is fixed per feature; i.e., it is not dependent on the instances. Therefore, we refer to the meta-features as prior knowledge. We use the training set to determine the relationship between the meta-feature values and feature usefulness. This in turn enables us to predict the quality of unseen features. This ability is very useful when there are a large number of potential features and it is expensive to measure the value of each feature. For this scenario we suggest a new algorithm called Meta-Feature based Predictive Feature

[1] The results presented in this chapter were first presented in a paper titled "Learning to Select Features using their Properties" submitted to JMLR at 08/2006 (a revision of the paper was submitted at 01/2008 [59]).

Selection (MF-PFS) which is an alternative to SVM-RFE. The MF-PFS algorithm uses predicted quality to select new good features, while eliminating many low-quality features without measuring them. We apply this algorithm to a visual object recognition task and show that it outperforms standard SVM-RFE. In the context of object recognition there is scientific interest in finding one set of features (referred to as *a universal dictionary*) that is sufficient for recognition of most kinds of objects. Serre et. al. [92] found that such a dictionary can be built by random selection of patches from natural images. Here we show what characterizes good universal features and demonstrate that their quality can be predicted accurately by their meta-features.

This chapter is organized as follows: we give a formal definition of the framework and define some notations in Section 4.1. In Sections 4.2 and 4.3 we show how to use this framework to predict the quality of individual unseen features and how this can be combined with RFE. In section 4.4 we apply MF-PFS on an object recognition task. Finally we conclude with some further research directions in Section 4.5. A detailed discussion on the issue of choosing meta-features wisely for both feature selection and extraction is deferred to the next chapter (Section 5.3).

4.1 Framework and Notation

In supervised learning it is assumed that we have a training set $S^m = \left\{\mathbf{x}^i, y^i\right\}_{i=1}^m$, $\mathbf{x}^i \in \mathbb{R}^N$ and $y^i = c\left(\mathbf{x}^i\right)$ where c is an unknown classification rule. The task is to find a mapping h from $\mathbb{R}^N$ to the label set with a small chance of erring on a new unseen instance, $\mathbf{x} \in \mathbb{R}^N$, that was drawn according to the same probability function as the training instances. The N coordinates are called *features*. The standard task of feature selection is to select a subset of features that enables good prediction of the label. This is done by looking for features which are more useful than others. We can also consider the instances as *abstract entities* in the space $\mathcal{S}$ and think of the features as *measurements* on the instances. Thus each feature f can be considered as a function from $\mathcal{S}$ to $\mathbb{R}$, i.e., $f : \mathcal{S} \to \mathbb{R}$. We denote the set of all the features by $\{f_j\}_{j=1}^N$. We use the term *feature* to describe both raw input variables (e.g., pixels in an image) and variables constructed from the original input variables using

Training set	Features described by meta-features
Test set	Unobserved features
Labels	Feature quality
Hypothesis class	Class of mappings from meta-features to quality
Generalization in feature selection	Predicting the quality of new features
Generalization in the joint problem	Low classification error

Table 4.1: Feature learning by meta-features as a form of standard supervised learning

some function (e.g., the product of 3 pixels in the image). We also use F to denote a set of features and S_F^m to denote the training set restricted to F; i.e., each instance is described only by the features in F.

Here we assume that each feature is described by a set of k properties $\mathbf{u}(\cdot) = \{u_r(\cdot)\}_{r=1}^k$ which we call *meta-features*. Formally, each $u_r(\cdot)$ is a function from the space of possible measurements to $\mathbb{R}$. Thus each feature f is described by a vector $\mathbf{u}(f) = (u_1(f), \ldots, u_k(f)) \in \mathbb{R}^k$. Note that the meta-features are not dependent on the instances. As already mentioned, and will be described in detail later, this enables several interesting applications. We also denote a general point in the image of $\mathbf{u}(\cdot)$ by $\mathbf{u}$. log is the base 2 logarithm while ln denotes the natural logarithm. A table that summarizes the above notations and additional notations that will be introduced later appears in the Section 4.6.

4.2 Predicting the Quality of Features

In this section we show how meta-features can be used for predicting the quality of unseen features. The ability to predict the quality of features without measuring them is advantageous for many applications. In the next section we demonstrate the its usage for efficient feature selection for SVM [111], when it is very expensive to measure each feature.

We assume that we observe only a subset of the N features; i.e., that in the training set we see only the value of some of the features. We can directly measure the quality (i.e., usefulness) of these features using the training set. Based on the quality of these features, our goal is to predict the quality of all features, including features that were not part of the training set. Thus we can think of the training set not only as a "training set of instances",

Algorithm 2 $\hat{Q}$ =quality_map(S^m, featquality, regalg)

1. measure the feature quality vector: $Y_{MF} = \text{featquality}\,(S^m)$
2. calculate the $N \times k$ meta-features matrix X_{MF}
3. use the regression alg. to learn a mapping from meta-feature value to quality: $\hat{Q} = \text{regalg}\,(X_{MF}, Y_{MF})$

but also as a "training set of features".

More formally, our goal is to use the training set S^m and the set of meta-features to learn a mapping $\hat{Q} : \mathbb{R}^k \longrightarrow \mathbb{R}$ that predicts the quality of a feature using the values of its meta-features. The quality can be based on any kind of standard evaluation function that uses the labeled training set to evaluate features (e.g., Infogain or wrapper based). Y_{MF} denotes the vector of measured qualities; i.e., $Y_{MF}(j)$ is the measured quality of the j's feature in the training set. Now we have a new supervised learning problem, with the **original features** as **instances**, the **meta-features** as **features** and Y_{MF} as the (continuous) target **label**. The analogy to the standard supervised problem is summarized in Table 4.1. Thus we can use any standard regression learning algorithm to find the required mapping from meta-features to quality. The above procedure is summarized in Algorithm 2. Note that this procedure uses a standard regression learning procedure. That is, the generalization ability to new features can be derived using standard generalization bounds for regression learning.

4.3 Efficient Feature Selection for SVM

Support Vector Machine (SVM) [111], is one of the most prominent learning algorithms in the last decade. Many feature selection algorithms for SVM have been suggested (see e.g., [116]). One of the popular methods for linear SVM is Recursive Feature Elimination [39]. In SVM-RFE one starts by training SVM using all the features, then eliminates the ones with the smallest absolute weights in the result linear classifier and repeat the same procedure with the remaining features until the set of selected features is small enough. The reason that features are eliminated iteratively and not in one step is that the weights given by SVM to a feature depend on the set of features that was used for training. Thus eliminating only

a small number of the worst features in each stage minimizes the unwanted effect of this phenomenon.

The main drawback of SVM-RFE is that we have to measure all the candidate features. This is infeasible when measuring each feature is computational expensive. Using meta-features we suggest an alternative version of SVM-RFE, that obviates the need to measure all the features. This algorithm is called Meta-Features based Predictive Feature Selection (MF-PFS). The main idea is to run SVM only on a small (random) subset of the features, then use the assigned weights for these features to predict the quality of all candidate features using their meta-features (Algorithm 2). Based on this prediction we exclude a group of low quality features, and repeat the process with smaller set of candidate features. The exact procedure is summarized in Algorithm 3. The suggested algorithm considers all the features while calculating only a small fraction of the them. Thus, it is very useful in a situation where there are a large number of candidate features and the cost of measuring each feature is very high. In the next section we demonstrate MF-PFS on such a data-set, and show that it achieves results equivalent to those obtained with standard RFE in a computation time lower in order of magnitude.

The number of measured features can be further reduced by the following methods. The relative number of features we measure in each round (α in step 2(a) of the algorithm) does not have to be fixed; e.g., we can start with a small value, since a gross estimation of the quality is enough to eliminate the very worst features, and then increase it in the last stages, where we fine-tune the selected feature set. This way we can save extra feature measurements without compromising on the performance. Typical values for α might be around 0.5 in the first iteration and slightly above 1 in the last iteration. For the same reason, in step 2(d), it makes sense to decrease the number of features we drop along the iterations. An example of specific choice of parameter values is given in the next section.

4.4 Experiments with Object Recognition

In this section we use the Caltech-101 data-set and adopt the setting of [92]. The dataset contains natural images of objects belonging to 101 different categories. The label of each

Algorithm 3 $\hat{Q}$ =MF-PFS(S^m, n, featquality, regalg)

(The algorithm selects n features out of the full set of N)

1. Initialize the set of selected features $F = \{f_j\}_{j=1}^{N}$ (all the features).
2. while $|F| > n$,
   (a) Select a set F_0 of random αn features out of F, measure them and produce a Straining set of features $S^m_{F_0}$.
   (b) Use Algorithm 2 to produce a map $\hat{Q}$ from meta-features values to quality:
   $$\hat{Q} = \text{quality\_map}(S^m_{F_0}, \text{featquality}, \text{regalg})$$
   (c) Use $\hat{Q}$ to estimate the quality of all the features in F.
   (d) Eliminate from F features with the lowest estimated quality.

image specifies which object appears in the image, but does not specify the location of the object in the image. Examples of the images are shown in Figure 4.1. [91] built a classification system for the above task using linear SVM on a sophisticated representation of the images. In their setting an image is represented by features inspired by the current model of the visual cortex. They show how the features can be build using a hierarchical feedforward system, where each layer mimics the behavior of the relevant layer in the cortex. The interested reader should consult their original paper for all the details on how the features are built. Here we simply summarize the description of the features they use, which is sufficient for our needs. First, original images are converted to a representation where the original pixels are replaced by the response to Gabor filters [32] of 4 different orientations and 16 different scales, followed by a local maximum of the absolute value over location, two adjacent scales and decimation (sub-sampling). Thus, each image is replaced by 8 quadruplets of lower resolution images. Each quadruplet corresponds to one scale, and includes all 4 orientations. The following description is over this complex representation. Each feature is defined by a specific patch *prototype*. The prototype is one such quadruplet of a given size (4x4, 8x8, 12x12 or 16x16). The value of the feature on a new image is calculated as follows:

1. The image is converted to the above complex representation.

2. The Euclidean distance of the prototype from every possible patch (i.e., at all locations and the 8 scales) of the same size in the image representation is calculated. The minimal distance (over all possible locations and scales) is denoted by d.

3. The value of the feature is $e^{-\beta d^2}$, where β is a parameter.

Step 2 is very computationally costly, and takes by far more resources than any other step in the process. This means that calculating the feature value on all images takes a very long time. Since each feature is defined by a prototype, the feature selection is done by selecting the set of prototypes. In Serre's paper the features are selected randomly by choosing random patches from a data set of natural images (or the training set itself). Namely, to select a prototype you chose an image randomly, convert it to the above representation, then randomly select a patch from it. In Serre's paper, after this random selection, the feature set is fixed, that is, no other selection algorithm is used. One method to improve the selected set of features is to use SVM-RFE for selecting the best features from a larger set of randomly selected candidate features. However, since SVM-RFE requires calculating all the feature values, this option is very expensive to compute. Therefore we suggest choosing the features using meta-features by MF-PFS. The meta-features we use are:

1. The size of the patch (i.e., 4, 8, 12 or 16).

2. The DC of the patch (i.e., average over the patch values).

3. The standard deviation of the patch.

4. The peak value of the patch.

5. Quantiles 0.3 and 0.7 of the values of the patch.

In the following we show that by using these meta-features we can predict the quality of new features with high accuracy, and hence we can drop bad features without measuring their values on the images. This significantly reduces the feature selection time. Alternatively, it can improve the classification accuracy since we are able to select from a large set of features in a reasonable training time (using feature selection by MF-PFS). In addition, we draw some interesting observations on the properties of more or less useful features.

Figure 4.1: Excerpts from the Caltech-101 data-set

4.4.1 Predicting the Quality of New Features

Since the whole discussion here relies on the assumption that we can predict the quality of a feature from its meta-features values, we first have to test whether this assumption holds. We need to prove we can indeed predict the quality of a patch using the above meta-features, i.e., from its size, DC ,std, peak value and quantile values. We measure features quality by SVM square weights of multi-class SVM [93]. Based on this quality definition, Figure 4.2 presents an example of prototypes of "good" and "bad" features of different sizes. In Figure 4.3 we explore the quality of features as a function of two meta-features: the patch size and standard deviation (std). We can see that for small patches (4x4), more details is better, for large patches (16x16) fewer details is better and for medium size patches (e.g., 8x8) an intermediate level of complexity is optimal. In other words, the larger the patch, the smaller the optimal complexity. This suggests that using the size of the patch together with certain measurements of its "complexity" (e.g., std) we should be able to predict its quality. In the following experiment we show that such a prediction is indeed possible.

We use Algorithm 2 with the sum square of SVM weights as the direct measure for patch quality and a weighted version of k-Nearest-Neighbor Regression [75] as a regressor. To solve the SVM, we use [93] online algorithm. This algorithm has the advantage of a built-in ability to deal with multi-class problems. We use 500 features as training features. Then we use the map returned by Algorithm 2 to predict the quality of another 500 features. The results presented in Figure 4.4 shows that the prediction is very accurate. The correlation coefficient between measured and predicted quality (on the test set of 500 features) is 0.94. We also

assessed the contribution of each meta-feature, by omitting one meta-feature each time and measuring the drop in the correlation coefficient. The meta-features which contributed most to the prediction are: size, mean (DC) and std.

The interesting point in the above result is that we show that feature quality can be predicted by its meta-feature values, which represent general statistical properties of the prototype. This observation is notable since it explains the existence of a universal set of features (prototypes) that enables recognition of most objects, regardless whether the prototypes were taken from pictures that contains the relevant objects or not. Indeed, [92] found that a set of features (prototypes) which consists of prototypes taken randomly from any natural images constitute such a universal set, however, they did not characterized which features are good. [108] also analyzed the properties of good features, where they use simpler representation of patches (e.g., without Gabor filter). Their conclusion is that intermediate complex features are the most useful features. However, in their work the features are object fragments, and hence their quality is dependent of the specific training set and not on general statistical properties as we found in this work.

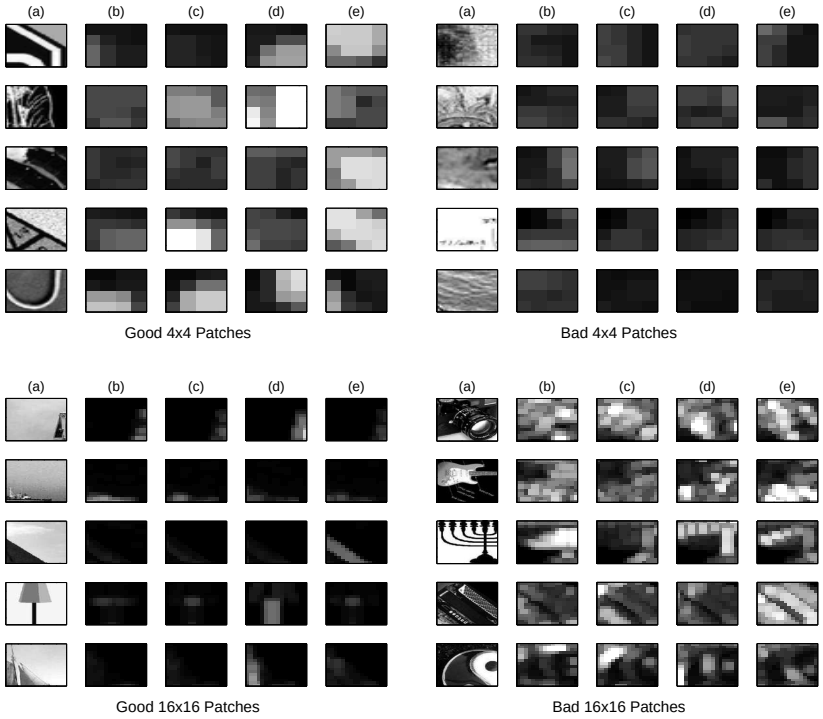


Figure 4.2: **Examples of "good" and "bad" patch prototypes of different sizes.** Each row represent one patch. The left column (a) is the image from which the patch was extracted and the other 4 columns (b,c,d,e) correspond to the 4 different orientations. Good small patches are rich in details while good large patches are relatively uniform.

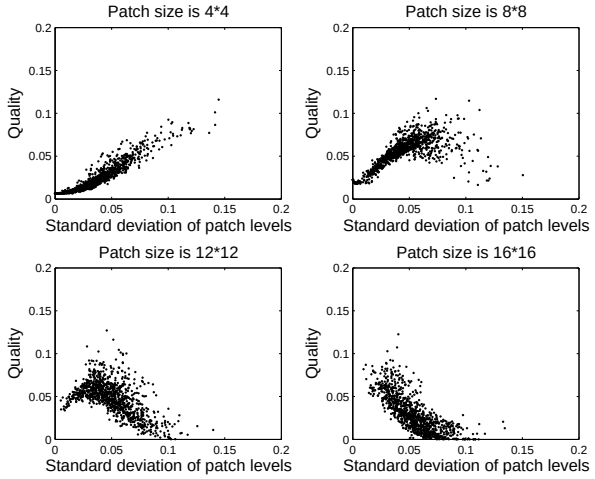


Figure 4.3: **Quality as function of the standard deviation of the meta-feature.** Good small patches are "complex" whereas good large patches are "simple".

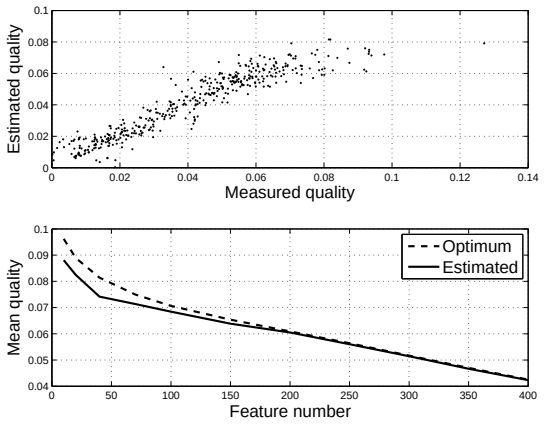


Figure 4.4: **Predicting the quality of patches.** top: Scatter plot of the predicted vs. measured quality. The correlation coefficient is 0.94. bottom: The mean quality of the top ranked features. Ranking using the predicted quality gives (almost) the same mean as ranking by the measured quality.

