Hierarchical Regularization Cascade for Joint Learning

Alon Zweig ALON.ZWEIG@MAIL.HUJI.AC.IL
Daphna Weinshall DAPHNA@CS.HUJI.AC.IL
School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel, 91904

Abstract

As the sheer volume of available benchmark datasets increases, the problem of joint learning of classifiers and knowledge-transfer between classifiers, becomes more and more relevant. We present a hierarchical approach which exploits information sharing among different classification tasks, in multi-task and multi-class settings. It engages a top-down iterative method, which begins by posing an optimization problem with an incentive for large scale sharing among all classes. This incentive to share is gradually decreased, until there is no sharing and all tasks are considered separately. The method therefore exploits different levels of sharing within a given group of related tasks, without having to make hard decisions about the grouping of tasks. In order to deal with large scale problems, with many tasks and many classes, we extend our batch approach to an online setting and provide regret analysis of the algorithm. We tested our approach extensively on synthetic and real datasets, showing significant improvement over baseline and state-of-the-art methods.

1. Introduction

Information sharing can be a very powerful tool in various domains. Consider visual object recognition where different categories typically share much in common: cars and trucks can be found on the road and both classes have wheels, cows and horses have four legs and can be found in the field together, etc. Accordingly, different information sharing approaches have been developed (Torralba et al., 2007; Obozinski et al., 2007; Quattoni et al., 2008; Amit et al., 2007; Duchi & Singer, 2009; Kim & Xing, 2010; Shalev-Shwartz et al., 2011; Kang et al., 2011) .

In this paper we focus on the multi-task and multi-class settings, where the learning is performed jointly for many tasks or classes. Multi-task (Caruana, 1997) is a setting where there are several individual tasks which are trained jointly, e.g., character recognition for different writers. Multi-class is the case where there is only a single classification task involving several possible labels (object classes), where the task is to assign each example a single label.

Intuitively the more data and tasks or classes there are, the more one can benefit from sharing information between them. Recent approaches (Obozinski et al., 2007; Quattoni et al., 2008; Amit et al., 2007; Duchi & Singer, 2009; Shalev-Shwartz et al., 2011) to information sharing consider all tasks as a single group without discriminating between them. However, applying such approaches to datasets with many diverse tasks focus the learning on shared information among **all** tasks, which might miss out on some relevant information shared between a smaller group of tasks.

To address this challenge, our work takes a hierarchical approach to sharing by gradually enforcing different levels of sharing, thus scaling up to scenarios with many tasks. The basic intuition is that it is desirable to be able to share a lot of information with a few related objects, while sharing less information with a larger set of less related objects. For example, we may encourage modest information sharing between a wide range of recognition tasks such as all road vehicles, and separately seek more active sharing among related objects such as all types of trucks.

Previous work investigating the sharing of information at different levels either assume that the structure of sharing is known, or solve the hard problem of clustering the tasks into sharing groups (Torralba et al., 2007; Kang et al., 2011; Jawanpuria & Nath, 2012; Kim & Xing, 2010; Zhao et al., 2011). The clustering approach solves a hard problem and can be used most ef-

Appearing in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013.

fectively with smaller data-sets, while clearly when the hierarchy is known methods which take advantage of it should be preferred. But under those condition where these methods are not effective, our implicit approach enables different levels of sharing without knowing the hierarchy or finding it explicitly.

More specifically, we propose a top-down iterative feature selection approach: It starts with a high level where sharing features among all classes is induced. It then gradually decreases the incentive to share in successive levels, until there is no sharing at all and all tasks are considered separately in the last level. As a result, by decreasing the level of incentive to share, we achieve sharing between different subsets of tasks. The final classifier is a linear combination of diverse classifiers, where diversity is achieved by varying the regularization term.

The diversity of regularization we exploit is based on two commonly used regularization functions: the l_1 norm (Tibshirani, 1996) which induces feature sparsity, and the l_1/l_2 norm analyzed in (Obozinski et al., 2007) which induces feature sparsity while favoring feature sharing between all tasks. Recently the sparse group lasso (Friedman et al., 2010) algorithm has been introduced, a linear combination of the lasso and group-lasso (Yuan & Lin, 2006) algorithms, which can yield sparse solutions in a selected group of variables, or in other words, it can discover smaller groups than the original group constraint (l_1/l_2).

The importance of hierarchy of classes (or taxonomy) has been acknowledged in several recent recognition approaches. A supervised known hierarchy has been used to guide classifier combination (Zweig & Weinshall, 2007), learn distance matrices (Hwang et al., 2011; Verma et al., 2012), induce orthogonal transfer down the hierarchy (Zhou et al., 2011) or detect novel classes (Weinshall et al., 2008). Recent work on multi-class classification (Bengio et al., 2010; Gao & Koller, 2011; Yang & Tsang, 2011) has also tried to infer explicitly some hierarchical discriminative structure over the input space, which is more efficient and accurate than the traditional multi-class flat structure. The goal in these methods is typically to use the discovered hierarchical structure to gain efficient access to data. This goal is in a sense orthogonal (and complementary) to our aim at exploiting the implicit hierarchical structure for information sharing for both multi-task and multi-class problems.

The main contribution of this paper is to develop an implicit hierarchical regularization approach for information sharing in multi-task and multi-class learning (see Section 2). Another important contribution is the extension to the online setting where we are able to consider a lot more learning tasks simultaneously, thus benefiting from the many different levels of sharing in the data (see Section 3 for algorithm description and regret analysis). In Section 4 we describe extensive experiments on both synthetic and seven popular real datasets. In Section 5 we briefly present the extension of our approach to a knowledge-transfer setting. The results show that our algorithm performs better than baseline methods chosen for comparison, and state of the art methods described in (Kang et al., 2011; Kim & Xing, 2010). It scales well to large data scenarios, achieving significantly better results even when compared to the case where an explicit hierarchy is known in advance (Zhao et al., 2011).

2. Hierarchical regularization cascade

We now describe our learning approach, which learns while sharing examples between tasks. We focus only on classification tasks, though our approach can be easily generalized to regression tasks.

Notations Let $\mathbf{k}$ denote the number of tasks or classes. In the multi-task setting we assume that each task is binary, where $\mathbf{x} \in \mathcal{R}^{\mathbf{n}}$ is a datapoint and $y \in \{-1, 1\}$ its label. Each task comes with its own sample set $\mathbf{S}^i = \{(\mathbf{x}_s, \mathbf{y}_s)\}_{s=1}^{\mathbf{m}_i}$, where $\mathbf{m}_i$ is the sample size and $i \in \{1...\mathbf{k}\}$. In the multi-class setting we assume a single sample set $\mathbf{S} = \{(\mathbf{x}_s, \mathbf{y}_s)\}_{s=1}^{\mathbf{m}}$, where $\mathbf{y}_s \in \{1..k\}$. Henceforth, when we refer to $\mathbf{k}$ classes or tasks, we shall use the term tasks to refer to both without loss of generality.