4.4.2 Applying MF-PFS to Object Recognition

After showing that we are able to predict patches quality, we have a reason to believe that by using MF-PFS (see Section 4.3) we can obtain an efficient feature selection in [92] setting. The features in this setting are very expensive to compute and thus a standard selection methods cannot consider many candidate features. MF-PFS on the other hand, allows us to explore a large set of features while measuring only a few of them. In this section we show that MF-PFS (with the above meta-features) indeed succeeds in selecting good features while keeping the computational cost low. Now we turn to present the experimental setup in details. Readers may also skip directly to the results.

We use Algorithm 3, with multi-class SVM [93] as a classifier and the square of weights as a measure of feature quality. The regression algorithm for quality prediction is based on a weighted version of the k-Nearest-Neighbor regression [75]. Since the number of potential features is virtually infinite, we have to select the initial set of N features in some way[2]. We always start with $N = 10n$ that were selected randomly from natural images in the same manner as in [92]. We make 4 elimination steps. We start with $\alpha = 0.4$ (see step 2(a) of

[2]In Section 5.1 we show that meta-features can be used to explore a space of an infinite number of potential features.

the algorithm) and increase it during the iterations up to $\alpha = 1.2$ in the last step. The algorithm measures only about $2.1n$ features out of the $10n$ candidates.

We compared MF-PFS to three different feature selection algorithms. The first was a standard RFE which starts with all the $N = 10n$ features and selects n features using 6 elimination steps (referred to as *RFEall*). The second method was also a standard RFE, but this time it starts with $2.1n$ features that were selected randomly from the $N = 10n$ features (referred to as *RFEsmall*). The rationale for this is to compare the performance of our algorithm to standard RFE that measures the same number of features. Since standard RFE does not predict the quality of unseen features, it has to select the initial set randomly. As a baseline we also compare to random selection of the n features as done in [92] (referred to as *Baseline*). Note that RFEall considers all the features MF-PFS considers, but makes many more measurements (with costs that become infeasible in many cases[3]). On the other hand RFEsmall, uses the same number of measurements as MF-PFS, but it considers about one fifth of the potential features. Finally, in order to estimate the statistical significance of the results we repeated the whole experiment 20 times, with different splits into train instances and test instances.

Results. The results are presented in Figure 4.5. MF-PFS is nearly as accurate as RFEall, but uses many fewer feature measurements. When RFE measures the same number of features (i.e., RFEsmall), it needs to select twice the number of selected features (n) to achieve the same classification accuracy as MF-PFS. Recall that in this setting, as [92] mentioned, measuring each feature is very expensive; thus these results represent a significant improvement.

[3] It took days on dozens of computers to measure the N=10,000 features on Caltech-101 required for *RFEall*. This is by far the most demanding computational part of the training.

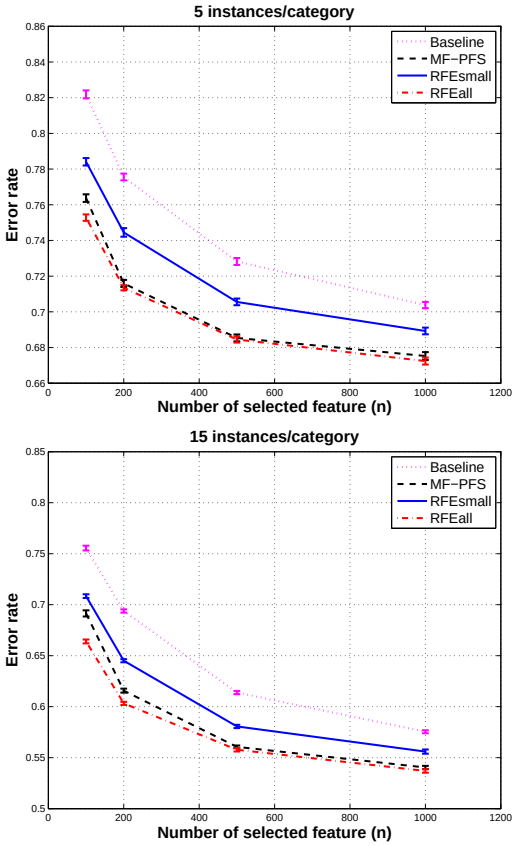


Figure 4.5: **Applying SVM with different feature selection methods for object recognition.** When the number of selected features (n) is not too small, our meta-features based selection (MF-PFS) achieves the same accuracy as RFEall which measures 5 times more features at training time. MF-PFS significantly outperforms RFEsmall which measures the same number of features at training time. To get the same classification accuracy RFEsmall needs about twice the number of features that MF-PFS needs. The results of the baseline algorithm that uses random selection are also presented (Baseline). Error bars show 1-std of the mean performance over the 20 runs.

4.5 Summary and Discussion

In this chapter we presented a novel approach to feature selection. Instead of merely selecting a set of better features out of a given set, we suggest learning the properties of good features. This approach can be used for predicting the quality of features without measuring them even on a single instance. We suggest exploring for new good features by assessing features

with meta-features values similar to those of known good features. Based on this idea, we presented an algorithm (MF-PFS) that uses feature prediction. This algorithm estimates the quality of individual features and obviates the need to calculate them on the instances. This is useful when the computational cost of measuring each feature is very high.

In the context of object recognition we showed that the feature (patch) quality can be predicted by its general statistical properties which are not dependent on the objects we are trying to recognize. This result supports the existence of a universal set of features (*universal-dictionary*) that can be used for recognition of most objects. The existence of such a dictionary is a very important issue in computer vision and brain research. We also showed that when the selection of features is based on meta-features it is possible to derive better generalization bounds on the combined problem of selection and classification.

In addition to the applications presented here which involve predicting the quality of unseen features, the meta-features framework can also be used to improve estimation of the quality of features that we do see in the training set. We suggest that instead of using direct quality estimation, we use some regression function on the meta-feature space (as in Algorithm 2). When we have only a few training instances, direct approximation of the feature quality is noisy; thus we expect that smoothing the direct measure by using a regression function of the meta-features may improve the approximation. This might be useful in the problem of tissue classification according to a gene expression array where each gene is one feature, and ontology-based properties may serve as meta-features. In most cases in this domain there are many genes and very few training instances; therefore standard feature selection methods tend to over-fit and thus yield meaningless results [26]. A meta-feature based selection can help as it may reduce the complexity of the class of possible selections.

4.6 Notation Table

In the next page we summarize the notations used in the current and next chapter.

Notation	Short description	Sections
N	total number of candidate features	
n	number of selected features	
k	number of meta-features	
m	number of training instances	
$\mathcal{S}$	(abstract) instances space	4.1, 5.2.1
f	a feature. formally, $f : \mathcal{S} \to \mathbb{R}$	
S^m	S^m is a labeled set of instances (a training set)	4.1, 4.2, 5.2.1
$u_i(f)$	the the value of the$i$'s meta-feature on feature f	4.1, 5.1
$\mathbf{u}(f)$	$\mathbf{u}(f) = (u_1(f), \ldots, u_k(f))$, a vector that describes the feature f	4.1, 5.1, 5.2.1
$\mathbf{u}$	a point (vector) in the meta-feature space	4.1, 5.1
$\hat{Q}$	a mapping from meta-features value to feature quality	4.2, 5.2.2
Y_{MF}	measured quality of the features (e.g., Infogain)	4.2
X_{MF}	$X_{MF}(i,j) =$ the value of the j's meta-feature on the i's feature	4.2
F, F_j	a set of features	5.1
V	a random variable indicating which features are selected	5.1
$p(v\|\mathbf{u})$	the conditional distribution of V given meta-features values ($\mathbf{u}$)	5.1, 5.3
h_c	a classification hypothesis	5.2.1
$\mathcal{H}_c$	the classification hypothesis class	5.2.1
d_c	the VC-dimension of $\mathcal{H}_c$	5.2.2
h_{fs}	a feature selection hypothesis - says which n features are selected	5.2.1
$\mathcal{H}_{fs}$	The feature selection hypothesis class	5.2.1
h_s	A mapping form meta-feature space to $\{0,1\}$	5.2.2
$\mathcal{H}_s$	Class of mappings from meta-feature space to $\{0,1\}$	5.2.2
d_s	The VC-dimension of $\mathcal{H}_s$	
J	number of iteration *Mufasa* (Algorithm 4) does	5.1, 5.2.1
$\|MF\|$	Number of possible different values of meta-features	5.2.1
$er_{\mathcal{D}}(h)$	generalization error of h	5.2.1
$\hat{er}_S^{\gamma}(h)$	γ-sensitive training error (instance with margin $< \gamma$ count as error)	5.2.1

Chapter 5

Meta-feature Guided Feature Extraction[1]

In the previous chapter we showed how meta-features can be used to improve feature selection by predicting feature quality in a scenario where the measurement of each feature is very costly. Here we show how the low dimensional representation of features by a relatively small number of meta-features enables efficient selection even when the number of potential features is very large or even infinite and the evaluation function is expensive (e.g., wrapper model). This is highly relevant to the feature extraction scenario. In feature extraction the original input features (e.g., pixels) are used to generate new, more complicated features (e.g., logical AND of sets of 4 binary pixels). Note that an algorithm such as MF-PFS (Section 4.3) is not adequate for this scenario, because we need to consider a huge (or even infinite) number of potentially extracted features, and thus even a fast prediction of the quality of all of them is not feasible. Thus we take another approach: a direct search in the meta-feature space, guided by an evaluation of only a subset of representative features and predicting which potential complex features have a good chance of being the most useful.

For this task we derive a selection algorithm (called *Mufasa*) that uses meta-features to explore efficiently a huge number of candidate features. We demonstrate *Mufasa* algorithm

[1] The results presented in this chapter were first presented in a paper titled "Learning to Select Features using their Properties" submitted to JMLR at 08/2006 (a revision of the paper was submitted at 01/2008).

on a handwritten digit recognition problem. We derive generalization bounds for the joint problem of feature selection (or extraction) and classification, when the selection uses meta-features. These bounds are better than the bounds obtained for direct selection.

This chapter is organized as follows. In Section 5.1 we show how the use of meta-features can be extended to sets of features, and present our algorithm for guiding feature extraction. We illustrate its abilities on the problem of handwritten digit recognition. Our theoretical results are presented in Section 5.2. We discuss the question of how to choose the meta-features wisely in Section 5.3, and summarize our results in Section 5.4.

5.1 Guided Feature Extraction

In the section we present an algorithm that searches in the meta-feature space for a good subset of features.

5.1.1 Meta-feature Based Search

Assume that we want to select (or extract) a set of n features out of large number of N potential features. We define a stochastic mapping from values of meta-features to selection (or extraction) of features. More formally, let V be a random variable that indicates which feature is selected. We assume that each point $\mathbf{u}$ in the meta-feature space induces density $p(v|\mathbf{u})$ over the features. Our goal is to find a point $\mathbf{u}$ in the meta-feature space such that drawing n features (independently) according to $p(v|\mathbf{u})$ has a high probability of giving us a good set of n features. For this purpose we suggest the *Mufasa* (for Meta-Features Aided Search Algorithm) (Algorithm 4) which uses stochastic local search in the meta-feature space.

Note that *Mufasa* does not use explicit prediction of the quality of unseen features as we did in MF-PFS, but it is clear that it cannot work unless the meta-features are informative on the quality. Namely, *Mufasa* can only work if the likelihood of drawing a good set of features from $p(v|\mathbf{u})$ is some continuous function of $\mathbf{u}$; i.e., a small change in $\mathbf{u}$ results in a small change in the chance of drawing a good set of features. If, in addition, the meta-

Algorithm 4 F_{best} =Mufasa(n, J)

1. Initialization: $q_{best} = maxreal$, $\mathbf{u}_0$ is the initial guess of $\mathbf{u}$.
2. For $j = 1 \ldots J$
   (a) Select (or generate) a new set F_j of n random features according to $p(v|\mathbf{u}_{j-1})$.
   (b) q_j = quality (F_j). (Any measure of quality, e.g., cross-validation classification accuracy)
   (c) If $q_j \geq q_{best}$
   $F_{best} = F_j$, $\mathbf{u}_{best} = \mathbf{u}_{j-1}$, $q_{best} = q_j$
   (d) Randomly select new $\mathbf{u}_j$ which is near $\mathbf{u}_{best}$. For example: $\mathbf{u}_j \leftarrow \mathbf{u}_{best} + noise$.
3. return F_{best}

features space is "simple"[2] we expect it to find a good point in a small number of steps J. In practice, we can stop when no notable improvement is achieved in a few iterations in a row. The theoretical analysis we presents later on suggests that overfitting is not a main consideration in the choice of J, since the generalization bound depends on J only logarithmically.

Like any local search over a non-convex target function, convergence to a global optimum is not guaranteed, and the result may depend on the starting point $\mathbf{u}_0$. Note that the random noise added to $\mathbf{u}$ (step 2d in Algorithm 4) may help avoid local maxima. However, other standard techniques to avoid local maxima can also be used (e.g., *simulated annealing*). In practice we found, at least in our experiments, that the results are not sensitive to the choice of $\mathbf{u}_0$, though it may affect the number of iterations, J, required to achieve good results. The choice of $p(v|\mathbf{u})$ is application dependent, but is typically quite simple if the number of meta-features is small. Another issue is how to choose the next meta-features point (step 2(d)). Standard techniques for optimizing the step size, such as gradually reducing it, can be used. However, in our experiment we simply added an independent Gaussian noise to each meta-feature.

In Section 5.1.2 we demonstrate the ability of *Mufasa* to efficiently select good features in the presence of a huge number of candidate (extracted) features on a handwritten digit recognition problem. In Section 5.2.1 we present a theoretical analysis of *Mufasa*.

[2] We elaborate on the meaning of "simple" in sections 5.2 and 5.3.

5.1.2 Illustration on a Digit Recognition Task

In this section we use a handwritten digit recognition problem to demonstrate how *Mufasa* works for feature extraction. We used the MNIST [67] dataset which contains images of 28×28 pixels of centered digits $(0 \ldots 9)$. We converted the pixels from gray-scale to binary by thresholding. We use extracted features of the following form: logical AND of 1 to 8 pixels (or their negation), which are referred to as *inputs*. This creates intermediate AND-based features which are determined by their input locations. The features we use are calculated by logical OR over a set of such AND-features that are shifted in position in a *shiftInfLen*shiftInvLen* square. For example, assume that an AND-based feature has two inputs in positions (x_1, y_1) and (x_2, y_2), *shiftInvLen=2* and both inputs are taken without negation. Thus the feature value is calculated by OR over 2*2 AND-based features as follows:

$$(Im(x_1, y_1) \wedge Im(x_2, y_2)) \vee (Im(x_1, y_1 + 1) \wedge Im(x_2, y_2 + 1)) \vee$$
$$(Im(x_1 + 1, y_1) \wedge Im(x_2 + 1, y_2)) \vee (Im(x_1 + 1, y_1 + 1) \wedge Im(x_2 + 1, y_2 + 1)),$$

where $Im(x, y)$ denotes the value of the image in location (x, y). This way we obtain features which are not sensitive to the exact position of curves in the image.

Thus a feature is defined by specifying the set of *inputs*, which of the inputs are negated, and the value of *shiftInvLen*. Similar features were already used by [63] on the MNIST dataset, but with a fixed number of inputs and without shift invariance. The idea of using shift invariance for digit recognition is also not new, and was used for example by [96]. It is clear that there are a huge number of such features; thus we have no practical way to measure or use all of them. Therefore we need some guidance for the extraction process, and this is the point where the meta-features framework comes in. We use the following four meta-features:

1. *numInputs*: the number of inputs (1-8).
2. *percentPos*: percent of logic positive pixels (0-100, rounded).
3. *shiftInvLen*: maximum allowed shift value (1-8).
4. *scatter*: average distance of the inputs from their center of gravity (COG) (1-3.5).

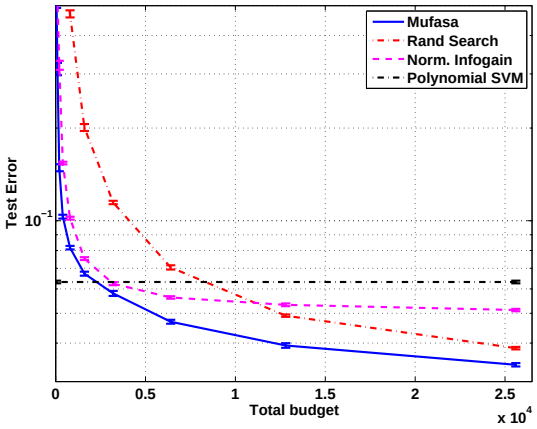


Figure 5.1: **Guided feature extraction for digit recognition**. The generalization error rate as a function of the available budget for features, using different selection methods. The number of training instances is 2000 (randomly selected from the MNIST training set). Error bars show a one standard deviation confidence interval. SVM is not limited by the budget, and always implicitly uses all the products of features. We only present the results of SVM with a polynomial kernel of degree 2, the value that gave the best results in this case.

In order to find a good value for the meta-features we use *Mufasa* (Algorithm 4), with different values of allowed budget. We use a budget (and not just the number of features) since features with large *shiftInvLen* are more computationally expensive. We defined the cost of measuring (calculating) a feature as $0.5\left(1+a^2\right)$, where a is the *shiftInvLen* of the feature; this way the cost is proportional to the number of locations where we measure the feature. We use 2000 images as a training set, and the number of steps, J, is 50. We choose specific features, given a value of meta-features, by re-drawing features randomly from a uniform distribution over the features that satisfy the given value of the meta-features until the full allowed budget is used up. We use 2-fold cross validation of the linear multi-class SVM [93, 22] to check the quality of the set of selected features in each step. Finally, for each value of allowed budget we check the results obtained by the linear SVM on the standard test set of MNIST using the selected features.

We compare the results with those obtained using the features selected by Infogain as follows. We first draw features randomly using a budget which is 50 times larger, then we sort

them by Infogain normalized by the cost[3] (i.e., value of Infogain divided by computational cost of calculating the feature as defined above). We then select the prefix that uses the allowed budget. This method is referred to as *Norm Infogain*. As a sanity check, we also compare the results to those obtained by doing 50 steps of choosing features of the allowed budget randomly, that is, over all possible values of the meta-features. Then we use the set with the lowest 2-fold cross-validation error (referred to as *Rand Search*). We also compare our results to SVM with a polynomial kernel of degree 1-4, that uses the original pixels as input features. This comparison is relevant since SVM with a polynomial kernel of degree k implicitly uses ALL the products of up to k pixels, and the product is equal to AND for binary pixels. To evaluate the statistical significance, we repeat each experiment 20 times, with a different random selection of training sets out of the standard MNIST training set. For the test set, we use the entire $10,000$ test instances of the MNIST dataset. The results are presented in Figure 5.1. It is clear that *Mufasa* outperforms the budget-dependent alternatives, and outperforms SVM for budgets larger than 3000 (that is, about 600 features). It is worth mentioning that our goal here is not to compete with the state-of-art results on MNIST, but to illustrate our concept and to compare the results of the same kind of classifier with and without using our meta-features guided search. Note that our concept can be combined with most kinds of classification, feature selection, and feature extraction algorithms to improve them, as discussed in Section 5.4.