Let $\mathbf{n}$ denote the number of features, matrix $\mathbf{W} \in \mathcal{R}^{\mathbf{n}\times\mathbf{k}}$ the matrix of feature weights being learnt jointly for the $\mathbf{k}$ tasks, and $\mathbf{w}^i$ the i'th column of $\mathbf{W}$. Let $\mathbf{b} \in \mathcal{R}^{\mathbf{k}}$ denote the vector of threshold parameters, where $\mathbf{b}^i$ is the threshold parameter corresponding to task i. $||\mathbf{W}||_1$ denotes the l_1 norm of $\mathbf{W}$ and $||\mathbf{W}||_{1,2}$ denotes its l_1/l_2 norm- $||\mathbf{W}||_{1,2} = \sum_{j=1}^{\mathbf{n}} ||\mathbf{w}_j||_2$, where $\mathbf{w}_j$ is the j'th row of matrix $\mathbf{W}$ and $||\mathbf{w}_j||_2$ its l_2 norm.

The classifiers we use for the i'th task are linear classifiers of the form $\mathbf{f}^i(\mathbf{x}) = \mathbf{w}^i * x + \mathbf{b}^i$. Binary task classification is obtained by taking the sign of $\mathbf{f}^i(\mathbf{x})$, while multi-class classification retrieves the class with maximal value of $\mathbf{f}^i(\mathbf{x})$.

To simplify the presentation we henceforth omit the explicit reference to the bias term $\mathbf{b}$; in this notation $\mathbf{b}$ is concatenated to matrix $\mathbf{W}$ as the last row, and each datapoint x has 1 added as its last element. Whenever the regularization of $\mathbf{W}$ is discussed, it is assumed that the last row of $\mathbf{W}$ is not affected. The classifiers now

take the form $\mathbf{f}^i(\mathbf{x}) = \mathbf{w}^i * x$.

To measure loss we use the binary and multi-class hinge loss functions (Crammer & Singer, 2002). The multitask loss is defined as follows:

$$\mathbf{L}(\{\mathbf{S}^i\}_{i=1}^{\mathbf{k}}, \mathbf{W}) = \sum_{i=1}^{k} \sum_{s \in \mathbf{S}^i} \max(0, 1 - \mathbf{y}_s * \mathbf{f}^i(\mathbf{x}_s)) \quad (1)$$

Without joint regularization this is just the sum of the loss of $\mathbf{k}$ individual tasks. The multi-class loss is defined as:

$$\mathbf{L}(\mathbf{S}, \mathbf{W}) = \sum_{s \in \mathbf{S}} \max(0, 1 + \max_{\mathbf{y}_s = i, j \neq i} (\mathbf{f}^j(\mathbf{x}_s) - \mathbf{f}^i(\mathbf{x}_s))) \quad (2)$$

For brevity we will refer to both functions as $\mathbf{L}(\mathbf{W})$. Note that the choice of the hinge loss is not essential, and any other smooth convex loss function (see (Wright et al., 2009)) can be used.

2.1. Hierarchical regularization

We construct a hierarchy of regularization functions in order to generate a diverse set of classifiers that can be combined to achieve better classification. The construction of the regularization cascade is guided by the desire to achieve different levels of information sharing among tasks. Specifically, at the highest level in the hierarchy we encourage classifiers to share information among all tasks by using regularization based on the l_1/l_2 norm. At the bottom of the hierarchy we induce sparse regularization of the classifiers with no sharing by using the l_1 norm. Intermediate levels capture decreasing levels of sharing (going from top to bottom), by using for regularization a linear combination of the l_1 and l_1/l_2 norms. We denote the regularization term of level l by ψ^l:

$$\psi^l(\mathbf{W}) = \phi^l((1 - \lambda^l)||\mathbf{W}||_{1,2} + \lambda^l||\mathbf{W}||_1) \quad (3)$$

where λ^l is the mixing coefficient and ϕ^l is the regularization coefficient. The regularization coefficient of the last row of $\mathbf{W}^l$ corresponding to bias $\mathbf{b}$ is 0.

For each individual task (column of $\mathbf{W}^l$) we learn L classifiers, where each classifier is regularized differently. Choosing the L mixing terms $\lambda^l \in [0..1]$ diversely results in inducing L different levels of sharing, with maximal sharing at $\lambda^l = 0$ and no incentive to share at $\lambda^l = 1$.[1]

2.2. Cascade algorithm

Learning all diversely regularized classifiers jointly involves a large number of parameters which increases multiplicatively with the number of levels L. A large number of parameters could harm the generalization properties of any algorithm which attempts to solve the optimization problem directly. We therefore developed an iterative approach presented in Algorithm 1, where each level is optimized separately using the optimal value from higher levels in the hierarchy.

Specifically, we denote by L the preset number of levels in our algorithm. In each level only a single set of parameters $\mathbf{W}^l$ is being learnt, with regularization uniquely defined by λ^l. We start by inducing maximal sharing with $\lambda^1 = 0$. As the algorithm proceeds λ^l monotonically increases, inducing decreased amount of sharing between tasks as compared to previous steps. In the last level we set $\lambda^L = 1$, to induce sparse regularization with no incentive to share.

Thus starting from $l = 1$ up to $l = L$, the algorithm for *sparse group learning cascade* solves

$$\mathbf{W}^l = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\mathbf{W} + \mathbf{W}^{l-1}) + \psi^l(\mathbf{W}) \quad (4)$$

The learnt parameters are aggregated through the learning cascade, where each step l of the algorithm receives as input the learnt parameters up to that point- $\mathbf{W}^{l-1}$. Thus the combination of input parameters learnt earlier together with a decrease in incentive to share is intended to guide the learning to focus on more task/class specific information as compared to previous steps.

Note also that this sort of parameter passing between levels works only in conjunction with the regularization; without regularization, the solution of each step is not affected by the solution from previous steps. In our experiments we set $\lambda^l = \frac{l-1}{L-1}$ for all $l \in \{1..L\}$, while the set of parameters $\{\phi^l\}_{l=1}^L$ is chosen using cross-validation.

Algorithm 1 Regularization cascade

Input : L , $\{\lambda^l\}_{l=1}^L$, $\{\phi^l\}_{l=1}^L$
Output : $\mathbf{W}$

1. $\mathbf{W}^1 = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\mathbf{W}) + \phi^1||\mathbf{W}||_{1,2}$
2. for $l = 2$ to L
   (a) $\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\mathbf{W} + \mathbf{W}^{l-1}) + \phi^l((1 - \lambda^l)||\mathbf{W}||_{1,2} + \lambda^l||\mathbf{W}||_1)$
   (b) $\mathbf{W}^l = \mathbf{W}^{l-1} + \mathbf{W}$
3. $\mathbf{W} = \mathbf{W}^L$

[1] Note that while each regularization term ψ^l induces the sparsity of $\mathbf{W}^l$, the output classifier $\sum_{l=1}^L \mathbf{W}^l$ may not be sparse.