[3] Infogain without normalization produces worse results.

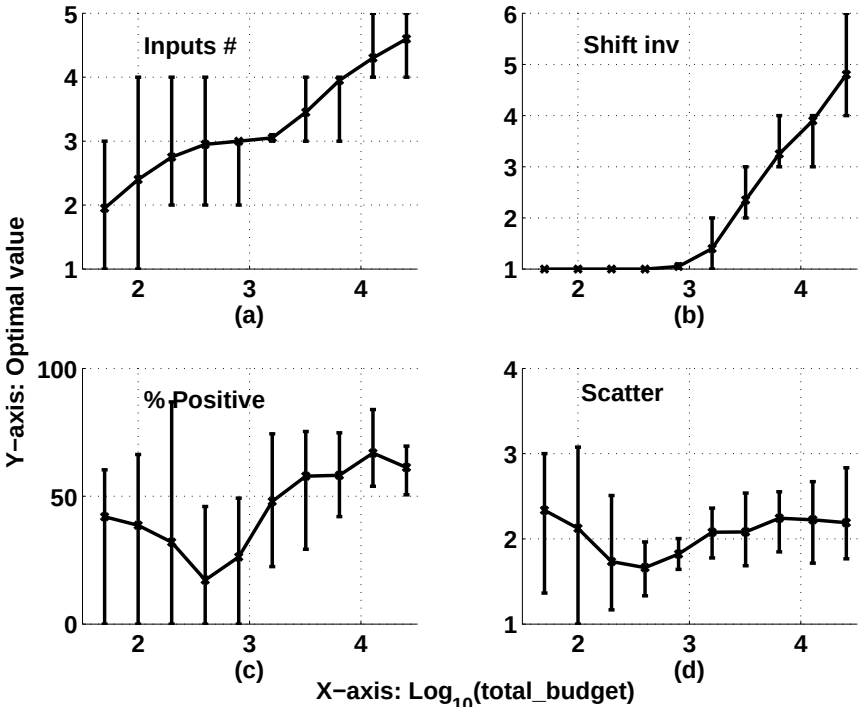


Figure 5.2: **Optimal value of the different meta-features as a function of the budget**. The results are averaged over 20 runs, and the error bars indicate the range where values fall in 80% of the runs. The size of the optimal shift invariance and the optimal number inputs increases with the budget.

Another benefit of the meta-features guided search is that it helps understand the problem. To see this we need to take a closer look at the chosen values of the meta-features ($\mathbf{u}_{best}$) as a function of the available budget. Figure 5.2 presents the average chosen value of each meta-feature as a function of the budget. As shown in Figure 5.2b, when the budget is very limited, it is better to take more cheap features rather than fewer more expensive shift invariant features. On the other hand, when we increase the budget, adding these expensive complex features is worth it. We can also see that when the budget grows, the optimal number of inputs increases. This occurs because for a small budget, we prefer features that are less specific, and have relatively high entropy, at the expense of "in class variance". For a large budget, we can permit ourselves to use sparse features (low probability of being 1), but gain specificity. For the scatter meta-features, there is apparently no correlation between the budget and the optimal value. The vertical lines (error bars) represent the range of selected

values in the different runs. It gives us a sense of the importance of each meta-feature. A smaller error bar indicates higher sensitivity of the classifier performance to the value of the meta-feature. For example, we can see that performance is sensitive to *shiftInvLen* and relatively indifferent to *percentPos*.

5.2 Theoretical Analysis

In this section we derive generalization bounds for the combined process of selection and classification, when the selection process is based on meta-features. We show that in some cases, these bounds are far better than the bounds that assume each feature can be selected directly. This is because we can significantly narrow the number of possible selections, and still find a good set of features. In Section 5.2.1 we analyze the case where the selection is made using *Mufasa* (Algorithm 4). In Section 5.2.2 we present a more general analysis, which is independent of the selection algorithm, and instead assumes that we have a given class of mappings from meta-features to a selection decision.

5.2.1 Generalization Bounds for *Mufasa* Algorithm

The bounds presented in this section assume that the selection is made using *Mufasa* (Algorithm 4), but they could be adapted to other meta-feature based selection algorithms. Before presenting the bounds, we need some additional notations. We assume that the classifier that is going to use the selected features is chosen from a hypothesis class $\mathcal{H}_c$ of real valued functions and the classification is made by taking the sign.

We also assume that we have a hypothesis class $\mathcal{H}_{fs}$, where each hypothesis is one possible way to select the n out of N features. Using the training set, our feature selection is limited to selecting one of the hypotheses that is included in $\mathcal{H}_{fs}$. As we show later, if $\mathcal{H}_{fs}$ contains all the possible ways of choosing n out of N features, then we get an unattractive generalization bound for large values of n and N. Thus we use meta-features to further restrict the cardinality (or complexity) of $\mathcal{H}_{fs}$. We have a combined learning scheme of choosing both $h_c \in \mathcal{H}_c$ and $h_{fs} \in \mathcal{H}_{fs}$. We can view this as choosing a single classifier from $\mathcal{H}_{fs} \times \mathcal{H}_c$. In the following paragraphs we analyze the equivalent size of hypothesis space

$\mathcal{H}_{fs}$ of *Mufasa* as a function of the number of steps in the algorithm.

For the theoretical analysis, we need to bound the number of feature selection hypotheses *Mufasa* considers. For this purpose, we reformulate *Mufasa* as follows. First, we replace step 2(a) with an equivalent deterministic step. To do so, we add a "pre-processing" stage that generates (randomly) J different sets of features of size n according to $p(v|\mathbf{u})$ for any possible value of the point $\mathbf{u}$ in the meta-feature space[4]. Now, in step 2(a) we simply use the relevant set, according to the current $\mathbf{u}$ and current j. Namely, in step j we use the jth set of features that was created in the pre-processing stage according to $p(v|\mathbf{u}_j)$, where $\mathbf{u}_j$ is the value of $\mathbf{u}$ in this step. Note that the result algorithm is identical to the original *Mufasa*, but this new formulation simplify the theoretical analysis. The key point here is that the pre-processing is done before we see the training set, and that now *Mufasa* can only select one of the feature sets created in the pre-processing. Therefore, the size of $\mathcal{H}_{fs}$, denoted by $|\mathcal{H}_{fs}|$, is the number of hypotheses created in pre-processing.

The following two lemmas upper bound $|\mathcal{H}_{fs}|$, which is a dominant quantity in the generalization bound. The first one handles the case where the meta-features have discrete values, and there are a relatively small number of possible values for the meta-features. This number is denoted by $|MF|$.

Lemma 5.1 *Any run of* Mufasa *can be duplicated by first generating $J|MF|$ hypotheses and then running* Mufasa *using these hypotheses only, that is, using $|\mathcal{H}_{fs}| \leq J|MF|$, where J is the number of iterations made by* Mufasa *and $|MF|$ is the number of different values the meta-features can be assigned.*

Proof:

We first generate J random feature sets for each of the $|MF|$ possible values of meta-features. The total number of sets we get is $J|MF|$. We have only J iterations in the algorithm, and we generated J feature sets for each possible value of the meta-features. This guarantees that all the hypotheses required by *Mufasa* are available. □

Note that in order to use the generalization bound of the algorithm, we cannot consider only the subset of J hypotheses that was tested by the algorithm. This is because this subset

[4] Later on we generalize to the case of infinite meta-feature space.

of hypotheses is affected by the training set (just as one cannot choose a single hypothesis using the training set, and then claim that the hypothesis space of the classifier includes only one hypothesis). However, from Lemma 5.1, the algorithm search within no more than $J|MF|$ feature selection hypotheses, that were determined *without* using the training set.

The next lemma handles the case where the cardinality of all possible values of meta-features is large relative to 2^J, or even infinite. In this case we can get a tighter bound that depends on J but not on $|MF|$.

Lemma 5.2 *Any run of* Mufasa *can be duplicated by first generating* 2^{J-1} *hypotheses and then running* Mufasa *using only these hypotheses, that is, using* $|\mathcal{H}_{fs}| \leq 2^{J-1}$*, where* J *is the number of iterations* Mufasa *performs.*

The proof of this lemma is based on PAC-MDL bounds [12]. Briefly, a codebook that maps between binary messages and hypotheses is built without using the training set. Thus, the generalization bound then depends on the length of the message needed to describe the selected hypothesis. For a fixed message length, the upper bound on the number of hypotheses is 2^l where l is the length of the message in bits.

Proof:

Mufasa needs to access to a random number generator in steps 2(a) and 2(d). To simplify the proof, we move the random number generation used within *Mufasa* to a pre-processing stage that stores a long vector of random numbers. Thus, every time *Mufasa* needs to access a random number, it will simply get the next stored random number. After this pre-processing, the feature set, which is the output of *Mufasa,* can be one of 2^{J-1} previously determined sets, since it only depends on the $J-1$ binary decisions in step 2(c) of the algorithm (in the first iteration the decision of step 2(c) is fixed, hence we only have $J-1$ decisions that depend on the training set). Hence, we can generate these 2^{J-1} hypotheses before we see the training set. □

Using the above PAC-MDL technique, we can also reformulate the last part of the proof by showing that each of the feature-set hypotheses can be uniquely described by $J-1$ binary bits, which describe the decisions in step 2(c). A better generalization bound can be obtained if we assume that in the final steps a new hypothesis will rarely be better than the

stored one, and hence the probability of replacing the hypothesis in step 2(c) is small. In this case, we can get a data-dependent bound that usually increases slower with the number of iterations (J), since the entropy of the message describing the hypothesis is likely to increase slowly for large J.

To state our theorem we also need the following standard definitions:

Definition 5.1 *Let $\mathcal{D}$ be a distribution over $\mathcal{S} \times \{\pm 1\}$ and $h : \mathcal{S} \to \{\pm 1\}$ a classification function. We denote by $er_{\mathcal{D}}(h)$ the generalization error of h w.r.t $\mathcal{D}$:*

$$er_{\mathcal{D}}(h) = Pr_{s,y\sim\mathcal{D}}[h(s) \neq y]$$

For a sample $S^m = \{(s_k, y_k)\}_{k=1}^m \in (\mathcal{S} \times \{\pm 1\})^m$ and a constant $\gamma > 0$, the γ-sensitive training error is:

$$\hat{er}_S^{\gamma}(h) = \frac{1}{m} |\{i : h(s_i) \neq y_i\} \quad or \quad (s_i \ \ has \ \ sample\text{-}margin < \gamma)|,$$

where the sample-margin measures the distance between the instance and the decision boundary induced by the classifier.

Now we are ready to present the main result of this section:

Theorem 5.1 *Let $\mathcal{H}_c$ be a class of real valued functions. Let S be a sample of size m generated i.i.d from a distribution $\mathcal{D}$ over $\mathcal{S} \times \{\pm 1\}$. If we choose a set of features using* Mufasa, *with a probability of $1 - \delta$ over the choices of S, for every $h_c \in \mathcal{H}_c$ and every $\gamma \in (0, 1]$:*

$$er_{\mathcal{D}}(h_c) \leq \hat{er}_S^{\gamma}(h_c) + \sqrt{\frac{2}{m}\left(d \ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta}\right) + g(J)\right)},$$

where

- *$d = fat_{\mathcal{H}_c}(\gamma/32)$ and $fat_{\mathcal{H}}(\cdot)$ denotes the fat-shattering dimension of class $\mathcal{H}$ [3].*

- $g(J) = \min(J \ln 2, \ln(J|MF|))$ *(where J is the number of steps* Mufasa *makes and $|MF|$ is the number of different values the meta-features can be assigned, if this value is finite, and ∞ otherwise).*

Our main tool in proving the above theorem is the following theorem:

Theorem 5.2 *(Bartlett, 1998)*

Let $\mathcal{H}$ be a class of real valued functions. Let S be a sample of size m generated i.i.d from a distribution $\mathcal{D}$ over $\mathcal{S} \times \{\pm 1\}$; then with a probability of $1-\delta$ over the choices of S, every $h \in \mathcal{H}$ and every $\gamma \in (0,1]$:

$$er_{\mathcal{D}}(h) \leq \hat{er}_S^{\gamma}(h) + \sqrt{\frac{2}{m}\left(d \ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta}\right)\right)},$$

where $d = fat_{\mathcal{H}}(\gamma/32)$

Proof: (of theorem 5.1)

Let $\left\{F_1, ..., F_{|\mathcal{H}_{fs}|}\right\}$ be all possible subsets of the selected features. From Theorem 5.2 we know that

$$er_{\mathcal{D}}(h_c, F_i) \leq \hat{er}_S^{\gamma}(h_c, F_i) + \sqrt{\frac{2}{m}\left(d \ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta_{F_i}}\right)\right)},$$

where $er_{\mathcal{D}}(h_c, F_i)$ denotes the generalization error of the selected hypothesis for the fixed set of features F_i.

By choosing $\delta_F = \delta/|\mathcal{H}_{fs}|$ and using the union bound, we get that the probability that there exist F_i $(1 \leq i \leq |\mathcal{H}_{fs}|)$ such that the equation below does not hold is less than δ

$$er_{\mathcal{D}}(h_c) \leq \hat{er}_S^{\gamma}(h_c) + \sqrt{\frac{2}{m}\left(d \ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta}\right) + \ln|\mathcal{H}_{fs}|\right)}.$$

Therefore, with a probability of $1-\delta$ the above equation holds for *any* algorithm that selects one of the feature sets out of $\left\{F_1, ..., F_{|H_{fs}|}\right\}$. Substituting the bounds for $|\mathcal{H}_{fs}|$ from Lemma 5.1 and Lemma 5.2 completes the proof. □

An interesting point in this bound is that it is independent of the total number of possible features, N (which may be infinite in the case of feature generation). Nevertheless, it can select a good set of features out of $O\left(2^J\right)$ candidate sets. These sets may be non-overlapping, so the potential number of features that are candidates is $O\left(n2^J\right)$. For comparison, [33] gives a same kind of bound but for direct feature selection. Their bound has the same form as our bound, but where $g(J)$ is replaced by a term of $O(\ln N)$, which is typically much larger than $J \ln 2$. If we substitute $N = n2^J$, then for the experiment described in Section 5.1.2 $n \ln N = Jn(\ln 2n) \cong 375000$ while $\ln(J|MF|) \cong 11$.

5.2.2 VC-dimension of Joint Feature Selection and Classification

In the previous section we presented an analysis which assumes that the selection of features is made using *Mufasa*. In this section we turn to a more general analysis, which is independent of the specific selection algorithm, and rather assumes that we have a given class $\mathcal{H}_s$ of mappings from meta-features to a selection decision. Formally, $\mathcal{H}_s$ is a class of mappings from meta-features value to $\{0,1\}$, that is, for each $h_s \in \mathcal{H}_s$, $h_s : \mathbb{R}^k \to \{0,1\}$. h_s defines which features are selected as follows:

$$f \text{ is selected} \iff h_s(\mathbf{u}(f)) = 1$$

where, as usual, $\mathbf{u}(f)$ is the value of the meta-features for feature f. Given the values of the meta-features of all the features together with h_s we get a single feature selection hypothesis. Therefore, $\mathcal{H}_s$ and the set of possible values of meta-features indirectly defines our feature selection hypothesis class, $\mathcal{H}_{fs}$. Since we are interested in selecting exactly n features (n is predefined), we use only a subset of $\mathcal{H}_s$ where we include only functions that imply the selection of n features[5]. For simplicity, in the analysis we use the VC-dim of $\mathcal{H}_s$ without this restriction, which is an upper bound of the VC-dim of the restricted class.

Our goal is to calculate an upper bound on the VC-dimension [112] of the joint problem of feature-selection and classification. To achieve this, we first derive an upper bound on $|\mathcal{H}_{fs}|$ as a function of VC-dim$(\mathcal{H}_s)$ and the number of features N.

[5] Note that one valid way to define $\mathcal{H}_s$ is by applying a threshold on a class of mappings from meta-feature values to feature quality, $\hat{Q} : \mathbb{R}^k \to \mathbb{R}$. See, e.g., example 2 at the end of this section.

Lemma 5.3 *Let $\mathcal{H}_s$ be a class of mappings from the meta-feature space ($\mathbb{R}^k$) to $\{0,1\}$, and let $\mathcal{H}_{fs}$ be the induced class of feature selection schemes; the following inequality holds:*

$$|\mathcal{H}_{fs}| \leq \left(\frac{eN}{\textit{VC-dim}\,(\mathcal{H}_s)}\right)^{\textit{VC-dim}(\mathcal{H}_s)}.$$

Proof:

The above inequality follows directly from the well known fact that a class with VC-dim d cannot produce more than $\left(\frac{em}{d}\right)^d$ different partitions of a sample of size m (see, for example, [54] pp. 57).

□

The next lemma relates the VC dimension of the classification concept class (d_c), the cardinality of the selection class ($|\mathcal{H}_{fs}|$) and the VC-dim of the joint learning problem.

Lemma 5.4 *Let $\mathcal{H}_{fs}$ be a class of the possible selection schemes for selecting n features out of N and let $\mathcal{H}_c$ be a class of classifiers over $\mathbb{R}^n$. Let $d_c = d_c(n)$ be the VC-dim of $\mathcal{H}_c$. If $d_c \geq 11$ then the VC-dim of the combined problem (that is, choosing $(h_{fs}, h_c) \in \mathcal{H}_{fs} \times \mathcal{H}_c$) is bounded by $(d_c + \log|\mathcal{H}_{fs}| + 1)\log d_c$.*

The proof of this lemma is given in Section 5.5.

Now we are ready to state the main theorem of this section.