2.3. Batch optimization

At each step of the cascade we have a single unconstrained convex optimization problem, where we minimize over a smooth convex loss function summed with a non-smooth regularization term (4). This type of optimization problems has been studied extensively in the optimization community in recent years (Wright et al., 2009; Beck & Teboulle, 2009). We used two popular optimization methods (Daubechies et al., 2004; Beck & Teboulle, 2009), which converge to the single global optimum with rate of convergence $O(\frac{1}{T^2})$ for (Beck & Teboulle, 2009). Both are iterative procedures which solve at iteration t the following sub-problem:

$$\min_{\Theta^t}(\Theta^t - \Theta^{t-1})\nabla\mathbf{L}(\Theta^t) + \frac{\alpha^t}{2}||\Theta^t - \Theta^{t-1}||_2^2 + \phi\psi(W^t)$$

where $\psi(W^t) = ((1-\lambda)||\mathbf{W}^t||_{1,2} + \lambda||\mathbf{W}^t||_1)$ and α^t is a constant factor corresponding to the step size of iteration t. This sub-problem has a closed form solution presented in (Sprechmann et al., 2011), which yields an efficient implementation for solving a single iteration of Algorithm 1. The complexity of the algorithm is L times the complexity of solving (4).

3. Online algorithm

When the number of training examples is very large, it quickly becomes computationally prohibitive to solve (4), the main step of Algorithm 1. We therefore developed an online algorithm which solves this optimization problem by considering one example at a time - the set of parameters $\mathbf{W}^l$ is updated each time a new mini-sample appears containing a single example from each task.

In order to solve (4) we adopt the efficient dual-averaging method proposed by (Xiao, 2010), which is a first-order method for solving stochastic and online regularized learning problems. Specifically we build on the work of (Yang et al., 2010), who presented a closed form-solution for the case of sparse group lasso needed for our specific implementation of the dual-averaging approach. The update performed at each time step by the dual averaging method can be written as:

$$\mathbf{W}_t = \operatorname*{argmin}_{\mathbf{W}} \bar{\mathbf{U}}\mathbf{W} + \psi^l(\mathbf{W}) + \frac{\gamma}{\sqrt{t}}h(\mathbf{W}) \qquad (5)$$

where $\mathbf{U}$ denotes the subgradient of $\mathbf{L_t}(\mathbf{W} + \mathbf{W}^{l-1})$ with respect to $\mathbf{W}$, $\bar{\mathbf{U}}$ the average subgradient up to time $\mathbf{t}$, $h(\mathbf{W}) = \frac{1}{2}||\mathbf{W}||_2^2$ an auxiliary strongly convex function, and γ a constant which determines the convergence properties of the algorithm.

Algorithm 2 describes our online algorithm. It follows from the analysis in (Xiao, 2010) that the run-time and memory complexity of the online algorithm based on update (5) is linear in the dimensionality of the parameter-set, which in our setting is nk. Note that in each time step a new example from each task is processed through the entire cascade before moving on to the next example.

Algorithm 2 Online regularization cascade

Input : L, γ, $\{\phi^l\}_{l=1}^L$, $\{\lambda^l\}_{l=1}^L$
Initialization: $\hat{\mathbf{W}}_0^l = 0$, $\bar{\mathbf{U}}_0^l = 0\ \forall l \in \{1..L\}$, $\mathbf{W}_t^0 = 0\ \forall t$

1. for t = 1,2,3,... do
   (a) for $l = 1$ to L
      i. Given $\mathbf{L_t}(\mathbf{W}+\mathbf{W}^{l-1})$, compute a subgradient $\mathbf{U}_t^l \in \partial\mathbf{L_t}(\mathbf{W} + \mathbf{W}^{l-1})$
      ii. $\bar{\mathbf{U}}_t^l = \frac{t-1}{t}\bar{\mathbf{U}}_{t-1}^l + \frac{1}{t}\mathbf{U}_t^l$
      iii. $\hat{\mathbf{W}}_t^l = \operatorname*{argmin}_{\mathbf{W}} \bar{\mathbf{U}}_t^l\mathbf{W} + \psi^l(\mathbf{W}) + \frac{\gamma}{\sqrt{t}}h(\mathbf{W})$
      iv. $\mathbf{W}_t^l = \mathbf{W}_t^{l-1} + \hat{\mathbf{W}}_t^l$
   (b) $\mathbf{W}_t = \mathbf{W}_t^L$

3.1. Regret analysis

In online algorithms, regret measures the difference between the accumulated loss over the sequence of examples produced by the online learning algorithm, as compared to the loss with a single set of parameters used for all examples and optimally chosen in hindsight. For T iterations of the algorithm, we can write the regret as $\mathbf{R}_T(\mathbf{W}_*) = \sum_{t=1}^T(\mathbf{L}_t(\mathbf{W}_t) + \psi(\mathbf{W}_t) - \mathbf{L}_t(\mathbf{W}_*) - \psi(\mathbf{W}_*))$, where $\mathbf{W}_* = \operatorname*{argmin}_{\mathbf{W}} \sum_{t=1}^T(\mathbf{L}_t(\mathbf{W}) + \psi(\mathbf{W}))$.

At each time step $\mathbf{t}$, Algorithm 2 chooses for each level $l > 1$ of the cascade the set of parameters $\hat{\mathbf{W}}_t^l$, to be added to the set of parameters $\mathbf{W}_t^{l-1}$ calculated in the previous level of the cascade. Thus the loss in level l at time t $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}) = \mathbf{L_t}(\hat{\mathbf{W}} + \mathbf{W}_t^{l-1})$ depends on both the example at time t and the estimate $\mathbf{W}_t^{l-1}$ obtained in previous learning stages of the cascade. We define the following regret function that compares the performance of the algorithm at level l to the best choice of parameters for all levels up to level l:

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) = \sum_{t=1}^T(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) + \psi^l(\hat{\mathbf{W}}_t^l)) - \sum_{t=1}^T(\mathbf{L}_{\mathbf{t},\mathbf{W}_*^{l-1}}(\hat{\mathbf{W}}_*^l) + \psi^l(\hat{\mathbf{W}}_*^l)) \qquad (6)$$

where $\hat{\mathbf{W}}_*^l = \operatorname*{argmin}_{\mathbf{W}} \sum_{t=1}^T(\mathbf{L}_{t,\mathbf{W}_*^{l-1}}(\mathbf{W}) + \psi^l(\mathbf{W}))$,

$\mathbf{W}_*^0 = 0$ and $\mathbf{W}_*^l = \sum_{k=1}^{l} \hat{\mathbf{W}}_*^k$.