Theorem 5.3 *Let $\mathcal{H}_s$ be a class of mappings from the meta-feature space ($\mathbb{R}^k$) to $\{0,1\}$, let $\mathcal{H}_{fs}$ be the induced class of feature selection schemes for selecting n out of N features and let $\mathcal{H}_c$ be a class of classifiers over $\mathbb{R}^n$. Let $d_c = d_c(n)$ be the VC-dim of the $\mathcal{H}_c$. If $d_c \geq 11$, then the VC-dim of the joint class $\mathcal{H}_{fs} \times \mathcal{H}_c$ is upper bounded as follows*

$$\textit{VC-dim}\,(\mathcal{H}_{fs} \times \mathcal{H}_c) \leq \left(d_c + d_s \log\frac{eN}{d_s} + 1\right)\log d_c.$$

where d_s is the VC-dim of $\mathcal{H}_s$.

The above theorem follows directly by substituting Lemma 5.3 in Lemma 5.4.

To illustrate the gain of the above theorem we calculate the bound for a few specific choices of $\mathcal{H}_s$ and $\mathcal{H}_c$:

1. First, we should note that if we do not use meta-features, but consider all the possible ways to select n out of N features the above bound is replaced by

$$\left(d_c + \log \left(\begin{array}{c} N \\ n \end{array} \right) + 1 \right) \log d_c \tag{5.1}$$

which is very large for reasonable values of N and n.

2. Assuming that both $\mathcal{H}_s$ and $\mathcal{H}_c$ are classes of linear classifiers on $\mathbb{R}^k$ and $\mathbb{R}^n$ respectively, then $d_s = k+1$ and $d_c = n+1$ and we get that the VC of the combined problem of selection and classification is upper bounded by

$$O\left((n + k \log N) \log n \right)$$

If $\mathcal{H}_c$ is a class of linear classifiers, but we allow any selection of n features the bound is (by substituting in 5.1):

$$O\left((n + n \log N) \log n \right)$$

which is much larger if $k \ll n$. Thus in the typical case where the number of meta-features is much smaller than the number of selected features (e.g., in Section 5.1.2) the bound for meta-feature based selection is much smaller.

3. Assuming that the meta-features are binary and $\mathcal{H}_s$ is the class of all possible functions from meta-feature to $\{0, 1\}$, then $d_s = 2^k$ and the bound is

$$O\left(\left(d_c + 2^k \log N \right) \log d_c \right)$$

which is still much better than the bound in equation 5.1 if $k \ll \log n$.

5.3 Choosing Good Meta-features

At a first glance, it might seem that our new setting only complicates the learning problem. One might claim that in addition to the standard hard task of finding a good representation of the instances, now we also have to find a good representation of the features by meta-features. However, we claim that while our setting might be more complicated to understand, in many cases it facilitates the hard and crucial problem of finding a good representation. It

gives us a systematic way to incorporate prior knowledge about the features in the feature selection and extraction process. It also enables us to use the experience gained so far in order to guide the search for good features. The fact that the number of meta-features is typically significantly smaller than the number of features makes it easy to understand the results of the feature selection process. It is easier to guess which properties might be indicative of feature quality than to guess which exact features are good. Nevertheless, the whole concept can work only if we use "good" meta-features, and this requires design. In sections 4.4 and 5.1.2 we demonstrated how to choose meta features for specific problems. In this section we give general guidelines on good choices of meta-features and mappings from meta-feature values to selection of features.

First, note that if there is no correlation between meta-features and quality, the meta-features are useless. In addition, if any two features with the same value of meta-features are redundant (highly correlated), we gain almost nothing from using a large set of them. In general, there is a trade-off between two desired properties:

1. Features with the same value of meta-features have similar quality.
2. Low redundancy between features with the same value of meta-features.

When the number of features we select is small, we should not be overly concerned about redundancy and rather focus on choosing meta-features that are informative on quality. On the other hand, if we want to select many features, redundancy may be dominant, and this requires our attention. Redundancy can also be tackled by using a distribution over meta-features instead of a single point.

In order to demonstrate the above trade-off we carried out one more experiment using the MNIST dataset. We used the same kind of features as in Section 5.1.2, but this time without shift-invariance and with fixed scatter. The task was to discriminate between 9 and 4 and we used 200 images as the training set and another 200 as the test set. Then we used *Mufasa* to select features, where the meta-features were either the (x, y)-location or the number of inputs. When the meta-feature was the (x, y)-location, the distribution of selecting the features, $p(v|\mathbf{u})$, was uniform in a 4×4 window around the chosen location (step 2a in *Mufasa*). Then we checked the classification error on the test set of a linear SVM

(which uses the selected features). We repeated this experiment for different numbers of features[6]. The results are presented in Figure 5.3. When we use a small number of features, it is better to use the (x, y)-location as a meta-feature whereas when using many features it is better to use the number of inputs as a meta-feature. This supports our prediction about the redundancy-homogeneity trade-off. The (x, y)-locations of features are good indicators of their quality, but features from similar positions tend to be redundant. On the other hand, constraints on the number of inputs are less predictive of feature quality but do not cause redundancy.

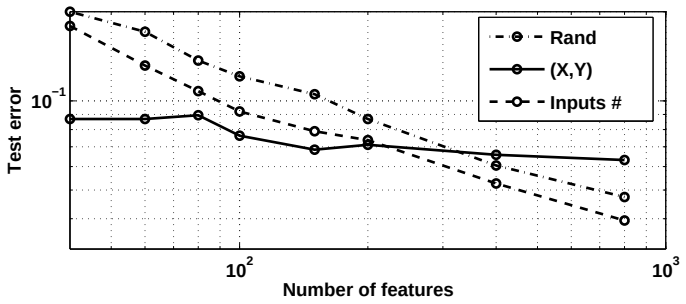


Figure 5.3: **Different choices of meta-features**. The generalization error as a function of the number of selected features. The two lines correspond to using different meta-features: (x, y)-location or number of inputs. The results of random selection of features are also presented.

5.4 Summary and Discussion

The second algorithm is *Mufasa*, which efficiently searches for a good feature set without evaluating individual features. *Mufasa* is very helpful in feature extraction, where the number of potential features is huge. It can also help avoid overfitting in the feature selection task.

In Section 5.1 we used meta-features to guide feature extraction. Our search for good features is computationally efficient and has good generalization properties because we do not examine each individual feature. However, avoiding the examination of individual features may also be considered as a disadvantage since we may include some useless individual

[6] We do not use shift invariance here; thus all the features have the same cost.

features. This can be solved by using a meta-feature guided search as a fast but rough filter for good features, and then applying more computationally demanding selection methods that examine each feature individually. This approach is similar to MF-PFS algorithm (see Section 4.3), where in the last selection step we evaluate the set of best candidate features directly.

5.5 Proof for Lemma 5.4

Proof:

For a given set of selected features, the possible number of classifications of m instances is upper bounded $\left(\frac{em}{d_c}\right)^{d_c}$ (see [54] pp. 57). Thus, for the combined learning problem, the total number of possible classifications of m instances is upper bounded by $|\mathcal{H}_{fs}|\left(\frac{em}{d_c}\right)^{d_c}$. The following chain of inequalities shows that if $m = (d_c + \log|\mathcal{H}_{fs}| + 1)\log d_c$ then $|\mathcal{H}_{fs}|\left(\frac{em}{d_c}\right)^{d_c} < 2^m$:

$$
\begin{aligned}
|\mathcal{H}_{fs}|\left(\frac{e\,(d_c + \log|\mathcal{H}_{fs}| + 1)\log d_c}{d_c}\right)^{d_c} &= |\mathcal{H}_{fs}|\,(e\log d_c)^{d_c}\left(1 + \frac{\log|\mathcal{H}_{fs}| + 1}{d_c}\right)^{d_c} \\
&\leq e\,(|\mathcal{H}_{fs}|)^{1+\log e}\,(e\log d_c)^{d_c} && (5.2)\\
&\leq (|\mathcal{H}_{fs}|)^{2+\log e}\,(e\log d_c)^{d_c} && (5.3)\\
&\leq (|\mathcal{H}_{fs}|)^{2+\log e}\,d_c^{d_c+1} && (5.4)\\
&\leq d_c^{d_c+1}\,(|\mathcal{H}_{fs}|)^{\log d_c} && (5.5)\\
&= d_c^{d_c+1} d_c^{(\log|\mathcal{H}_{fs}|)} && (5.6)\\
&= 2^{(d_c+1+\log|\mathcal{H}_{fs}|)\log d_c},
\end{aligned}
$$

where we used the following equations / inequalities:

(5.2) $(1 + a/d)^d \leq e^a \qquad \forall a, d > 0$

(5.3) here we assume $|\mathcal{H}_{fs}| > e$, otherwise the lemma is trivial

(5.4) $(e\log d)^d \leq d^{d+1} \qquad \forall d \geq 1$

(5.5) $\log d_c > 2$ (since $d_c \geq 11$)

(5.6) $a^{\log b} = b^{\log a} \qquad \forall a, b > 1$

Therefore, $(d_c + \log|\mathcal{H}_{fs}| + 1)\log d_c$ is an upper bound on VC-dim of the combined learning problem.

□

Chapter 6

Incorporating Prior Knowledge on Features into Learning[1]

In the introduction (Section 1.1.2) we discussed the importance of incorporating prior knowledge into learning, and indicated several methods in the literature that do this. Since the nature of prior knowledge is different from task to task, many of the methods designed to incorporate prior knowledge are task-specific. For example, one of the methods applied in digit recognition is to increase the training set size by creating virtual examples; in other words, new instances are created using transformations on the original instances. Although this method works well for digit recognition, it is not clear how to use it in other tasks such as text classification. Since prior knowledge may be available in very different forms, it is not clear how to develop a general approach that is applicable to *any* learning problem.

As in the previous chapters, we focus on learning problems where prior knowledge on each feature can be represented directly by a vector of meta-features. In the last two chapters we used meta-features as prior knowledge for feature selection. We assumed that meta-features are informative about feature *relevance*. In this chapter, however, the assumption is that meta-features are informative about the feature *weights* (including sign) of a linear classifier. We represent the possible mappings between meta-features and weights by a class

[1] Most of the results presented in this chapter were first presented in our AISTATS'07 paper [61].

of functions. In general, we expect these functions to be smooth; e.g., movies with similar meta-features (genres) should tend to have similar weights. Therefore, we can define a prior on the mappings from the meta-features to the weight that "prefers" smooth functions. Such a prior induces a prior on the weights which leads to better generalization.

As a simple example, consider a two-class problem of handwritten digit recognition, where the instances are represented by vectors of pixel gray levels. We use a linear discriminator: $h_{\mathbf{w}}(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x}\right)$. Assuming the data are linearly separable, the maximal-margin approach of linear SVM selects the weight vector with a minimal L2 norm, $\|\mathbf{w}\|^2$, under the constraint $\left(\mathbf{w}^T\mathbf{x}_i\right) y_i \geq 1$, for all instances in the training set $\{\mathbf{x}_i, y_i\}_{i=1}^m$, where $y_i \in \{\pm 1\}$ is the label. Here, however, we want to exploit additional information - the position of the pixel. This position is represented as a 2-element vector of meta-features (row, column) for each of the pixels. In general, we expect adjacent pixels to have similar weights. In other words, we expect the weights to be a smooth function of the meta-features. One way to achieve this is to use a Gaussian prior on $\mathbf{w}$, defined by a covariance matrix, $\mathbf{C}$. To encourage smoothness, the covariance between a pair of weights is taken to be a decreasing *function* of the distance between their meta-features, i.e. the corresponding pixel positions. This *covariance function* defines a Gaussian process on the meta-feature space, which encourages shift invariance. Then, instead of selecting the minimal norm $\|\mathbf{w}\|^2$, we select $\mathbf{w}$ with a minimal weighted norm, $\mathbf{w}^T\mathbf{C}^{-1}\mathbf{w}$ (i.e., maximum likelihood) under the same constraint $\left(\mathbf{w}^T\mathbf{x}_i\right) y_i \geq 1$. Obviously, performance can be improved by using richer features and meta-features. In section 6.2 we show that using monomial features with meta-feature based feature selection and a prior on weights can improve accuracy compared to other monomial-based kernels. Moreover, although our classifier does not use the "kernel trick", it requires significantly less memory and has lower computational complexity when classifying test instances compared to other designed kernels (see e.g., [87]).

This chapter is organized as follows: we show how meta-features can be used for kernel design in Section 6.1. In Section 6.2 we demonstrate this concept on handwritten digit recognition. We show how the same framework can be used for regression in the context of collaborative filtering in Section 6.3. In Section 6.4 we analyze the theoretical advantage of using meta-features as a prior on feature weight. We conclude the chapter by a discussion

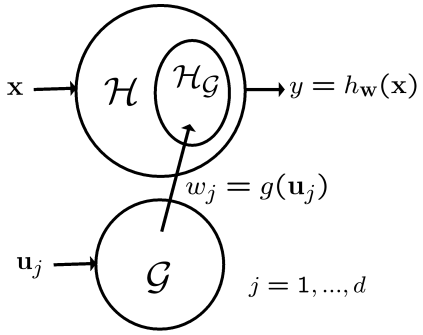


Figure 6.1: The hypothesis space of the basic setup. $\mathcal{H}$ is the hypothesis space of the standard learning problem, parametrized by $\mathbf{w}$. In our setup, we define a class of mappings, $\mathcal{G}$, from a vector of meta-features $\mathbf{u}$, to the weight of the corresponding feature. Given the values of the meta-features of all the features $\{\mathbf{u}_j\}_{j=1}^d$, the mapping $g(\mathbf{u}) \in \mathcal{G}$ indirectly defines a hypothesis $h \in \mathcal{H}$ by its weights $\{w_j\}_{j=1}^d$. Moreover, a prior on the mappings in $\mathcal{G}$, indirectly defines a prior on $\mathcal{H}$, i.e., on $\mathbf{w}$. As discussed in Section 6.4, $\mathcal{G}$ can also be used to define a subclass $\mathcal{H}_\mathcal{G} \in \mathcal{H}$, and hence reduce the hypothesis space.

in Section 6.5.

6.1 Kernel Design by Meta-features

In this section we use meta-features to define a Gaussian process prior on the meta-features to weights mapping. We now present the notations used in the current chapter, where most of them are similar to the notations we used in Chapter 4. Consider a training set $\{\mathbf{x}_i, y_i\}_{i=1}^m$, where y_i is the label and $\mathbf{x}_i \in \mathbb{R}^d$ can be the raw input variable or features extracted from the inputs. Each of the d elements in the vector $\mathbf{x}$ is a feature. Let $\{\mathbf{u}_1, \ldots, \mathbf{u}_d\}$ be vectors of meta-features, where $\mathbf{u}_j \in \mathbb{R}^k$ are the meta-features of the jth feature. A key point is that the value of the meta-features is fixed per feature, i.e., it is not dependent on the instances. Therefore, we refer to the meta-features as prior knowledge.

Let $\mathcal{H}$ be the class of homogeneous linear discrimination functions, where each function in the class $h_\mathbf{w} \in \mathcal{H}$ is defined by the weight vector $\mathbf{w}$, i.e., $h_\mathbf{w}(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x}\right)$. In order to use the meta-features, we view the weights as a function of the meta-features. Let $\mathcal{G}$ be the class of meta-features to weights mapping, then for $g \in \mathcal{G}$, each weight satisfies $w_j = g(\mathbf{u}_j)$ $(j = 1, \ldots, d)$. The prior on $\mathbf{w}$ is defined by a prior on the functions in $\mathcal{G}$, i.e., the prior on

meta-features to weight mapping. This is illustrated in Figure 6.1.

A learning algorithm uses the training set to select $\mathbf{w}$. Assuming that the training set is linearly separable, there exists an infinite set of $\mathbf{w}$ such that all instances in training set $\left(\mathbf{w}^T\mathbf{x}_i\right) y_i \geq 1$. In the standard approach of linear SVM, we solve the following optimization problem

$$\mathbf{w} = \underset{\mathbf{w}:(\mathbf{w}^T\mathbf{x}_i)y_i \geq 1,\ i=1,\ldots,m}{\operatorname{argmin}} \|\mathbf{w}\|^2 . \tag{6.1}$$

This can be viewed as finding the maximum-likelihood hypothesis, under the above constraint, where we have a Gaussian prior on $\mathbf{w}$

$$P(\mathbf{w}) \propto e^{-\frac{1}{2}\mathbf{w}^T\mathbf{C}^{-1}\mathbf{w}} \tag{6.2}$$

and the covariance matrix $\mathbf{C}$ equals the unit matrix, that is, all weights are assumed to be independent and have the same variance. However, we can exploit the meta-features in order to create a better prior on $\mathbf{w}$. For this purpose we define the covariance between weight i and j as a function of the meta-features associated with features i and j, i.e.

$$\mathbf{C}_{ij} = C(\mathbf{u}_i, \mathbf{u}_j), \tag{6.3}$$

where C is a covariance *function*. In general, we expect weights of features with similar meta-features to be similar. Therefore, the covariance function should be some decreasing function of the distance between $\mathbf{u}_i$ and $\mathbf{u}_j$. Note that the prior defined by equations 6.2 and 6.3 is a *Gaussian process* prior on the function class $\mathcal{G}$, i.e., the weights can be viewed as a Gaussian process in the meta-feature space. Therefore, we can use known results from the literature of Gaussian processes (see e.g., [117]) to define a valid covariance function, that is, guarantee that the result covariance matrix is positive semi-definite.

Once the covariance function is defined, we need to solve the following equation

$$\mathbf{w} = \underset{\mathbf{w}:(\mathbf{w}^T\mathbf{x}_i)y_i \geq 1,\ i=1,\ldots,m}{\operatorname{argmin}} \mathbf{w}^T\mathbf{C}^{-1}\mathbf{w}, \tag{6.4}$$

this can be done by mapping the original input by defining $\tilde{\mathbf{x}_i} = \mathbf{C}^{1/2}\mathbf{x}_i$ $(i = 1, ..., m)$, and solving the standard linear SVM optimization for $\tilde{\mathbf{w}}$

$$\tilde{\mathbf{w}} = \underset{\tilde{\mathbf{w}}:(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}_i})y_i \geq 1,\ i=1,\ldots,m}{\operatorname{argmin}} \|\tilde{\mathbf{w}}\|^2 , \tag{6.5}$$

where $\mathbf{w} = \mathbf{C}^{1/2}\tilde{\mathbf{w}}$. Equations 6.5 and 6.4 are equivalent since $\mathbf{w}^T\mathbf{x}_i = \tilde{\mathbf{w}}^T\mathbf{C}^{1/2}\mathbf{C}^{-1/2}\tilde{\mathbf{x}}_i = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i$ and $\mathbf{w}^T\mathbf{C}^{-1}\mathbf{w} = \|\tilde{\mathbf{w}}\|^2$. This means that our solution is also equivalent to using kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{C}\mathbf{x}_j$. Since this kernel is based on linear transformation we can save memory and computations at classification time. By transforming the result weights using $\mathbf{w} = \mathbf{C}^{1/2}\tilde{\mathbf{w}}$, we can directly calculate $\mathbf{w}^T\mathbf{x}$ for each new instance in the test set without having to keep the support vector.

We showed that the prior on the weights in eq. 6.2 can be achieved by multiplying $\mathbf{x}$ by $\mathbf{C}^{1/2}$. Using other considerations, Scholkopf et al. [87] proposed multiplying the input by a matrix $\mathbf{B}$, where $\mathbf{B}$ is optimized to achieve invariance to local transformations. In our setup, the invariance is incorporated by the assumption of smoothness of weights in the meta-feature space. The advantage of this approach is that it can be applied to other applications, e.g., collaborative filtering, where "local transformations" are not defined, but meta-features that describe features can easily be defined (see Table 1.2).

The link between Gaussian process and SVM has been pointed out by a number of authors (see e.g., [100]). In previous works, the Gaussian process is on the mapping $\mathbf{x} \rightarrow y$ and therefore reflects the smoothness of y (or $P(y|\mathbf{x})$) in the *feature* space. The key difference is that in our work, the Gaussian process is on the mapping $\mathbf{u} \rightarrow w$, that is, the meta-features to weights mapping. The advantage of this approach is that we incorporate the prior knowledge available in the meta-features.

6.1.1 A Toy Example

We first apply this method to a toy problem of a binary classifier for handwritten digit recognition that learns from two instances (one for each label). For this purpose we use the MNIST dataset [67] once more. Since the training set is extremely small, direct usage of the gray level of pixels as features is good enough, that is, we do not need to use feature extraction. The meta-features, $\mathbf{u}$, are the (row, column) position of each feature. The covariance function we use is

$$C(\mathbf{u}_i, \mathbf{u}_j) = e^{-\frac{1}{2\sigma^2}\|\mathbf{u}_i - \mathbf{u}_j\|^2}, \tag{6.6}$$

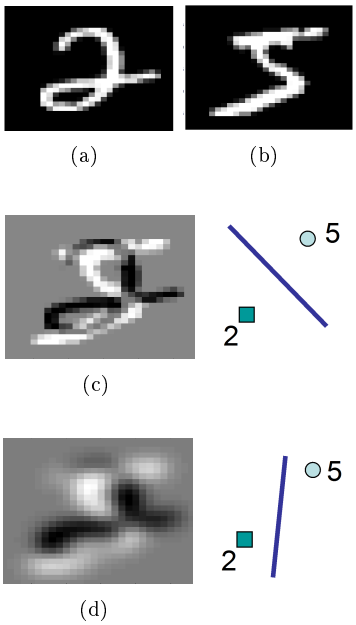


Figure 6.2: A toy example. Result weights of meta-features based prior vs. max-margin when training from two instances. (a,b) Gray level of the two digits used for training. (c) The weight per pixel of max-margin classifier (white is positive, black is negative). (d) The weights when using a Gaussian process prior on meta-features to weight mapping. The accuracy improved from 82% to 88% relative to max-margin.

where $\mathbf{u}_i$, $\mathbf{u}_j$ are the meta-features of the weights related to features i and j. σ is a parameter. This prior reflects our expectation of smoothness of the weights as a function of the feature position, and indirectly encourages shift invariance. The accuracy improved from 82% of 88% relative to max-margin. Figure 6.2 shows the result weights per pixel after training with the standard linear max-margin approach, compared to using a Gaussian process prior of meta-features to weights mapping. In the case of a larger training set we need to use richer features and meta-features, as done in Section 6.2.

6.1.2 Discussion

At this point one may wonder about the advantage of defining and using the meta-features explicitly. After all, there are other methods to incorporate knowledge of shift-invariance in an image. For example, we can find invariant hyperplanes [87] or smooth the input images by some filter. The answer is two-fold. First, our framework is more general, and can be

used for more sophisticated features, meta-features and priors. This is demonstrated in section 6.2 for a more realistic problem of handwritten digit recognition. Actually, in the above toy example, filtering the input image by 2-D convolution with a Gaussian kernel can be considered a *special case* of our framework, since it can be derived from the solution of $\tilde{\mathbf{x}}_i = \mathbf{C}^{1/2}\mathbf{x}_i$, where $\mathbf{C}$ is built using eq. 6.6. The second, and more important part of the answer is that our method can be successfully addressed to other types of learning problems where concepts such as filtering or transforming the instances do not apply but meta-features can be used (see Table 1.2). One key aim is to use similar mechanism of incorporating prior knowledge for different tasks.

6.2 Handwritten Digit Recognition Aided by Meta-features

In kernel methods, we map the input vectors into a high-dimensional feature space using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. The kernel trick (see e.g., [112]) enables us to implement a linear classifier in a high or even infinite-dimensional space of features, without explicitly calculating the features. The memory size and required number of operations at test time is in order of the number of support vectors times the number of raw inputs, instead of the number of features in the high-dimensional space. This method has proven itself to be very successful on a wide range of applications. However, using the kernel trick does not enable us to "freely" select or design the features we use. In many cases, this is not a problem, since "feature design" can be replaced by kernel design. For the task of handwritten recognition, Scholkopf et al. [87] designed a kernel that improves generalization by using local correlations between pixels.

We propose a new way to apply SVM to handwritten digit recognition. First, we *explicitly* calculate a relatively small subset of the monomials of the polynomial kernel. This subset is selected by defining meta-features for the candidate monomials, and selecting a "good" subset by the value of their meta-features. Second, we apply our framework to build a prior on the weights, based on the meta-features of the monomial features. Perhaps surprisingly, in addition to achieving better accuracy relative to other specially-designed kernels, our classifier requires significantly less memory and computations per test instance relative to

other kernels which are based on the dual representation. This is because our approach achieves good performance with a relatively small number of features (monomials), and we do not have to keep the support vectors.

Outline. The proposed method of incorporating meta-features to "standard" learning can be summarized as follows:

1. Define simple features and meta-features (Fig. 6.3). Meta-features should be informative on feature relevance or relations between features.

2. Select a subset of features by their meta-features (This is the initial selection, which can be modified later).

3. Define a Gaussian process prior of meta-features to weights mapping, i.e., covariance function $C(\mathbf{u}_i, \mathbf{u}_j)$ (eq. 6.7).

4. Build the covariance matrix $\mathbf{C}$ and solve SVM by kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{C} \mathbf{x}_j$ or linear transformation $\tilde{\mathbf{x}}_i = \mathbf{C}^{1/2}\mathbf{x}_i$ (eq. 6.4, 6.5). Calculate result weights ($\tilde{\mathbf{w}}$) and assign $\mathbf{w} = \mathbf{C}^{1/2}\tilde{\mathbf{w}}$. Only $\mathbf{w}$ is required for the classifier (test time).

5. Optimize selected features by tuning value of meta-features (e.g., by using *Mufasa* algorithm - see Algorithm 4 in Chapter 5).

Experimental setup. Again, we use the MNIST [67] dataset. Each pixel was normalized to range $[-1, 1]$. We randomly selected 5000 out of 60000 instances from the MNIST training set, and checked the performance using the MNIST test set, which includes 10000 instances. We repeated the experiment with four different randomly selected training sets. The reported results are the average of these selections. As a general reference, we used multi-class SVM [22] with a polynomial kernel of degree 4, which is the optimal degree, and obtained an error rate of 4.2%. The feature space of this kernel includes $O(10^{10})$ features, where each feature is a monomial (multiplication of up to 4 pixels).

Features and meta-features. We use the same type of features as the polynomial kernel uses indirectly. However, in order to build a prior on the weights, we need to explicitly calculate the features. Obviously, we cannot use all the $O(10^{10})$ features; therefore we

Table 6.1: List of features used (for 3 inputs).

base len. (u_r)	Orientations (u_ϕ)	Pos. (u_x, u_y)
2	$0^o, 45^o, \ldots, 315^o$	20×20 area
4	$0^o, 22.5^o, \ldots, 337.5^o$	20×20 area
6	$0^o, 22.5^o, \ldots, 337.5^o$	20×20 area

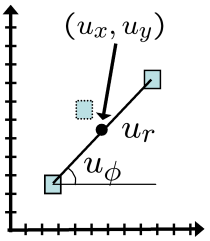


Figure 6.3: Features used in the experiments. Each feature is a monomial, multiplying the gray level of two or three pixels. The input pixels of a monomial form an isosceles triangle with its height equal to a quarter of its base (for two inputs, the middle input is omitted). Each feature is described by five meta-features $(u_t, u_r, u_\phi, u_x, u_y)$. u_t is the number of inputs (2 or 3). u_r is the distance between the two pixels at the base of the triangle. u_ϕ is the orientation of the base. (u_x, u_y) is the position of the feature in the 28×28 image.

selected a small subset of them. Since each feature is a monomial, it is defined uniquely by the position of its inputs. We used monomials with two or three inputs. The meta-features of two input monomials are the distance between the two inputs, the orientation and the center position. The three input monomials we used are defined by the same meta-features as the two input monomials, but with an additional pixel between the two pixels, forming an isosceles triangle . These features and meta-features are described in Figure 6.3.

The meta-features of the features we used are described in Table 6.1. Our initial selection was determined somewhat arbitrarily, and only the height of the triangle and maximum length ($u_r \leq 6$) were tuned using simple locate search (similar to Mufasa algorithm in Chapter 5). The total number of features with 3 inputs is $40 \times 20 \times 20 = 16000$ (40 stands for the u_r and u_ϕ, and 20×20 for the center positions). Note that for two inputs, we get the same feature for a rotation of 180^o; therefore the number of features with two inputs is only 8000. We have a total of 24000 features.

Note that here we use simpler feature than the shift invariant features we used in Chapter

5. The reason is that the shift invariance is achieved here by the prior on weights rather than by creating complex features.

The learning process. As previously, the prior on weights is determined by the covariance function C. First we use the simple covariance function

$$C(\mathbf{u}_i, \mathbf{u}_j) = \begin{cases} e^{-\frac{1}{2\sigma_P^2}\|\mathbf{u}_i - \mathbf{u}_j\|^2} & \text{if features } i,j \text{ have} \\ & \text{the same value of } u_t, u_r, u_\phi \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

where σ_P is a parameter. Note that we assume that the weights across features with different sizes, orientations or number of inputs are uncorrelated. The correlation is determined by the distance between the center position of the features. Options for more sophisticated priors on weights are discussed below. The resulting covariance matrix $\mathbf{C}$ is a block diagonal, with 60 identical blocks of size 400×400. Hence, $\mathbf{C}^{1/2}$ can be calculated efficiently[2]. Note also that $\tilde{x}$ only needs to be calculated for training instances.

The hyper-parameter σ_P can be optimized using cross-validation. However, the performance is not sensitive to the exact value of this parameter. We achieved nearly the same performance for $4 \leq \sigma_P \leq 9$. The reported results are for $\sigma_P = 6$. After calculating $\tilde{x}$, the optimization was solved using a multi-class linear SVM [22]. Since it is a linear SVM, it can also be solved online without calculating the kernel matrix [93].

Results. The results are summarized in Table 6.2. The error range is about $\pm 0.1\%$ and includes the results of all four random selections of the training set. A significant performance improvement relative to the 4th degree polynomial SVM was achieved just by using the subset of monomials (4.2% to 2.9%). The overall improvement of our method (4.2% to 1.8%) is slightly better than what was reported by Scholkopf *et al.* [87] for a combined method of virtual examples (but only with translations) and a specially designed kernel (4% to 2%)[3]. The effect of the prior on weights is presented in Figure 6.4. We believe

[2] Since the function C is Gaussian and stationary it can be shown (using Fourier analysis) that the solution of $\mathbf{C}^{1/2}\mathbf{x}$ is equivalent to 2D convolution of each of the 20×20 "maps" in feature space, with a Gaussian kernel with a standard deviation equal to $\sqrt{\sigma_P}$. Hence, $\tilde{x}$ can be calculated efficiently without explicitly calculating $\mathbf{C}^{1/2}\mathbf{x}$.

[3] They used the same size of training and test sets as we did. However, they did not report which of the

Table 6.2: Test performance on MNIST (with 5000 training instances)

Classifier	Test Err
Multiclass SVM, 4th degree polynomial	4.2%
Linear SVM on a subset of monomials	2.9%
Gaussian process on meta-feature space	1.8%
Local kernel and virtual SVM [87]	2.0%

that further research regarding the choice of features and the covariance function may lead to even better results. Since we calculate the features explicitly, we can choose any type of feature, and are not restricted to monomials. In addition, we can also incorporate methods of creating virtual examples.

For a training set of 5000 instances, our method needs about 80% less memory and computations to predict labels of new instances (after training) relative to polynomial SVM. For each test instance we need about 300,000 multiplications (calculating the features and multiplying by weights). Using the kernel trick with a 4th degree polynomial requires about 1,500,000 multiplications (there are about 2000 support vectors). The computational advantage is even greater when compared to specially designed kernels in the dual representation [87] and use of virtual examples which may increase the number of support vectors. Moreover, a preliminary experiment shows that we can significantly reduce the number of features with only a slight drop in accuracy. However, in order to apply our method to larger training sets with significantly improved performance we need to use (and design) more features. This is a disadvantage of our approach compared to most other kernel methods which scale easier with the size of training set by increasing the number of support vectors. Further research is required to determine the number of required features vs. training set size. In particular, we need to compare it to the increase in the number of support vectors. An interesting question is whether the computational advantage of our approach relative to methods that work on the dual representation increases or decreases for large training sets.

Discussion. The covariance function in eq. 6.7 exploits only part of the prior knowledge available in the meta-features. We can incorporate small orientation invariance and the effect

instances they used for training or for test.

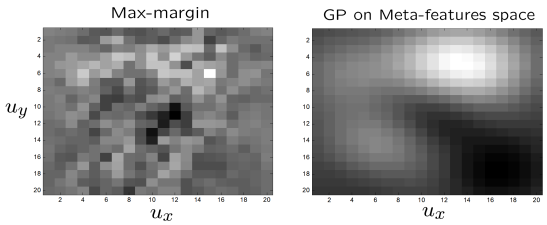


Figure 6.4: Weights of features as a function of meta-features after training with 5000 instances. The 400 weights in the above figures are for fixed (u_t, u_r, u_ϕ), and plotted as a function of the position (u_x, u_y). Left: The result weights for standard large margin classifier. It is very noisy due to the finite training set. Right: The weights when using a Gaussian process prior on meta-features space. As expected, the prior encourages the result weights to be smoother.

Table 6.3: Mean Square Error (MSE) in rating prediction. $\hat{R}$ denotes the empirical risk.

Method	Learning	MSE
Baseline I, fixed weights	$\forall j\ w_j = 1$	0.752 ± 0.002
Baseline II, direct weight learning ($\mathcal{H}$)	$\min_{\mathbf{w}} \hat{R}(h_{\mathbf{w}})$	0.753 ± 0.002
Learning with meta-features ($\mathcal{H}_{\mathcal{G}}$)	$\min_{\alpha} \hat{R}(h_{\mathbf{w}} \mid w_j = g(\mathbf{u}_j; \alpha))$	0.717 ± 0.002

of base length and number of inputs on the variance by multiplying the covariance function in eq. 6.7 by

$$A_t(u_t) A_r(u_r) e^{-\frac{(\Delta\phi_{ij})^2}{2\sigma_\phi^2}}$$

where $\Delta\phi_{ij}$ is the orientation difference between the features i and j. $A_t(u_t)$ is a function of the number of inputs, that can control the balance between 2 and 3 input monomials. Similarly, $A_r(u_r)$ can control the prior variance as a function of distance between inputs (u_r in Figure 6.3); we expect large triangles to have smaller weights. We found that the contribution of adding these terms is not significant relative to the contribution of the prior in eq. 6.7. However, for small sizes of training sets, the improvement is significant.

6.3 Collaborative Filtering using Meta-features

In this section we demonstrate our framework for another choice of function classes $\mathcal{H}$ and $\mathcal{G}$, in the context of collaborative filtering. Collaborative filtering refers to methods of making

predictions about a given user's preferences by collecting the preferences of many users. Collaborative filtering for movie ratings can generate predictions about movie rating by a user given a partial list of ratings from this user and many other users. One of the commonly used approaches in collaborative filtering is to make a prediction based on users with similar tastes in movies (see e.g., [14]).

Dataset. We used MovieLens (www.movielens.umn.edu), which is a movie rating data set. It contains approximately 1 million ratings for 3900 movies by 6040 users, i.e. the ratings are sparse. Ratings are integers on a scale of 1 to 5. We only used a subset consisting of 2000 movies by 2000 users with the maximal number of ratings.

Experimental setup. In each run of the experiment, we selected one of the movies to be the target label, and all the other movie ratings were used as features. Each instance is a vector of movie ratings by one user. Each movie (excluding the target label) is viewed as a feature, where the rating is the value of the feature. Only included users who rated the target movie. We randomly split this set of users into training and test sets, where 70% of the users was included in the training set. We repeated the experiment for each of the 200 movies with the highest number of ratings. The reported results show the mean square error (MSE) of all predictions, averaged over five random splits into training and test sets.

Features and meta features. Since our features are movies, the meta-features are properties of the movies. These properties include a set of 18 binary values which describe the movie genres (action, drama etc.). In addition we used the year the movie was made as an additional meta-feature. This meta-feature was normalized using the function $\tanh\left(\left(year - 1980\right)/15\right)$, in order to scale the year to a similar range as the other meta-features. Thus, we have a total of 19 meta-features.