We note that the key difference between the usual definition of regret above and the definition in (6) is that in the usual regret definition we consider the same loss function for the learnt and optimal set of parameters. In (6), on the other hand, we consider two different loss functions - $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$ and $\mathbf{L}_{t,\mathbf{W}_*^{l-1}}$, each involving a different set of parameters derived from previous levels in the cascade.

We now state the main result, which provides an upper bound on the regret (6) and thus proves convergence.

Theorem 1. *Suppose there exist* $\mathbf{G}$ *and* $\mathbf{D}$ *such that* $\forall \mathbf{t}, l \ ||\mathbf{U}_t^l|| < \mathbf{G}$ *and* $h(\mathbf{W}) < \mathbf{D}^2$*, and suppose that* $\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \geq -C\sqrt{T}$*; then*

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq A\sqrt{T} + B(T+1)^{\frac{3}{4}} \tag{7}$$

where $A = (\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})$, $B = \frac{4}{3}(l-1)\mathbf{G}\sqrt{\frac{2M}{\sigma}A}$, $C = -(M-1)(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})$ *for some constant* $M > 1$*, and* σ *denotes the convexity parameter of* ψ*.*

Proof. See Appendix in (Zweig & Weinshall, 2012). □

4. Experiments

Comparison Methods. We compare our algorithms to three baseline methods, representing three common optimization approaches: 'NoReg' - where learning is done simply by minimizing the loss function without regularization. 'L12' - a common approach to multi-task learning where in addition to minimizing the loss function we also regularize for group sparseness (enforcing feature sharing) using the l_1/l_2 norm. 'L1'- a very common regularization approach where the loss function is regularized in order to induce sparsity by using the l_1 norm. All methods are optimized using the same algorithms described above, where for the non-hierarchical methods we set $L = 1$, for 'NoReg' we set $\phi = 0$, for 'L12' we set $\lambda = 0$, and for 'L1' we set $\lambda = 1$. The parameter ϕ for 'L12' and 'L1' is also chosen using cross validation.

We also use for comparison three recent approaches which exploit relatedness at multiple levels. (i) The single stage approach of (Kang et al., 2011) which simultaneously finds the grouping of tasks and learns the tasks. (ii) The tree-guided algorithm of (Kim & Xing, 2010) which can be viewed as a two stage approach, where the tasks are learnt after a hierarchical grouping of tasks is either discovered or provided. We applied the tree-guided algorithm in three conditions:

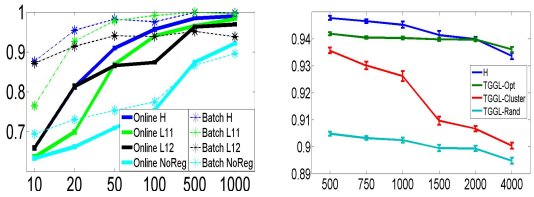


Figure 1. Synthetic data results. The 'Y'-axis measures the average accuracy over all tasks, where accuracy results correspond to 10 repetitions of the experiment. Left: performance as a function of sample size ('X'-axis). We show comparisons of both the batch and online algorithms, where 'H' denotes our hierarchical Algorithm with 5 levels, and 'L11' ,'L12' and 'NoReg' - the different baseline methods. Right: Performance as a function of the number of random features. We show comparison to the tree-guided group lasso algorithm based on the true hierarchy 'TGGL-Opt', clustered hierarchy 'TGGL-Cluster' and random hierarchy 'TGGL-Rand'.

when the true hierarchy is known, denoted 'TGGL-Opt'; when it is discovered by agglomerative clustering (as suggested in (Kim & Xing, 2010)), denoted 'TGGL-Cluster'; or when randomly chosen (random permutation of the leafs of a binary tree), denoted 'TGGL-Rand'. (iii) The method described in (Zhao et al., 2011) which is based on the work of (Kim & Xing, 2010) and extends it to a large scale setting where a hierarchy of classes is assumed to be known.

4.1. Synthetic data

In order to understand when our proposed method is likely to achieve improved performance, we tested it in a controlled manner on a synthetic dataset we had created. This synthetic dataset defines a group of tasks related in a hierarchical manner: the features correspond to nodes in a tree-like structure, and the number of tasks sharing each feature decreases with the distance of the feature node from the tree root. We tested to see if our approach is able to discover (implicitly) and exploit the hidden structure thus defined. The inputs to the algorithm are the sets of examples and labels from all k tasks $\{\mathbf{S}^i\}_{i=1}^k$, without any knowledge of the underlying structure.

Classification performance: We start by showing in Fig. 1-left that our hierarchical algorithm achieves the highest accuracy results, both for the batch and online settings. For the smallest sample size the 'L12' baseline achieves similar performance, while for the largest sample size the 'L1' baseline closes the gap indicating that given enough examples, sharing of information between classes becomes less important. We also see that the online Algorithm 2 converges to the performance of the batch algorithm after seeing enough examples.

The advantage of the hierarchical method is not due simply to the fact that it employs a combination of classifiers, but rather that it clearly benefits from sharing information. Specifically, when the cascade was applied to each task separately, it achieved only 93.21% accuracy as compared to 95.4% accuracy when applied to all the 100 tasks jointly.

To evaluate our implicit approach we compared it to the Tree-Guided Group Lasso (Kim & Xing, 2010) (TGGL) where the hierarchy is assumed to be known - either provided by a supervisor (the *true* hierarchy), clustered at pre-processing, or randomly chosen. Fig. 1-right shows results for 100 tasks and 20 positive examples each. We challenged the discovery of task relatedness structure by adding to the original feature representation a varying number (500-4000) of irrelevant features, where each irrelevant feature takes the value '-1' or '1' randomly. Our method performs much better than TGGL with random hierarchy or clustering-based hierarchy. Interestingly, this advantage is maintained even when TGGL gets to use the true hierarchy, with up to 1500 irrelevant features, possibly due to other beneficial features of the cascade.

4.2. Real data

Small Scale (Kang et al., 2011) describe a multi-task experimental setting using two digit recognition datasets MNIST (LeCun et al., 1998) and USPS (Hull, 1994), which are small datasets with only 10 classes/digits. For comparison with (Kang et al., 2011), we ran our method on these datasets using the same representations, and fixing $L = 3$ for both datasets. Table 1 shows all results, demonstrating clear advantage to our method. The results of our basic baseline methods 'NoReg' and 'L1' achieve similar or worse results,[2] comparable to the single task baseline approach presented in (Kang et al., 2011). Thus, the advantage of the cascade 'H' does not stem from the different optimization procedures, but rather reflects the different approach to sharing.