Primary learning problem. For prediction, we used a kernel that weights each of the users based on their similarity. This falls into the category of memory-based methods [14] in collaborative filtering. The estimated rating for user l is

$$\hat{y}_l = \overline{y} + \gamma \frac{\sum_{i=1}^{m} ws\left(l,i\right)\left(y_i - \overline{y}\right)}{\sum_{i=1}^{m} ws\left(l,i\right)},$$

where γ is a parameter and $\overline{y}$ is the average of all ratings of the target movie in the training

set, i.e., $\overline{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$. ws is a function of the similarity between the ratings of the ith and lth users. We used the following similarity functions

$$ws(l,i) = e^{-\frac{\beta \sum_j w_j \left(x_{ij} - x_{lj}\right)^2}{\sum_j w_j}},$$

where x_{ij} is the rating of the jth movie by the ith user. The sum in the exponent is over all movies which were rated by both users l and i. The parameters β and γ are optimized as part of the training process. w_j is the weight of the jth movie. The motivation of using a different weight for each movie is the assumption that not all movies have the same relevance for our target label (movie). We hope to improve performance by giving higher weights to more relevant movies. As described below, these weights can be learned either directly or indirectly through mapping from meta-features.

Baseline I, fixed weights. As a baseline, we checked the performance using empirical risk minimization (ERM), over optimization of γ and β only. In this case $w_j = 1$ for all values of j.

Baseline II, direct weight learning. In addition to optimizing γ and β, we minimized the empirical risk by finding the optimal weights $(w_1, ..., w_{1999})$ directly. i.e., we did not use the meta-features.

Learning weights as mapping from meta-feature space. We learn a mapping g from meta-features to weights. This mapping is parametrized by the column vector $\alpha = \{\alpha_1, ..., \alpha_{19}\}$, and is given by

$$w_i = g(\mathbf{u}_i; \alpha) = e^{\alpha^T \mathbf{u}_i},$$

this means that $\mathbf{w}$ is determined indirectly from α and the meta-features. The values of α (together with γ and β) are optimized using ERM.

Results. The results are summarized in Table 6.3 (as a general reference, the MSE of k-means on the same data is 0.770). Note that direct learning of weights does not improve performance relative to using fixed weights for all movies, due to overfitting. However, when learning through meta-features, the performance improves. Using the meta-features, we did not directly learn the weight of each specific movie, but rather a function from the movie genres to weights. As described in Figure 6.1, the hypothesis space ($\mathcal{H}_\mathcal{G}$) is smaller and this prevents the overfitting.

Discussion. The usage of meta-features enabled us to utilize knowledge about our features, i.e., the movies. Note that we can also utilize demographic attributes of users, simply by adding these attributes to the vector of features. This means that our method enables us to utilize additional information both about users and about items (movies). Another interesting method that incorporate both user and item attributes was suggested by Basilico and Hofmann [4].

6.4 Towards a Theory of Meta-feature Based Prior

In this section we analyze the theoretical advantage of using meta-features as a prior on feature weight. We focus on a simple case, where the meta-features are used to reduce the hypothesis space, and some special cases of mappings from meta-features to weights. For example, if we allow only smooth mappings from meta-features to weights, only a subset of the hypotheses in $\mathcal{H}$ can be used. We denote this subset by $\mathcal{H}_{\mathcal{G}}$ (see Figure 6.1). Although the proposed theorems are simple, they illustrate the advantages of using meta-features.

We compare the VC-dimension (VCD) [112] of $\mathcal{H}_{\mathcal{G}}$ relative to the VC-dim of $\mathcal{H}$. Without using the meta-features, VCD ($\mathcal{H}$) is the number of *features*. Our results show that for a large number of features, the complexity of mappings from meta-features to weights ($\mathcal{G}$) is more important than the number of features. This is especially valuable when one can measure or generate many features (feature extraction), and hence increase the complexity of $\mathcal{H}$, while keeping the complexity of $\mathcal{G}$ low. The next theorem states that when $\mathcal{G}$ is the class of linear regression functions, VCD($\mathcal{H}_{\mathcal{G}}$) is upper bounded by the number of *meta-features*.

Theorem 6.1 *Let $\mathcal{H}$ be the class of linear discrimination functions on $\mathbb{R}^d$. We assume that the hyperplane passes through the origin, i.e., $\forall h \in \mathcal{H}$, $h_{\mathbf{w}}(\mathbf{x}) = sgn\left(\mathbf{w}^T\mathbf{x}\right)$. Let $\mathcal{G}$ be the class of homogeneous linear regression functions from meta-features to weights $\mathbb{R}^k \to \mathbb{R}$, i.e., $\forall g \in \mathcal{G}$, $g(\mathbf{u};\mathbf{c}) = \mathbf{c}^T\mathbf{u}$, where $\mathbf{c}$ is a vector of k parameters. Then VCD($\mathcal{H}_{\mathcal{G}}$) $\leq \min(k,d)$.*

Proof: Let $\mathbf{U}$ be a $k \times d$ matrix, where the jth column is $\mathbf{u}_j$, i.e., the vector of meta-features of the jth feature. By definition, $\mathbf{w} = \mathbf{U}^T\mathbf{c}$ and hence we get $h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}\left(\mathbf{c}^T\tilde{\mathbf{x}}\right)$, where $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{x}$. Since $\tilde{\mathbf{x}} \in \mathbb{R}^k$, we get an equivalent linear discrimination problem with k

parameters. Therefore VCD $(\mathcal{H}_{\mathcal{G}}) \leq k$. Since $\mathcal{H}_{\mathcal{G}} \in \mathcal{H}$ we also get VCD $(\mathcal{H}_{\mathcal{G}}) \leq$ VCD $(\mathcal{H}) = d$. □

From theorem 6.1 we can conclude that our theoretical gain from using meta-features is significant when the number of meta-features is small relative to the number of features. The next corollary generalizes Theorem 6.1 to a richer class of functions $\mathcal{G}$.

Corollary 6.1 *Using the above definition, but $\mathcal{G}$ here is the class of linear regression functions from a set of t fixed functions $\psi_1, ..., \psi_t$ of $\mathbf{u}$*

$$g(\mathbf{u}) = \sum_{j=1}^{t} c_j \psi_j(\mathbf{u}),$$

where $c_1, ..., c_t$ are the parameters that define g. Then VCD$(\mathcal{H}_{\mathcal{G}}) \leq \min(t, d)$.

The proof of corollary 6.1 is similar to that of Theorem 6.1. We simply replace each column of the matrix $\mathbf{U}$ from $\mathbf{u}_j$ to $\{\psi_1(\mathbf{u}_j), ..., \psi_t(\mathbf{u}_j)\}$, forming a $t \times d$ matrix. An example of a possible choice of a set of fixed functions is the Radial Basis Functions (RBF).

In the next Theorem, we use corollary 6.1 to relate the smoothness of the functions in $\mathcal{G}$ to VCD $(\mathcal{H}_{\mathcal{G}})$. Our measure of smoothness is the bandwidth of the functions in $\mathcal{G}$. The idea is rather simple – consider the case where the number of features is very large, but the features are crowded in the meta-feature space. If the meta-features to weights mapping is smooth, then the weights depend strongly on each other. Therefore VCD $(\mathcal{H}_{\mathcal{G}})$ is determined by the smoothness (bandwidth) of g and the volume in the meta-feature space where the features reside.

Theorem 6.2 *Let $\mathcal{H}$ be the class of linear discrimination functions on $\mathbb{R}^d$, where the hyperplane passes through the origin. Let $\mathcal{G}$ be the class of periodic functions of the meta-features, where the period in the rth dimension is T_r, i.e., $g(u_{j1}, ..., u_{jr}, ..., u_{jk}) = g(u_{j1}, ..., u_{jr} + T_r, ..., u_{jk})$ $(\forall g \in \mathcal{G}, \forall \mathbf{u}_j)$. We further assume that for all $g \in \mathcal{G}$, the bandwidth of g is limited as follows,*

$$\hat{g}(f_1, f_2, ..., f_k) = 0 \text{ if there exists r such that } |f_r| > B_r$$

for some constants $B_1, ..., B_k$, where $\hat{g}$ is the multi-dimensional continuous Fourier trans-

form of g and $f_1, ..., f_k$ are the frequencies. Then

$$VCD\left(\mathcal{H}_{\mathcal{G}}\right) \leq \prod_{r=1}^{k} \left(2B_r T_r + 1\right).$$

Proof: There are at most $t = \prod_{r=1}^{k} \lceil 2B_r T_r + 1 \rceil$ non-zero Fourier coefficients for all $g \in \mathcal{G}$. Let the set of fixed functions defined in theorem 6.1 $\{\psi_1, ..., \psi_t\}$ be the (sin / cos) functions corresponding to the Fourier basis functions that can have a non-zero coefficient. Any $g \in \mathcal{G}$ is a linear sum of these functions. Using corollary 6.1 we get theorem 6.2. □

Note that the requirement that g must be a periodic function is not a significant limit on the function class. Assuming the value of meta-features is bounded, we can choose T_r such that all our meta-feature vectors are within a half period of g. However, VCD $(\mathcal{H}_{\mathcal{G}})$ increases with T_r.

The algorithmic application of Theorem 6.2 is that we can reduce the dimensionality by representing the features in the frequency domain of the meta-feature space, and discard features that represent high frequencies. This dimensionality reduction is different from principle component analysis (PCA), since it is not based on the joint statistics of the features (empirical covariance matrix) but on a fixed prior information (the meta-features).

6.5 Discussion

We introduce a new framework to incorporate knowledge about features into learning by defining a prior on the mapping from meta-feature space to weights. We show how the prior information can be incorporated using standard classification methods such as SVM for primary learning and Gaussian processes for the meta-features to weights mapping. We further show that a smoothness assumption in meta-feature space ($\mathbf{u} \rightarrow w$) captures domain knowledge such as shift invariance better than the standard assumption of smoothness in the feature space ($\mathbf{x} \rightarrow y$). The theoretical advantage of using meta-features was shown by relating the smoothness in meta-feature space to the VC-dimension. We demonstrated the use of meta-features for handwritten digit recognition, and showed that we can achieve competitive accuracy and lower computational complexity with respect to other designed kernels. These results show that for this task a better classifier can be achieved by using

meta-features and explicitly calculating monomial features; in other words, without using the kernel trick. We also show that a prior on weight can also improve movie-rating predictions. An interesting future research direction would be to try our method on the relatively new movie rating Netflix dataset. This dataset is much larger, and has more meta-features available.

One of the methods for regularization that is widely used in artificial neural networks is weight sharing (see e.g., the work of Lecun et al. [67]). In weight sharing the weights of specific different connections between neurons in the network are forced to be equal. Many authors have shown that with a correct decision on which weights to share, generalization error drops significantly . In our context, this can be considered as a special case of a prior that maps between meta-features to weight. Although meta-features are not explicitly defined in the previous work, weights that share some similar property are forced to be equal (correlation coefficient one in our formulation). In our formulation we also allow for "soft sharing"; i.e., a correlation coefficient that is below one, but above zero.

Our framework may also be applicable to text classification where a document is represented as bag of words. There is a trade-off between representing each word as a feature vs. clustering some words as a single feature. The cluster can be based on stemming, synonyms, or even clustered by the empirical distribution of words [8]. Using our framework, we can represent each word as a different feature (without word clustering) but also "link" between the words by meta-features. Then, when building the covariance matrix we can control the correlation between the weights of each word. If we choose correlation one, then the weights are identical, and this is exactly like using a single feature; i.e., clustering the words. If the correlation is zero, we would not explore the prior information about the relation between words. It is reasonable to assume that the optimal value is between these two extremes.

One key motivation for explicitly defining and using meta-features is as follows. By representing prior knowledge on features as vectors of meta-features, we can develop common tools and theoretical results to incorporate prior knowledge into a variety of learning tasks. In this and previous chapters we introduce several such tools for feature selection, extraction and defining priors on weights.

These different uses of meta-features leads to different considerations as regards which

meta-features we should use. In Chapters 4 and 5 we assumed that meta-features are informative about feature relevance while in the current chapter, we assume that meta-features are informative about the feature *weights* (including sign) of a linear classifier. This implies that the selection of meta-features might be different in these two cases, and the requirement for feature selection presented in Section 5.3 that two features with similar meta-features should not be redundant, does not imply the use of a prior on weights. An interesting future research direction would be to combine these concepts of using meta-features into a single framework.

Chapter 7

Meta-Features and Regularization in the Cortex

In this chapter we propose a possible qualitative relation between meta-features and learning mechanisms in biological neural networks. We start with a short review of relevant learning mechanisms in the cortex.

In Section 1.4 we reviewed the Hebb learning rule and show how Hebb-like learning rules can implement several unsupervised learning models such as PCA and ICA. Recall that the basic Hebb rule refers to the change in synaptic weights of two active neurons. Hebb-like learning rules typically take the following form. The weight update of synapse i is a function of the activity of output neuron (y) and input x_i:

$$\Delta w_i = \delta\left(x_i, y\right) \tag{7.1}$$

However, Engert and Bonhoeffer [28] found evidence that weights (synaptic strength) to input synapses in close spatial proximity (up to $70\mu m$) tend to be potentiated together, even if some of the input neurons are not active at all. The authors called this the break down of synaptic specificity of long term potentiation[1]. For example, consider two pre-synaptic neurons denoted by A, B that are the inputs of the same post-synaptic neuron C. Consider

[1] Long term potentiation (LTP) is long-lasting enhancement of the synaptic connection between two neurons (weight increase).

the case where A and C are repeatedly activated (fire an output spike) together, but B is inactive. From the Hebb rule, we expect an increase in the synaptic weight only between A and C. However, from [28] the increase in the synaptic weight between B and C should also be observed if the geometric distance between the synapses is small. We can modify the form of Hebb-like learning rules to include this spatial inspecificity in a variety of ways. For example, if $\mathbf{u}_j$ is the position of synapse j then the weight update can be in the form of

$$\Delta w_i \propto \sum_j \delta(x_j, y)\, g(\|\mathbf{u}_i - \mathbf{u}_j\|) \tag{7.2}$$

where g is a decreasing function. Radulescu *et al.* [83] investigated the effects of the above Hebbian inspecificity in two standard unsupervised learning models, PCA and ICA. Their assumption is that the inspecificity may follow from some biological constraints and they analyzed how it reduces the accuracy of calculation of the PCA and the ICA. Later, we propose that there is a *functional* role for this spatial spreading of plasticity rather than regarding it as a biological limitation. We now review the relationship between spatial position and function of neurons.

In many areas of the Central Nervous System (CNS) and the peripheral nervous system, neurons are spatially organized by their functionality. Proximate neurons tend to respond to semantically similar stimuli. For instance, in the retina, neurons are organized by position in the visual field; i.e., adjacent neurons respond to light in adjacent positions in the visual field. In the Cochlea, there is tonotopical organization of the hair cells and auditory nerve fibers [52]. There is also semantic organization beyond sensory input. For instance, in the primary visual cortex (V1), two simple cells that are proximate are usually tuned to different but parallel proximate bars in the visual field. In cases of sensory input, the organization adheres to the physical structure of the sensor. Many models have been put forward to characterize how the organization of neurons in V1 is formed (see e.g., [58, 102, 46]). However, the role of this topographic organization remains unclear. Generally, it is assumed to reflect anatomic constraints such as wiring cost. For instance, Chen *et. al.* [18] evaluated the hypothesis that neuronal placement is determined by minimum wiring costs for a given functional constraint and required connectivity between neurons. They found that the Caenorhabditis elegans wiring diagram reasonably fits this hypothesis. Hyvarinen [45] proposed that topographic

organization also reflects the statistics of the sensory input, such as the statistical dependency structure. He claimed that the topographic representation of dependencies can significantly decrease the number of free parameters required for learning. Based on this and other statistical arguments Hyvarinen claims that topographic organization is useful even if we ignore the anatomical wiring constraints [45].

Our goal here is to relate topographic organization of neurons, the spatial spread of weight update [28] and the use of meta-features to define priors on weights as described in Chapter 6.

7.1 Representing and Using Meta-features in the CNS

In the previous chapters, we defined meta-features as side information we have about each of the features which can be used in addition to the feature statistics to improve learning. Meta-features are useful when there is a relationship between their values and the weights of the classifier. For example, the weights are changed smoothly as a function of meta-features. To relate this to learning in the CNS we take the simplifying assumption that a feature value is represented by the activity of a single neuron. Moreover, we assume that neurons are spatially organized by their meta-features; that is, spatially proximate neurons typically represent features that are proximate in meta-feature space. This is supported by the topographic organization of neurons. Each simple cell in V1 is tuned to a bar at a specific position, orientation and scale in the visual field. The (position, orientation, scale) are the meta-features that describe this neuron. Hence, the topographic organization can be interpreted as a spatial organization of neurons by their meta-features. In general, if the dimension of the meta-feature space is larger than three, it might be embedded in the three (or two) dimensions of the cortex in a way that approximately preserves local distances in the meta-feature space. In other words, the spatial distance between two neurons is related to the distance in the meta-feature space between the features they represent.

In Chapter 6 we showed that a smoothness prior on weights in the meta-feature space can improve generalization in supervised learning. The modification we proposed in Chapter 6 to the standard max-margin classifier shares an important property with the modification

of the Hebb rule in Eq. 7.2. In the standard max-margin approach, the weight of a feature which is constantly zero in the training set, is zero. The weight of a feature which is highly correlated with the label is large. This is similar to what we expect from the Hebb rule. We expect small weights for inactive input synapses, and large weights for input synapses which are highly correlated with the output (post-synaptic) neuron. By adding the smoothness of weight in the meta-feature space prior (see Chapter 6), the weight of a feature, which is constant in the training set, can also be large, if an adjacent feature (in the meta-feature space) is highly correlated with the label. That is, a weight update affects nearby features. Hence, the modification of the original weight update of the max-margin is similar to the modification of the Hebb rule when we incorporate the spatial effects observed by Engert and Bonhoeffer [28]. The functional role of this modification is to incorporate a prior on the relationship between synaptic weights and the position of the synapse (which is related to its position in the meta-feature space). We learn from the weight of one feature (neuron) about the weight of other features (neurons). As shown in Chapter 6, this can boost learning and generalization.

We expect the smoothness of weights as function of position to be reflected in the weights themselves; that is, not only in the prior. Possible evidence for this is the existence of simple and complex cells in V1 [44]. Simple cells respond to bars in specific positions and orientations. Complex cells are less specific for position, and respond to bars at a specific orientation and are invariant to the exact spatial position. In this way, complex cells are invariant to the exact position of a detected edge, which is a desirable property for object recognition (see Section 4.4). One of the models put forward to characterize the implementation of complex cells suggests that each complex cell receives inputs from multiple simple cells responding to the same orientation but in slightly different positions. Most of these simple cells are spatially adjacent, and the weight of all of them is similar.