Table 1. Error rates on digit datasets

	USPS	MNIST
H	6.8% ± 0.2	13.4% ± 0.5
Kang et al.	8.4% ± 0.3	15.2% ± 0.3

Medium Scale We tested our batch approach with four medium sized data sets: Cifar100 (Krizhevsky & Hinton, 2009), Caltech101, Caltech256 (Griffin et al., 2007) and MIT-Indoor Scene dataset (Quattoni & Torralba, 2009) with 100, 102, 257 and 67 categories in each dataset respectively. We tested both the multi-class and multi-task settings. For the multi-task setting we consider the 1-vs-all tasks. For Cifar-100 we fixed $L = 5$ for the number of levels in the cascade, and for the larger datasets of Caltech101/256 and Indoor-Scene we used $L = 4$.

We also investigated a variety of features: for the Cifar-100 we used the global Gist representation embedded in an approximation of the RBF feature space using random projections as suggested by (Rahimi & Recht, 2007), resulting in a 768 feature vector. For the Caltech101/256 we used the output of the first stage of Gehler et al's kernel combination approach (Gehler & Nowozin, 2009) (which achieves state of the art results on Caltech101/256) as the set of features. For the MIT-Indoor Scene dataset we used *Object Bank* features (Li et al., 2010), which achieves state-of-the-art results on this and other datasets.

We tested the Cifar-100 dataset in the multi-task and multi-class settings. We used 410 images from each class as the training set, 30 images as a validation set, and 60 images as the test set. For the multi-task setting we considered the 100 1-vs-rest classification tasks. For each binary task, we used all images in the train set of each class as the positive set, and 5 examples from each class in the 'rest' set of classes as the negative set. The experiment was repeated 10 times using different random splits of the data. Results are shown in Table 2, showing similar performance for the cascade and the competing Tree-Guided Group Lasso method.

Table 2. Cifar-100: accuracy results. 'Baselines' denotes - 'L1', 'L12' and 'NoReg' which showed similar performance.

	multi-class	1-vs-rest
H	21.93 ± 0.38	79.91 ± 0.22
Baselines	18.23 ± 0.28	76.98 ± 0.17
TGGL-Cluster	-	79.97 ± 0.10
TGGL-Rand	-	80.25 ± 0.10

In our experiments with the MIT-Indoor Scene dataset we used 20, 50 and 80 images per scene category as a training set, and 80, 50 and 20 images per category as test set respectively. We repeated the experiment 10 times for random splits of the data, including the single predefined data split provided by (Quattoni & Torralba, 2009) as a benchmark. Fig. 2-left shows the classification results of the cascade, which significantly outperformed the baseline methods and the previously reported state of the art result of (Li et al., 2010) on the original data split of (Quattoni & Torralba, 2009), using the exact same feature representation. We also significantly outperformed 'TGGL-

[2] 'NoReg' - 9.5% ± 0.2 and 17% ± 0.5 for USPS and MNIST respectively; 'L1' - 8.8% ± 0.5 and 16% ± 0.8 for USPS and MNIST respectively.

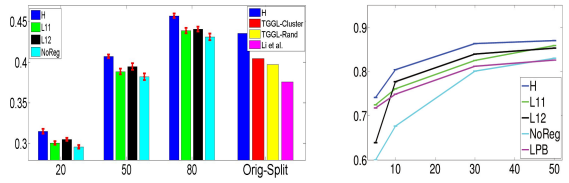


Figure 2. Real data results. 'Y'-axis measures the average accuracy over all tasks. Left, Multiclass accuracy results on the MIT-Indoor-Scene dataset, for 4 experimental conditions: 20, 50, and 80 images used for training respectively, and 'OrigSplit' - the single predefined split of (Quattoni & Torralba, 2009). Right, Multi-task 1-vs-rest results for the Caltehc256 dataset, where 'LPB' denotes our implementation of the binary version of the approach presented in (Gehler & Nowozin, 2009) (see text). The 'X'-axis varies with sample size.

Cluster' and 'TGGL-Rand'.

With Caltech101 and Caltech256 we used the data provided by (Gehler & Nowozin, 2009) for comparisons in both their original multi-class scenario and a new multi-task scenario. We tested our approach using the exact same experimental setting of (Gehler & Nowozin, 2009) given the scripts and data provided by the authors. In the original multi-class setting addressed in (Gehler & Nowozin, 2009) our results compare to their state-of-the-art results both for 30 training images (78.1%) in the Caltech101 and for 50 images (50%) in the Caltech256.

In the multi-task scenario we trained a single 1-vs-rest classifier for each class. In addition to our regular baseline comparisons we implemented a variant of the ν-LPB method, which was used in (Gehler & Nowozin, 2009) as the basis to their multi-class approach.

Fig. 2-right, shows results for the multi-task setting on the Caltech256. First we note that our algorithm outperforms all other methods including ν-LPB. We also note that given this dataset all regularization approaches exceed the NoReg baseline, indicating that this data is sparse and benefits from information sharing. (Results for the Caltech101 are similar and have therefore been omitted.)

Large Scale We demonstrate the ability of our online method to scale up to large datasets with many labels and many examples per each label by testing it on the ILSVRC(2010) challenge (Berg et al., 2010). This is a large scale visual object recognition challenge, with 1000 categories and 668-3047 examples per category. With so many categories the usual l_1/l_2 regularization is expected to be too crude, identifying only a few shared features among such a big group of diverse classes. On the other hand, we expect our hierarchical method to capture varying levels of useful information to share.

We compared our method to the hierarchical scheme of (Zhao et al., 2011) (using their exact same feature representation). Rather than compute the hierarchy from the data, their method takes advantage of a known semantic word-net hierarchy, which is used to define a hierarchical group-lasso regularization and calculate a similarity matrix augmented into the loss function. The comparison was done on the single split of the data provided by the challenge (Berg et al., 2010) .

Table 3. ILSVRC(2010): Classification accuracy of the best of N decisions, Top N.

	Top 1	Top 2	Top 3	Top 4	Top 5
Alg 2	0.285	0.361	0.403	0.434	0.456
Zhao	0.221	0.302	0.366	0.411	0.435

Table 3 shows our performance as compared to that of (Zhao et al., 2011). We show accuracy rates when considering the 1-5 top classified labels. In all settings we achieve significantly better performance using the exact same image representation and much less labeled information.

4.3. Discussion

For small and medium scale datasets we see that our cascade approach and the batch Algorithm 1 outperform the baseline methods significantly. For the large scale dataset the online Algorithm 2 significantly outperforms all other baseline methods. It is interesting to note that even when the alternative baseline methods perform poorly, implying that the regularization functions are not beneficial on their own, combining them as we do in our hierarchical approach improves performance. This can be explained by the fact that a linear combination of classifiers is known to improve performance if the classifiers are accurate and diverse.

When comparing to the recent related work of (Kim & Xing, 2010) denoted TGGL, we see that our implicit approach performs significantly better with the synthetic and MIT-Indoor scene datasets, while on the Cifar dataset we obtain similar results. With the synthetic dataset we saw that as the clustering of the task hierarchy becomes more challenging, TGGL with clustering degrades quickly while the performance of our method degrades more gracefully. We expect this to be the case in many real world problems where the underlining hierarchy is not known in advance. Furthermore, we note that with the small scale digit datasets our approach outperformed significantly the reported

results of (Kang et al., 2011). Finally, we note that our approach can be used in a pure online setting while these two alternative methods cannot.