This analysis can be extended to the possible relation between feature selection based on meta-features and spatial shape of dendrites. Dendrites are limited in their spatial distribution; that is, a single dendrite reaches only a limited volume in the cortex. Therefore, the probability of interaction with other neurons depends on the spatial position of its axon. As above, we use the simplifying assumption that feature value is represented by

Statistical Learning Theory	Neuroscience
Meta-features	Represented by spatial position of neurons and axons
Prior of smooth weights as a function of meta-features	Weight update of spatially proximate synapses [28]
Meta-features of high-level (extracted) features	Semantic topographic organization of feature maps [52, 73]
Feature selection aided by meta-features	Spatial shape and position of dendrites and axons [19]
Individual feature selection	Synaptic rewiring [66]

Table 7.1: Summary of relations between terms in statistical learning and neuroscience.

neuron activity, and that neurons are spatially organized by meta-features. Under these assumptions, the growth of a dendrite to a region in the brain is analogous to feature selection by their meta-features. Instead of selecting individual neurons as candidates for synaptic connection, the dendrite first selects regions in meta-features of relevant features (neurons), and then connects to neurons in these regions. In other words, the probability of connecting to a pre-synaptic neuron depends not only on the joint statistics of the neuron, but also on its spatial position. Thus the biological constraints on dendritic shape also have a functional role in terms of learning. After the general volume in meta-feature space of the relevant pre-synaptic neurons (input features) is determined by the volume of dendrite, individual features can be selected by joint activity. These kinds of changes in local connections between neurons based on activity is known as synaptic rewiring [66].

7.2 Discussion

We suggest a relationship between our framework of learning with meta-features and learning mechanisms in biological neural networks. The spatial arrangement of neurons, dendrites and synapses, together with a synaptic modification mechanism might be the regularization method for achieving invariance to small irrelevant changes, as exists in complex cells in V1, and might be related to the meta-feature based regularization we present in Chapter 6 for handwritten digit recognition. This relation is summarized in Table 7.1.

This relation can be used as a framework to explore relations between spatial learning rules and the organization of neurons and synapses. The prediction of this framework is

that spatial learning rules, like the one found by Engert and Bonhoeffer [28], are present in more areas of the cortex and are related to the functional organization of neurons and synapses of the input layers. We expect that neurons in adjacent spatial positions are likely to have similar effects on higher layers of the processing hierarchy. This is achieved by the prior of smoothness of weights; namely that if neurons are adjacent, related post-synaptic neurons have similar weights.

The framework proposed above is still very preliminary and some of the abstractions we proposed should be addressed by further research. For example, the inspecific Hebb rule found by Engert and Bonhoeffer [28] refers to the position of *synapses* rather than to the position of *neurons*. One way to deal with this difference is to assume that if neurons are in a similar position, then their axons also occupy a similar volume in the cortex. Alternatively, this can potentially lead to richer models that take into account the axon shape as well.

Another issue that needs to be addressed is our definition of meta-features. In the previous chapters meta-features represented prior knowledge, defined as side information that is available in addition to the training instances. However, the position (organization) of neurons in the cortex strongly depends on the statistical properties of neuron activity. Hence, while it seems appealing to use meta-features for the regularization of weights, the definition of meta-features should be extended to include a "summarized" representation of the joint statistics of neuron activity.

Chapter 8

Epilogue

In this thesis we adapted machine learning tools that are usually used for prediction of new instances to predict information about new features. In Chapters 2 and 3 we analyzed generalization properties of clustering and the nearest neighbor algorithm to features which were unobserved by the learner. The basic assumption we used is that the set of observed features is a random subset drawn independently from some huge set of features. In Chapters 4 and 5 we used tools from supervised learning and regression to predict quality of features. For this purpose, we described each feature by a set of meta-features. We showed how this setup is useful for feature selection and feature extraction. In Chapter 6 we used meta-features to define a prior on the weights of a linear classifier. This prior enabled us to get better estimation of the weights, by learning from related features. In Chapter 7 we pointed out several directions for future research that may relate our meta-feature framework and learning in biological neural networks.

The assumption that features are sampled from some distribution may also be adapted for analysis of inductive transfer [5, 106, 15]. In inductive transfer the learner tries to use knowledge acquired in previous tasks to improve performance on the current task. One future research direction, would be to assume that the *tasks* are sampled randomly and independently from some distribution. Then, we can predict and analyze the value of knowledge transferred from current tasks (training set of tasks) to future tasks (currently unobserved

tasks). In addition, we can define "features of tasks", which is a vector of properties that describe information about each task. This may help improve the knowledge transferred between tasks, similar to the way that meta-features help relating features.

In the next section we discuss the analogy between some models in machine learning and learning in humans. Then, we rephrase the research question of learning from observed to unobserved features in a general context with analogies to learning in humans. By doing so, we propose directions for extending our framework.

8.1 Learning in Humans and Machines

Some of the terminology in machine learning draws on analogies to learning in humans. In supervised learning the teacher shows the learner instances and provides their labels (names). One analogy to learning in humans is, for example, a parent who shows a baby objects and says their names ("a doll", "a bed"...). However, there are some fundamental differences between models in machine learning and in humans. Learning in humans does not depend on huge number of *labeled* examples. Most of the time we observe the world without an explicit teacher. After a baby explores the world without a teacher, he or she is "ready" to learn the labels from a few examples. Therefore, a more appropriate machine learning model, where the learner uses both labeled and unlabeled data, is *semi-supervised learning* (see Section 1.3.1).

By greatly oversimplifying learning in humans, we could view it as unsupervised clustering of the objects in the world, augmented by a few labels from the teacher to complete the learning. Nevertheless, this analogy is very limited. When a teacher points to an object and *says* its name ("a doll"), he or she does not directly supply the label as in supervised learning. The learner (baby) needs to identify the word as well. Thus it is not clear whether the viewed object (a doll) is the label of the word that has been heard or whether the heard word is the label of the viewed object.

IMAX [7, 6] is a machine learning model which is related to this symmetry between two sources of observations, where each can be served as label for the other source (see Section 1.3). Recall that the goal of IMAX is to find a common cause in the external world by

maximizing the mutual information between the outputs of two neural modules (e.g., vision and hearing). Theoretically, with a large enough training set, we could learn to recognize both spoken words (or related sounds) and images by applying an IMAX-like principle. The input to one module is the image of object, and the input to the second module is the name of the object (as a spoken word) or related sounds (e.g., barking). However, in practice even with an explicit label, as in supervised learning, the problem of visual object recognition by computer is extremely difficult from a reasonably sized training set. In fact, IMAX does not solve the problem, but suggests an alternative to learning without explicit labels. To return to the analogy of learning in humans, we use the unlabeled data of each modality separately (e.g., by clustering the visual objects and independently clustering heard words), and only in advanced stage of this unsupervised learning do we relate and fine-tune the clustering using an IMAX-like principle from two (or more) input sources.

A semi-supervised algorithm that is based on a principle similar to IMAX is co-training [13] (see Section 1.3.1). Recall that this algorithm includes two classifiers each of which receives input from a different view of the same instance. First, each classifier is trained on the labeled data, using standard supervised learning. Then, each classifier is used to generate labels (predictions) on the unlabeled data. The output labels of each classifier are used to train the other classifier. The algorithm is used iteratively on the unlabeled data, where in general the prediction improves during the iterations. Note, however, that the algorithm is based on first using the labeled instances, and only then using the unlabeled instances. This can be related to fine-tuning of labeling after initial learning, and hence it does not solve the hard problem of initial learning without explicit labels as done by humans. We expect that there is a learning mechanism that makes it possible to extract relevant information from each sensory model separately. Then, we use multiple sensory inputs to link learned concepts and refine them.

Many unsupervised learning models have shown that it is possible to learn which information is relevant to some target, unseen input or concept. For example, lines in an image are very relevant for visual object recognition. Indeed, simple cells in the primary visual cortex are specialized in detecting lines in images. Interestingly, many unsupervised learning models showed that *learning* a simple-cell like response is possible by observing statistical

properties of *local* natural image patches, without explicitly defining higher level concepts like objects (see, e.g., [79]).

This leads to an interesting question: when viewing only a subset of possible measurements of each observation (e.g., a small patch in the visual field or viewing an image without the sound of its name), how can we know what is likely to be relevant for predicting the unseen relevant features of the observations? To approach this problem, we need to make some assumptions about the structure of data in the world, and have some model regarding our access to these data (i.e., what part of it we observe and what unobserved part we need to predict). One common approach is generative models where we assume the existence of hidden variables, and model the relation between them and the observed variables. In Chapter 2 we approached this problem using statistical arguments and modeled the process of observing the features: the set of observed features is a randomly selected subset of features, which we analyzed in a simple setup of clustering.

The main contribution of this approach is to model the relation between observed and unobserved features as some random selection process. This approach enables rigorous mathematical analysis. However, the case we dealt with is rather simple. Future work may tackle a more general question concerning the relation between observed and unobserved features. For example, in many real world problems we typically have some hierarchies and several types of input variables. In these cases, it is not reasonable to assume that the set observed features is a random independently selected subset, and we need to use richer models which incorporate the hierarchies of features. Formulating sampling assumptions on the feature observation model, while defining our knowledge on the features by meta-features may be useful tool to extend the framework.

Bibliography

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 2005.

[2] E. Backer and A. Jain. A clustering performance measure based on fuzzy set decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1981.

[3] P.L. Bartlett. The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 1998.

[4] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *International Conference on Machine Learning (ICML)*, 2004.

[5] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1), 1997.

[6] S. Becker. Mutual information maximization: Models of cortical self-organization. *Network: Computation in Neural Systems*, 7:7–31, 1996.

[7] S. Becker and G. E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163, 1992.

[8] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. Distributional word clusters vs. words for text categorization. 2003.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2003.

[10] A. Bell and T. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[11] J. Blitzer, A. Globerson, and F. Pereira. Distributed latent variable models of lexical co-occurrences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2005.

[12] A. Blum and J. Langford. Pac-mdl bounds. *Learning Theory and Kernel Machines*, 2003.

[13] A. Blum and T. Mitchell. Combining llabeled and unlabeled data with co-training. In *COLT'98 - Proceedings of the 11th Annual Conference on Computational Leanrning Theory*, 1998.

[14] J.S. Breese, D. Heckerman, and K. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

[15] R. Caruana and V. R. de Sa. Promoting poor features to supervisors: Some inputs work better as outputs. In *Advances in Neural Information Processing Systems (NIPS)*, 1997.

[16] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[17] G. Chechik and N. Tishby. Extracting relevant structures with side information. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[18] BL Chen, DH Hall, and DB Chklovskii. Wiring optimization can relate neuronal structure and function. *PNAS*, 2006.

[19] DB Chklovskii, BW Mel, and K. Svoboda. Cortical rewiring and information storage. *Nature*, 2004.

[20] T. M. Cover and J. A. Thomas. *Elements Of Information Theory*. Wiley Interscience, 1991.

[21] K. Crammer. MCSVM_1.0: C Code for Multiclass SVM. 2003. http://www.cis.upenn.edu/~crammer.

[22] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2001.

[23] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3):161–190, 2002.

[24] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 1977.

[25] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

[26] L. Ein-Dor, O. Zuk, and E. Domany. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proc Natl Acad Sci U S A*, 103(15):5923–5928, April 2006.

[27] G. Elidan and N. Friedman. The information bottleneck EM algorithm. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[28] F. Engert and T. Bonhoeffer. Synapse specificity of long-term potentiation breaks down at short distances. *Nature*, 1997.

[29] E. Fix and j. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. Technical Report 4, USAF school of Aviation Medicine, 1951.

[30] N. Friedman, D. Geiger, and Goldszmidt. Bayesian network classifiers. *Machine Learning*, 1997.

[31] N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[32] D. Gabor. Theory of communication. *J. IEE*, 93:429–459, 1946.

[33] R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin based feature selection - theory and algorithms. In *Proc.* 21^{st} *(ICML)*, pages 337–344, 2004.

[34] A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, 2006.

[35] D. Goldberg, D. Nichols, and Terry D. Oki, B. M. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 1992.

[36] D. Gondek and T. Hofmann. Non-redundant data clustering. In *Fourth IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2004.

[37] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 2003.

[38] I. Guyon, A. Saffari, G. Dror, and G. Cawley. Agnostic learning vs. prior knowledge challenge. *International Joint Conference on Neural Networks, IJCNN 2007.*, pages 829–834, 2007.

[39] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46, 2002.

[40] T. S. Han. Nonnegative entropy measures of multivariate symmetric correlations. *Information and Control*, 36:133–156, 1978.

[41] J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[42] M. H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, 1995.

[43] D.O. Hebb. The organization of behavior. 1949.

[44] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. In *The Journal of Physiology*, 1968.

[45] A. Hyvärinen. Topography as a property of the natural sensory world. *Natural Computing*, 2002.

[46] A. Hyvärinen, PO. Hoyer, and M. Inki. Topographic independent component analysis. *Neural Computation*, 2001.

[47] A. Hyvärinen and E. Oja. Independent component analysis by general nonlinear hebbian-like learning rules. *Signal Processing*, 64:301–313, 1998.

[48] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[49] I.T. Jolliffee. *Principal Component Analysis*. Springer Varlag, 1986.

[50] M. W. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Mach. Learn.*, 58:179–216, 2005.

[51] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[52] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*.

[53] S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of IJCNN'98*, volume 1, 1998.

[54] Michael J. Kearns and Umesh V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994.

[55] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proc. 29th ACM Symposium on Theory of Computing*, 1997.

[56] J. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[57] R. Kohavi and G.H. John. Wrapper for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[58] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 2001.

[59] E. Krupka, A. Navot, and N. Tishby. Learning to select features using their properties. *Submitted to Journal of Machine Learning Research*.

[60] E. Krupka and N. Tishby. Generalization in clustering with unobserved features. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[61] E. Krupka and N. Tishby. Incorporating prior knowledge on features into learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[62] E. Krupka and N. Tishby. Generalization from observed to unobserved features by clustering. *Journal of Machine Learning Research*, 2008.

[63] E. Kussul, T. Baidyk, L. Kasatkina, and V. Lukovich. Rosenblatt perceptrons for handwritten digit recognition. In *Int'l Joint Conference on Neural Networks*, pages 1516–20, 2001.

[64] K. Lang. Learning to filter netnews. *Proc. 12th International Conf. on Machine Learning*, pages 331–339, 1995.

[65] F. Lauer and G. Bloch. Incorporating prior knowledge in support vector machines for classification: a review. *Submitted to Neurocomputing*, 2006.

[66] JV Le Bé and H. Markram. Spontaneous and evoked synaptic rewiring in the neonatal neocortex. *PNAS*, 2006.

[67] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[68] S. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *International Conference on Machine Learning (ICML)*, 2007.

[69] S. P. Lloyd. Least squares quantization in pcm. Technical report, Bell Laboratories, 1957. Published in 1982 in IEEE Transactions on Information Theory 28, 128-137.

[70] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[71] H. Markram, J. Lubke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275:213–215, 1997.

[72] B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.

[73] R. Miikkulainen, J. A. Bednar, Y. Choe, and J. Sirosh. *Computational Maps in the Visual Cortex*. Springer, Berlin, 2005.

[74] J. Mutch and D. G. Lower. Multiclass object recognition with sparse, localized features. *cvpr*, 1:11–18, 2006.

[75] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia. Nearest neighbor based feature selection for regression and its application to neural activity. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[76] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[77] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[78] E. Oja. Neural networks, principal component, and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[79] B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 1997.

[80] L. Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15:1101–1253, 2003.

[81] J. Pearl, editor. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[82] J. R. Quinlan. Induction of decision trees. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

[83] A. Radulescu, K. Cox, and P. Adams. Hebbian inspecificity in unsupervised learning. In *Computational and Systems Neuroscience (COSYNE)*, 2007.

[84] R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *International Conference on Machine Learning (ICML)*, 2006.

[85] J. Rissanen. Modeling by the shortest data description. 1978.

[86] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[87] B. Scholköpf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In *Advances in Neural Information Processing Systems (NIPS)*, 1998.

[88] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[89] G. Schwarz. Estimating the dimension of a model. 1978.

[90] M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2002.

[91] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

[92] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[93] S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 2006.

[94] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, July and October 1948.

[95] J. Shawe-Taylor, P.L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 1998.

[96] P. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, pages 239–27, 1996.

[97] P. Y. Simard, Y. A. Le Cun, and Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems (NIPS)*. 1993.

[98] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[99] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000.

[100] P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46, 2002.

[101] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2005.

[102] N. V. Swindale. The development of topography in the visual cortex: a review of models. *Computation in Neural Systems*, 1996.

[103] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[104] B. Taskar, M. F. Wong, and D. Koller. Learning on the test data: Leveraging unseen features. In *International Conference on Machine Learning (ICML)*, 2003.

[105] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

[106] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Proc. 15th Conference on Neural Information Processing Systems (NIPS) 8*, 1996.

[107] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. *Proc. 37th Allerton Conf. on Communication and Computation*, 1999.

[108] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 2002.

[109] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence.*, 1998.

[110] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[111] V. N. Vapnik. *The Nature Of Statistical Learning Theory*. Springer-Verlag, 1995.

[112] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[113] V. N. Vapnik and A. Y. Chervonenkis. On the uniform covergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[114] U. von Luxburg and S. Ben-David. Towards a statistical theory of clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005.

[115] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, 1999.

[116] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggion, and V. Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[117] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems (NIPS)*, 1996.

[118] D. H. Wolpert. On the connection between in-sample testing and generalization error. *Complex Systems*, 1992.

[119] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16, May 2005.

[120] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. *cvpr*, 02:2126–2136, 2006.

[121] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin - Madison, 2007.

הכללה ממאפיינים נצפים למאפיינים חבויים

חיבור לשם קבלת תואר דוקטור לפילוסופיה
מאת

איל קרופקה

הוגש לסנט האוניברסיטה העברית בירושלים בשנת תשס"ח

עבודה זו נעשתה בהנחייתו של פרופסור נפתלי תישבי

תקציר

תזה זו דנה בהיבטים שונים של למידה מסט מאפיינים אחד לסט מאפיינים אחר במסגרת למידה חישובית. אנו מציעים מודל למידה והכללה ממאפיין למאפיין כמודל אפשרי לאשכול (clustering), כשיטה לשיפור ברירת מאפיינים (feature selection) וכשיטה להכללת ידע מוקדם ללמידה מונחה. לבסוף, אנו דנים בקשר אפשרי בין המודלים שפתחנו לבין למידה ברשתות עצביות ביולוגיות. בתקציר זה נתחיל במבוא קצר על למידה חישובית תוך התמקדות בנושאים הרלבנטיים לתזה זו, ובהמשכו נסכם את התוצאות החדשות המרכזיות שהושגו במסגרת התזה.