We also compared with a third method (Zhao et al., 2011) using the ILSVRC(2010) challenge (Berg et al., 2010); this is a challenging large scale visual categorization task, whose size - both the number of categories and the number of examples per category, provides the challenges particularly suitable for our approach. The online algorithm makes it possible to scale up to such a big dataset, while the hierarchical sharing is important with possibly many relevant levels of sharing between the tasks. Particularly encouraging is the improvement in performance when compared to the aforementioned work where an explicit hierarchy was provided to the algorithm.

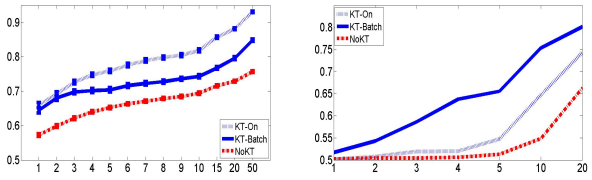
5. Knowledge-Transfer

We now briefly outline two natural extensions of our multi-task learning algorithms to achieve knowledge-transfer to novel related tasks which arrive with too few training examples. The transfer of information is based on the cascade of matrices $\{\mathbf{W}^l\}_{l=1}^L$ learnt during pre-training. The extension of the batch algorithm is based on dimensionality reduction of pre-trained models. The extension of the online method maintains the same regularization structure and parameters of the online multi-task setting.

Batch Method The method is based on projecting the data into the subspaces defined by the learnt models $\{\mathbf{W}^l\}_{l=1}^L$. Consequently the new task is represented in L sub-spaces that capture the structure of the shared information between the previous k tasks. Specifically, we define each projection matrix $\mathbf{P}^l$ by the first z columns of the orthonormal matrix $\mathbf{U}^l$, where $svd(\mathbf{W}^l) = \mathbf{U}^l\Sigma\mathbf{V}^l$. At each level l we project the new data-points onto $\mathbf{P}^l$. Thus the modeling of the new task involves $z * L$ parameters, wherein the l'th level data is projected onto the unique subspace characteristic of level l. In order to pass the parameters to the next level we project back to the original feature space.

Online Method We assume that a related set of tasks has been learnt using the online Algorithm 2. In order to initialize the online learning of a new single task or group of tasks, we concatenate the parameters of the new tasks to the previously learnt parameters, thus influencing the structure of the newly learnt task parameters.

The batch method is particularly useful when the data lies in a high dimensional feature space, and the number of examples from the novel task is too small to learn effectively in such a space. The online approach is particularly useful for bootstrapping the online learning of novel classes, achieving higher performance at the early stages of the learning as compared to a non transfer approach. Results are shown in Fig. 3.



(a) (b)

Figure 3. Experimental results. The 'Y'-axis corresponds to average accuracy over all tasks, and the 'X'-axis to the sample size. *KT-On* denotes the online knowledge-transfer method. *KT-Batch* denotes the batch knowledge-transfer method. *NoKT* denotes the no knowledge-transfer control: the multi-task batch algorithm with $L = 1$ and $\phi = 0$. (a) Results with the synthetic data described above. 99 tasks were used as the pre-trained tasks and a single task as the unknown novel task. The experiment was repeated 100 times, each repetition choosing a different task as the novel task. (b) Results for the large size ILSVRC2010 experiment. 1000 1-vs-rest tasks were considered, each task separating a single class from the rest. 900 tasks where chosen as the known and 100 as the unknown.

6. Summary

We presented a cascade of regularized optimization problems designed to induce implicit hierarchical sharing of information in multi-task, multi-class and knowledge-transfer settings. We described efficient batch and online learning algorithms implementing the cascade. For the online algorithm we provided regret analysis from which it follows that the average regret of our learning method converges. The method was tested on synthetic data and seven real datasets, showing significant advantage over baseline methods, and similar or improved performance as compared to alternative state of the art methods.

Acknowledgements

We would like to thank Bin Zhao and Prof. Eric Xing for providing the code for representing the data in our comparison to their work (Zhao et al., 2011), and Uri Shalit for helpful discussions and literature pointers. This work was supported in part by grants from the Israel Science Foundation and the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

Amit, Y., Fink, M., Srebro, N., and Ullman, S. Uncovering shared structures in multiclass classification. *ICML*, 2007.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Bengio, S., Weston, J., and Grangier, D. Label embedding trees for large multi-class tasks. *NIPS*, 2010.

Berg, A., Deng, J., and Fei-Fei, L. Large scale visual recognition challenge 2010, 2010. URL http://www.image-net.org/challenges/LSVRC/2010/index.

Caruana, R. Multitask learning. *Machine Learning*, 1997.

Crammer, K. and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2002.

Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

Duchi, J. and Singer, Y. Boosting with structural sparsity. In *ICML*, pp. 297–304. ACM, 2009.

Friedman, J., Hastie, T., and Tibshirani, R. A note on the group lasso and a sparse group lasso. *Arxiv preprint arXiv:1001.0736*, 2010.

Gao, T. and Koller, D. Discriminative learning of relaxed hierarchy for large-scale visual recognition. *ICCV*, 2011.

Gehler, P. and Nowozin, S. On feature combination for multiclass object classification. In *ICCV*, 2009.

Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. 2007.

Hull, J.J. A database for handwritten text recognition research. *T-PAMI, IEEE*, 16(5):550–554, 1994.

Hwang, S.J., Grauman, K., and Sha, F. Learning a tree of metrics with disjoint visual features. *NIPS*, 2011.

Jawanpuria, P. and Nath, J.S. A convex feature learning formulation for latent task structure discovery. *ICML*, 2012.

Kang, Z., Grauman, K., and Sha, F. Learning with whom to share in multi-task feature learning. *NIPS*, 2011.

Kim, S. and Xing, E.P. Tree-guided group lasso for multi-task regression with structured sparsity. *ICML*, 2010.

Krizhevsky, A. and Hinton, GE. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

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

Li, L.J., Su, H., Xing, E.P., and Fei-Fei, L. Object bank: A high-level image representation for scene classification and semantic feature sparsification. *NIPS*, 2010.

Obozinski, G., Taskar, B., and Jordan, M. Joint covariate selection for grouped classification. *Department of Statistics, U. of California, Berkeley, TR*, 743, 2007.

Quattoni, A. and Torralba, A. Recognizing indoor scenes. *CVPR*, 2009.

Quattoni, A., Collins, M., and Darrell, T. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *NIPS*, 2007.

Shalev-Shwartz, S., Wexler, Y., and Shashua, A. Shareboost: Efficient multiclass learning with feature sharing. *Proc. NIPS*, 2011.