למידה חישובית עוסקת בנושאים אלגוריתמיים, תיאורטיים ויישומיים של למידה מנתונים. בקצרה, מטרת הלמידה הינה חילוץ ידע שימושי מתוך הנתונים. קיימים מספר מודלים של למידה, כאשר ההבדל בינהם הוא במידע אותו מחלצים וסוג הנתונים בהם מטפלים. המודל הבסיסי ביותר הוא למידה מונחה. בלמידה מונחה הלומד מקבל דוגמאות כאשר כל דוגמא כוללת תבנית והתיוג שלה. בדרך כלל, כל תבנית הינה וקטור של מדידות, כאשר כל מדידה נקראת מאפיין (feature). מטרת הלומד הינה לחלץ פונקציה המקבלת כקלט תבנית ומוציאה כפלט תחזית לתיוג התבנית. לדוגמא, בבעיית זיהוי ספרות בכתב יד יקבל הלומד סט תמונות של ספרות ולכל תמונה תהיה תווית (label) המציינת את הספרה שבתמונה. סט זה נקרא סט אימון. הלומד ילמד כלל המתאים לכל תמונה את הספרה המתאימה. לאחר מכן, תשמש הפונקציה הנלמדת לזיהוי תמונות חדשות. במילים אחרות, נשתמש בפונקציה כדי לחזות את התוויות של תבניות חדשות. אחת השיטות הבסיסיות לסיווג תבנית חדשה, הינה מציאת הדוגמא הקרובה ביותר בסט האימון לתבנית החדשה (למשל לפי מרחק אוקלידי), ושימוש בסיווג דוגמא קרובה זו לסיווג תבנית זו. למשל, בספרות נמצא את התמונה בסט האימון שהכי דומה לתמונה אותה יש לסווג, ונניח שהסיווג שלהם זהה. כלל זה נקרא כלל השכן הקרוב (Nearest neighbor rule).

מושג מרכזי בלמידה חישובית הוא מושג ההכללה. שגיאת האימון של מסווג הינה החלק

היחסי של הסיווגים השגויים מתוך סט האימון. באופן כללי, קל ללמוד פונקציה הממפה כל תבנית מסט האימון לתיוג המתאים לה, ובכך להשיג שגיאת אימון אפס. עם זאת, האתגר המרכזי בלמידה הוא להצליח ללמוד פונקציה שתזהה נכון גם תבניות חדשות, שלא הופיעו בסט האימון. מדד מרכזי לבדיקת ביצועי מערכת לומדת הינה בדיקת שגיאת ההכללה שלה, כלומר הסתברות השגיאה בסיווג דוגמאות חדשות. אחד הכלים התיאורטיים לניתוח אלגוריתם לימוד, הינו חישוב חסם על ההפרש בין שגיאת האימון לשגיאת ההכללה. ההנחה הבסיסית העומדת בחלק גדול מסוגי ניתוח אילו הינה שקיים פילוג הסתברות, שאינו ידוע למסווג, שמתוכו נדגמים דוגמאות האימון. שגיאת ההכללה מתייחסת להסתברות השגיאה בסיווג של דוגמא המוגרלת מאותו פילוג.

כדי להשיג הכללה טובה, נדרש ידע מוקדם על בעיית הלימוד שאנו פותרים. ידע מוקדם (prior knowledge) בלמידה חישובית הינו כל מידע זמין על בעיית הלמידה מעבר לדוגמאות האימון. חלק מהידע המוקדם הינו ספציפי לבעיות מסוימות, למשל בסיווג ספרות מתוך תמונות אנו יכולים להשתמש בידע שהזזה של מיקום הספרה בתמונה לצדדים אינו משנה את סיווגה. למרבה המזל, יש סוגים של ידע מוקדם הרלבנטי לרוב בעיות הלימוד המעשיות. למשל, הנחה המתקיימת כמעט בכל בעיות הלימוד המעשיות הינה שלשתי נקודות הקרובות מאוד אחת לשנייה במרחב המאפיינים סביר שהתווית תהיה זהה. קיום הנחות מסוג זה, מאפשר תכנון אלגוריתמים היכולים להתאים למגוון גדול של בעיות, כדוגמת Support Vector Machine (SVM) [111]. עם זאת, בדרך כלל שימוש בידע מוקדם ספציפי לבעיה, מאפשר שיפור ביצועי המסווג.

מודל מרכזי אחר בלמידה נקרא אשכול (clustering). במודל זה מקבל הלומד מספר כלשהו של עצמים או תבניות (ללא תוויות), ומחלק אותם לקבוצות לפי דמיון בין העצמים. במודל זה, בניגוד ללמידה מונחה, לא נדרש להניח שיש עצמים נוספים שאינם נצפים. במקרים רבים אין מדד ברור להערכת איכות החלוקה, מכיוון שאין משימת חיזוי ברורה כמו בלמידה מונחה. בחלק מהמקרים אנו מצפים שהחלוקה לקבוצות תתאים לתוויות שאינן ידועות לאלגוריתם הלמידה. במקרה זה לא ברור איך לבחור את החלוקה כך שתתאים לתוויות לא ידועות. הבעיה קשה במיוחד, מכיוון שמדד איכות האשכול תלוי בבחירה המסוימת של התוויות. לדוגמא, אשכול מסמכים לפי נושאים עשוי להיות שונה מאוד מאשכול לפי מחבר המסמך.

בפרק 2 אנו מציעים מודל אשכול שמטרתו לחזות מאפיינים לא נצפים של העצמים. אנו מניחים שכל העצמים נצפים על־ידי הלומד, אך חלק מהמאפיינים של אותם עצמים אינם

גלויים ללומד. מטרת הלומד היא ללמוד מתוך סט המאפיינים הנצפה ולהכליל לסט מאפיינים חדש, שאינו נצפה בזמן הלמידה. לדוגמא, בחלוקה של פירות לקבוצות בהתבסס על תכונות נצפות כמו צורה, צבע וגודל, אנו מעוניינים לחזות תכונות לא נצפות כגון ערך תזונתי ורעילות. דוגמא אחרת, כאשר מחלקים אנשים לקבוצות לפי הדירוג שנתנו לסט סרטים, מטרת הלומד היא שהחלוקה של האנשים לקבוצות תתאים לסרטים שלא דורגו, ואף סרטים שעדיין לא נוצרו. על מנת לנתח את תכונות ההכללה של הלומד, אנו נדרשים להניח הנחה כלשהי על מבנה הבעיה. ההנחה הבסיסית בה אנו משתמשים הינה שסט התכונות הנצפות, הינו תת־סט שנבחר בצורה אקראית מתוך תת־סט גדול מאוד של תכונות. אנו מראים שתחת מספר תנאים טכניים, סיווג העצמים לפי תת־הסט הנצפה מגיע לסיווג טוב באיכותו כמעט כמו סיווג לפי כל תכונות האובייקט. המדד שלנו לאיכות הסיווג הינו האינפורמציה ההדדית [20] הממוצעת בין החלוקה לקבוצות לבין סט התכונות הלא נצפה. אנו מוכיחים משפטי הכללה המשווים בין התאמת החלוקה לתכונות נצפות לעומת ההתאמה לתכונות לא נצפות, וזאת באנלוגיה למשפטי ההכללה בלמידה מונחה. אנו משתמשים בכלים אילו על מנת לנתח תכונות הכללה של שני אלגוריתמי חלוקה לקבוצות מוכרים: Maximum Likelihood Mixture Model ו־k-means [41]. אנו מדגימים את תצורת לימוד זו על אשכול מסמכים ועל אשכול צופים לפי טעם בסרטים, כאשר אנו מראים כיצד חלוקה זו מתאימה לסרטים שלא היו ידועים לאלגוריתם האשכול.

בפרק 3 אנו מרחיבים את מסגרת העבודה מעבר לאשכול, ומנתחים תכונות הכללה למאפיינים חבויים של כלל הלמידה מבוסס שכן קרוב. נציג חסם הכללה המתייחס למרחק של המאפיינים הנצפים של השכן הקרוב לעומת מרחק המאפיינים החבויים של אותו שכן.

למידה והכללה מקבוצת מאפיינים אחת לקבוצה מאפיינים אחרת שימושית גם בלמידה מונחה. ברוב מודלי הלמידה, דוגמאות מיוצגות כוקטור של מאפיינים. ברוב בעיות העולם האמיתי, יש לנו מידע נוסף על כל אחד מהמאפיינים, מעבר לערכם בכל דוגמא. מידע נוסף זה יכול להיות מיוצג על ידי סט מאפיינים נוסף, שאנו מכנים מטא־מאפיינים (meta-features). למשל, בתמונות של ספרות בכתב יד, כאשר המאפיינים הינם ערכי הפיקסלים, המטא־מאפיינים יכולים להיות המיקום (שורה, עמודה) של כל אחד מהפיקסלים. נשים לב שערכם של המטא־מאפיינים אינו תלוי בסט האימון, למשל, אנו יודעים את מיקום כל פיקסל (שורה, עמודה) ללא קשר לסט האימון. אנו משתמשים במטא־מאפיינים כדי לשפר את הביצועים של אלגוריתמי למידה מונחה בשתי שיטות שונות. בשיטה הראשונה, אנו משפרים את תהליך ברירת מאפיינים (ראה פירוט בהמשך) בהתבסס על מטא־מאפיינים ותצפיות של

ערכי תת־קבוצה של מאפיינים. השיטה השנייה משפרת תכונות הכללה של למידה מונחה ע"י שימוש במטא־מאפיינים כידע מוקדם על בעיית הלימוד.

ברירת מאפיינים (feature selection) הינה משימה הכוללת בחירת תת־קבוצה קטנה של מאפיינים המספיקים לחיזוי אמין של התוויות. עבודה על קבוצה קטנה יותר של תכונות מאפשרת להשתמש באלגוריתמים מהירים יותר לסיווג, ולעיתים אף משפרת דיוק המסווג באמצעות התעלמות ממאפיינים רועשים, ושימוש במאפיינים טובים בלבד. בפרק 4, במקום לנסות להחליט אילו מאפיינים הם הטובים ביותר באופן ישיר, אנו מציעים אלגוריתם שלומד לחזות את איכות המאפיינים מתוך המטא־מאפיינים שלהם. גישה זו מאפשרת להשמיט מאפיינים לא רלוונטיים על־ידי חיזוי איכותם מתוך מטא־מאפיינים ללא צורך בחישוב ערכם בסט האימון. באמצעות שיטה זו, אנו מציעים אלגוריתם בחירת מאפיינים חדש, המסוגל לחפש מאפיינים טובים בצורה מאוד יעילה כאשר קבוצת המאפיינים מהם יש לבחור מאוד גדולה. אלגוריתם זה מוריד מצורה משמעותית את מספר המאפיינים שיש לחשב. אנו מציגים ניתוח של חסם הכללה של אלגוריתם המשלב בחירת מאפיינים באמצעות מטא־מאפיינים, מתוך ניתוח זה ניתן להבין היתרון של שימוש במטא־מאפיינים. אנו מדגימים את ישום של האלגוריתם על זיהוי עצמים בתמונות. בנוסף, אנו מראים כיצד שימוש במטא־מאפיינים מאפשר להבין את הבעיה טוב יותר - במקום לקבל רשימה ארוכה של אילו מאפיינים טובים, פלט האלגוריתם כולל אפיון של מאפיינים טובים. קיימות עבודות המראות שבזיהוי עצמים בתמונות, אפשר למצוא קבוצת מאפיינים המתאימה לזיהוי קבוצה גודלה מאוד של עצמים. סט מאפיינים זה מכונה מילון אוניברסאלי (universal dictionary). בעבודה זו, אנו מנתחים תכונות של המאפיינים הנכללים במילון זה, ומראים שניתן לחזות את איכות המאפיינים בצורה טובה מתוך מטא־מאפיינים. בכך, אנו יכולים להבין את תכונות המאפיינים שיש לכלול במילון אוניברסאלי.

מטא־מאפיינים יכולים לשמש גם להחלטה על המאפיינים שיש לבחור בתהליך מיצוי מאפיינים (feature extraction). במיצוי מאפיינים אנו מחשבים מאפיינים חדשים שהינם פונקציה של המאפיינים הנמדדים ישירות (מאפיינים גולמיים) של העצם. למשל כאשר המאפיין הגולמי הינו ערך פיקסל, אנו יכולים ליצור מאפיינים חדשים שהם מכפלת זוגות פיקסלים. לפונקציות שאנו יוצרים מוגדרים על־ידי הקלט שלהם, ולעיתים פרמטרים נוספים שאנו יכולים להגדירם כמטא־מאפיינים. בתיאוריה, יכולנו ליצור את כל הפונקציות האפשריות ממשפחה פרמטרית מסוימת, ולאחר מכן להשתמש בברירת מאפיינים. אך בבעיות מעשיות, בדרך כלל מספר הפונקציות האפשריות גדול מאוד ולכן פתרון זה אינו מעשי. לכן, נדרשת

שיטה המאפשרת חיפוש יעיל של המאפיינים הרלבנטיים, ללא חישוב ישיר של כל המאפיינים. בפרק 5 אנו מציגים ומנתחים דרך כזו, ומדגימים אותה על פתרון בעיית זיהוי ספרות בכתב יד.

בפרק 6 אנו מציעים גישה חדשה לשימוש במטא־מאפיינים ללמידה מונחה. במסגרת גישה זו אנו מניחים שהמסווג משייך משקל לכל אחד מהמאפיינים, כמו במקרה של מסווג לינארי. אנו מגדירים פונקציה הממפה מוקטור מטא־מאפיינים למשקל, ובתהליך הלימוד ההתאמה בין מאפיין למשקל שלו מתבצעת באמצעות פונקציה זו המופעלת על המטא־מאפיינים שלו. ההנחה הבסיסית שלנו הינה שפונקציה זו הינה פונקציה חלקה, כלומר ערכי המשקל המשויך למאפיינים הקרובים במרחב המטא־מאפיינים הינם דומים. לדוגמא, בבעיית זיהוי כתב יד, כאשר המאפיינים הם רמות האפור של הפיקסלים, נגדיר את המטא־מאפיינים כמיקום הפיקסל (שורה, עמודה). במקרה זה משמעות הנחת החלקות הינה שלפיקסלים הסמוכים במיקום סביר שיהיה משקל דומה.

מתוך גישה זו של שימוש במטא־מאפיינים, אנו גוזרים אלגוריתם למידה מעשי, המשפר את יכולת ההכללה. אלגוריתם זה משתמש בתהליך גאוסי (Gaussian Process) על־מנת להגדיר את המידע המוקדם שלנו: פילוג ההסתברות של הקשר בין משקלי מאפיינים שונים לפי ערכי המטא־מאפיינים שלהם. אנו מיישמים אלגוריתם זה לתכנון מסווג לזיהוי ספרות בכתב יד. המאפיינים בהם אנו משתמשים הם מכפלות של מספר פיקסלים. מטא־המאפיינים מתארים את מיקום והיחס המרחבי בין הפיקסלים ששימשו לחישוב על מאפיין. מתוך תוצאות הניסוי האמפירי, אנו משיגים דיוק גבוהה יותר בעלות חישובית נמוכה יותר ממספר שיטות קודמות [87] להכללת ידע מוקדם ל־SVM. בניסוי נוסף, אנו מדגימים כיצד אנו משפרים באמצעות מטא־מאפיינים יכולת חיזוי של דירוג סרטים על־ידי צופים. כמו־כן אנו דנים ביתרונות התיאורטיים של הכללת מטא־מאפיינים בלמידה.

בפרק 7 אנו דנים באפשרות שעקרון השימוש במטא־מאפיינים כמידע נוסף המשפר הכללה ישים ללמידה ברשתות נוירונים (תאי עצב) ביולוגיות. אנו מצביעים על קשר אפשרי בין שימוש במטא־מאפיינים כמידע מוקדם לשיפור למידה למספר מנגנוני למידה בנוירונים המושפעים ממיקום גיאומטרי שלהם. לצורך זה אנו מניחים מספר הנחות הפשטה. הנחה ראשונה היא שערך של מאפיין מיוצג על ידי פעילות הנוירון. הנחה שניה היא שנוירונים במערכת העצבים מסודרים בצורה מרחבית לפי ערכי המטא־מאפיינים הקשורים לכל נוירון. כלומר, נוירונים המייצגים מאפיינים דומים ממוקמים סמוך אחד לשני. דמיון בין מאפיינים מתייחס כאן לקרבה במרחב המטא־מאפיינים. הנחה זו נתמכת בקיום מפות ארגון (organization maps)

בקליפת המוח. למשל, בקליפת המוח קיים אזור (V1) השייך לעיבוד מידע חזותי. באזור זה קיימים נוירונים המגיבים לקווים, כאשר כל נוירון מגיב לקו במיקום מסוים בשדה הראיה וזווית מסוימת. נוירונים הסמוכים מרחבית זה לזה יגיבו בדרך כלל לקווים במיקומים וזוויות דומים. אם נגדיר את המטא־מאפיינים של כל נוירון כמיקום וכיוון הקו בשדה הראיה אליהם הוא מגיב, נוכל לפרש את הארגון של הנוירונים כארגון לפי מטא־מאפיינים. ההנחה השלישית היא שכלל הלמידה (עדכון המשקולות) של נוירון תלוי לא רק בפעילות של הנוירונים, אלא גם במיקומם המרחבי היחסי. בפרט, קיים מתאם בעדכון משקולות לנוירוני קלט סמוכים. קיימות עדויות לסוג זה של למידה במח: למשל בעבודה של Engert ו־Bonhoeffer [28] נמצא שמשקולות (קשר סינפטי) לנוירוני קלט סמוכים, נוטים להתחזק ביחד, אפילו אם חלק מהנוירונים אינם פעילים. כלומר, חיזוק הקשר מושפע מהקרבה הפיזית בין הסינפסות. אנו מציעים קשר אפשרי בין עדות זו על השפעת המרחק הפיזי בין סינפסות על עדכון המשקולות לבין השפעת המטא־מאפיינים על כלל הלמידה שמוצג בפרק 6 של תזה זו.