Sprechmann, P., Ramírez, I., Sapiro, G., and Eldar, Y.C. C-hilasso: A collaborative hierarchical sparse modeling framework. *Signal Processing, IEEE Transactions on*, 59(9):4183–4198, 2011.

Tibshirani, R. Regression shrinkage and selection via the lasso. *J. Royal Statist. Soc.. B*, 58:267–288, 1996.

Torralba, A., Murphy, K.P., and Freeman, W.T. Sharing visual features for multiclass and multiview object detection. *T-PAMI, IEEE*, 29(5):854–869, 2007.

Verma, N., Mahajan, D., Sellamanickam, S., and Nair, V. Learning hierarchical similarity metrics. In *CVPR*. IEEE, 2012.

Weinshall, D., Hermansky, H., Zweig, A., Luo, J., Jimison, H., Ohl, F., and Pavel, M. Beyond novelty detection: Incongruent events, when general and specific classifiers disagree. In *Proc. NIPS*, volume 8, 2008.

Wright, S.J., Nowak, R.D., and Figueiredo, M.A.T. Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on*, 57(7):2479–2493, 2009.

Xiao, L. Dual averaging methods for regularized stochastic learning and online optimization. *JMLR*, 2010.

Yang, H., Xu, Z., King, I., and Lyu, M. Online learning for group lasso. *ICML*, 2010.

Yang, Jian-Bo and Tsang, Ivor. Hierarchical maximum margin learning for multi-class classification. *UAI*, 2011.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *J. Royal Statist. Soc.. B*, 68(1):49–67, 2006.

Zhao, B., Fei-Fei, L., and Xing, E.P. Large-scale category structure aware image categorization. *Proc. NIPS*, 2011.

Zhou, D., Xiao, L., and Wu, M. Hierarchical classification via orthogonal transfer. In *ICML*, 2011.

Zweig, A. and Weinshall, D. Exploiting Object Hierarchy: Combining Models from Different Category Levels. *Proc. ICCV*, 2007.

Zweig, A and Weinshall, D. Hierarchical regularization cascade. *TR 2012-9 Leibniz Center, HUJI*, 2012.

Appendix: Supplementary material

A. Proof of Thorem 1

Proof. We assume in the statement of the theorem that

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \geq -(M-1)(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})\sqrt{T} \qquad (8)$$

for some constant $M > 1$. This assumption is justified if the accumulated cost obtained by the online algorithm is larger than the cost obtained by the optimal batch algorithm for most of the training examples. We argue that if this assumption is violated then the online algorithm is revealed to be a very good performer, and it therefore makes little sense to judge its performance by the deviation from another algorithm (the "optimal" batch algorithm) whose performance is worse for a significant fraction of the training sample.

Using the definition of $\mathbf{L}_{t,\mathbf{W}_*^{l-1}}$ and $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$, we rewrite the regret (6) with a single loss function

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) = \sum_{t=1}^{T}(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) + \psi^l(\hat{\mathbf{W}}_t^l)) - [\sum_{t=1}^{T}(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_*^l + \mathbf{W}_*^{l-1} - \mathbf{W}_t^{l-1}) + \psi^l(\hat{\mathbf{W}}_*^l))]$$

From the convexity of $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$ it follows that

$$\sum_{t=1}^{T}(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) - \mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_*^l + \mathbf{W}_*^{l-1} - \mathbf{W}_t^{l-1})) \leq \sum_{t=1}^{T} < U_t^l, \hat{\mathbf{W}}_t^l - \hat{\mathbf{W}}_*^l - \mathbf{W}_*^{l-1} + \mathbf{W}_t^{l-1} >$$

Under the conditions of the theorem, it is shown in (corolllary 2a (Xiao, 2010)) that

$$\sum_{t=1}^{T} < U_t, \mathbf{W}_t - \mathbf{W}_* > +\psi(\mathbf{W}_t) - \psi(\mathbf{W}_*)) \leq \Delta_T$$

where $\Delta_T = (\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})\sqrt{T}$. Using this result and the sublinearity of the inner product, we get

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + \sum_{t=1}^{T} < U_t^l, \mathbf{W}_t^{l-1} - \mathbf{W}_*^{l-1} > \qquad (9)$$

From the definition of $\mathbf{W}_t^{l-1} = \sum_{k=1}^{l-1}\hat{\mathbf{W}}_t^k$ and $\mathbf{W}_*^{l-1} = \sum_{k=1}^{l-1}\hat{\mathbf{W}}_*^k$ and the Cauchy-Schwarz inequality we get

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + \sum_{k=1}^{l-1}\sum_{t=1}^{T} < U_t^l, \hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k > \leq \Delta_T + \sum_{k=1}^{l-1}\sum_{t=1}^{T} ||U_t^l||||\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k|| \qquad (10)$$

We use the bound on $||\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k||$ from (theorem 1b (Xiao, 2010)) and assumption (8) to obtain

$$\sum_{t=1}^{T}||\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k|| \leq \sum_{t=1}^{T}\sqrt{\frac{2M\Delta_t}{\sigma t + \gamma\sqrt{t}}} \leq Q\sum_{t=1}^{T} t^{-\frac{1}{4}} \leq Q\frac{4}{3}(T+1)^{\frac{3}{4}}$$

where $Q = \sqrt{\frac{2M}{\sigma}(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})}$. Inserting this last inequality into (10), and since $\forall \mathbf{t}, l\ ||\mathbf{U}_t^l|| < \mathbf{G}$, we obtain

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + (l-1)\mathbf{G}Q\frac{4}{3}(T+1)^{\frac{3}{4}} \qquad (11)$$

from which (7) immediately follows. □

B. Synthetic Data Experiments

The synthetic data is created in the following manner: we define a binary tree with k leaves. Each leaf in the tree represents a single binary classification task. Each node in the tree corresponds to a single binary feature $f \in \{-1, 1\}$. For a single task we divide the features into two groups: task-specific and task-irrelevant. Task-irrelevant features have equal probability of having the value 1 or -1 for all examples in the task. Task-specific features are assigned the value 1 for all positive examples of the task. All negative examples of the task must have at least one feature from the task-specific feature set with a value of -1.

The task-specific feature set is the set of features corresponding to all nodes on the path from the leaf to the root, while all other nodes define the task-irrelevant feature set. For each group of tasks, their shared features are those corresponding to common ancestors of the corresponding leaves. An illustration of the binary tree and a an example of a sample set of a specific task is given in Fig. 4.

Structure discovery: Feature weights learnt at different learning steps of the cascade for an experiment with 100 synthetic tasks, 199 features and 20 positive and negative samples per task, are shown in Fig 5. As can be seen, the first learning stages, $l = 1$ and $l = 2$ capture shared information, while the last stages,

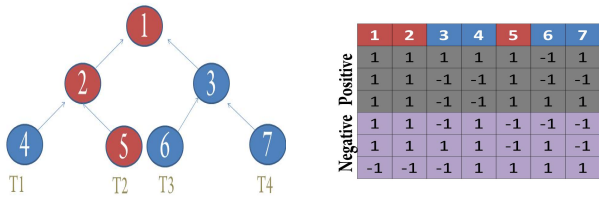


Figure 4. Synthetic data illustration. Left graph shows a tree of features corresponding to four tasks. The task specific features of task 2, 'T2', are highlighted in red. The task-irrelevant features are marked blue. The table on the right shows an example of a sample set sampled for task 'T2' given the task tree to the left. Each row denotes a sample. Each column denotes a feature. 'Positive' and 'Negative' denote the positive and negative sample sets, respectively.

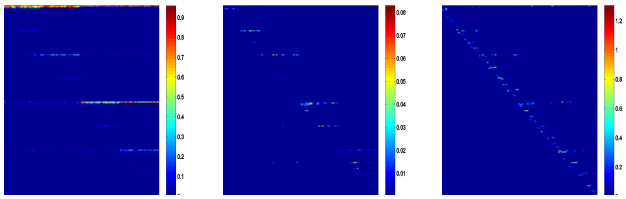


Figure 5. Synthetic experiment learnt parameters $\mathbf{W}^l$. Each plot corresponds to a single matrix learnt at stage l of the cascade. The rows correspond to features and each column corresponds to a single task.

e.g. stage $l = 4$ capture task specific features. The hierarchical shared structure of features is discovered in the sense that higher levels in the cascade share the same set of features and as the cascade progresses the chosen features are shared by less tasks. The pattern of learnt feature weights fits the synthetic data pattern of feature generation.

Parameter Robustness Fig. 6 shows the robustness of the hierarchical approach with respect to the regularization parameter ϕ. For all three regularizing approaches we varied the value of ϕ in the range [0.01-2] with 0.01 jumps. For the hierarchal approach we set $\phi^1 = \phi$ and $\phi^l = \frac{\phi^{l-1}}{2}$ for all $l \in [2..L]$.

We also found the method to be quite robust to the parameter L determining the number of levels in the cascade. Varying the value of L between 3 to 7 on the synthetic data with 100 tasks gave close results in the range 94.5% to 95.5%.

Single Level Comparison Fig. 7 we show a comparison of the online cascade to a group of single level

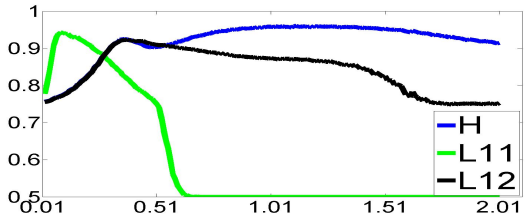


Figure 6. Performance as a function of the regularization parameter ϕ. Synthetic data with 100 examples per task. The 'Y'-axis corresponds the average accuracy of all tasks on 10 repetitions of the experiment. The 'X'-axis corresponds to the value of ϕ. Note that the max values of each method are: 96.02, 94.23 and 92.30 for 'H', 'L11' and 'L12' respectively.

Table 4. Varying the number of levels L

L = 2	L = 3	L = 4	L = 5	L = 6	L = 7	L = 8
92.42	95.47	95.73	94.74	95.21	94.48	93.52

regularization schemes, using the same set of λ values we used in the cascade. Clearly no single-level regularization achieves as good performance as the hierarchical method.

Adding Tasks We examined the effect of the number of tasks in the multitask setting. Ideally adding more tasks should never reduce performance, while in most cases leading to improved performance. We tested two scenarios - adding tasks which are similarly related to the existing group of tasks Fig. 8a, and adding tasks which are loosely related to all other tasks but strongly related among themselves Fig. 8b.

With additional tasks of the same degree of relatedness, we increase the amount of information available for sharing. As expected, we see in Fig. 8a that performance improves with increasing number of tasks, both for the hierarchical algorithm and for 'L12Reg'. 'L1Reg' is not intended for information sharing, and

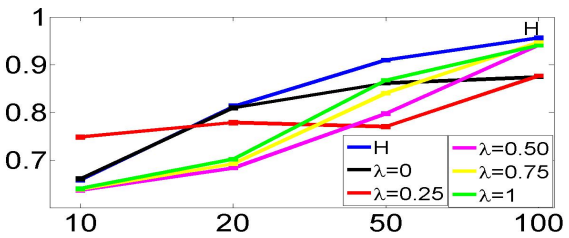


Figure 7. A comparison of our online cascade approach 'H' to variants corresponding to intermediate levels in the cascade, defined by the value λ. 'Y'-axis measures the average accuracy over all tasks, and the 'X'-axis the sample size.

therefore it is not affected by increasing the number of tasks. When adding loosely related tasks, we see in Fig. 8b that the performance of the hierarchical algorithm increases as we add more tasks; for the 'L12Reg' method, on the other hand, we see a significant drop in performance. This is because in this case the overall relatedness among all tasks decreases as additional tasks are added; the 'L12Reg' method still tries to share information among all tasks, and its performance therefore decreases.

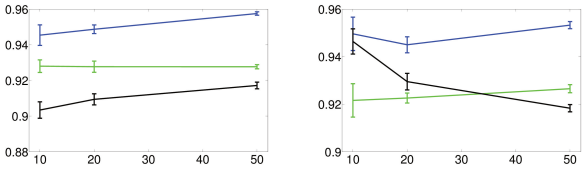


Figure 8. Adding tasks results. Plots correspond to the average 1-vs-rest accuracy as a function the number of tasks, when (a) adding similarly related tasks and (b) adding loosely related tasks. Blue denotes our hierarchical Algorithm 1, green the 'L1Reg' baseline and black the 'L12Reg' baseline method.

Data Rotation Next, we wish to isolate the two factors of sparsity and shared information. The synthetic data was constructed so that there is an increased level of shared information between classes as a function of the distance between their respective leaves in the defining tree of features. The shared features are also sparse. In order to maintain the shared information and eliminate sparseness, we rotate the vector of features; when the rotation is applied to more features, the amount of sparseness decreases respectively.

Table 5 shows the comparative results both for the case when all features are rotated and when only half are rotated (in which case the features being rotated are chosen randomly). As expected, the regularization-free method - 'NoReg' - is not effected by any of this. The performance of 'L1Reg', which assumes sparsity, drops as expected, reaching baseline with full rotation, presumably because during cross-validation a very low value for the regularization parameter is chosen. The two methods which exploit shared information, our hierarchical algorithm and the 'L12Reg' baseline method, perform better than baseline even with no sparseness (full rotation), showing the advantage of being able to share information.

Table 5. Performance comparison for the different methods applied to the Synthetic data. 'T 100 S 20' denotes the multi-task setting with 100 tasks and 20 samples each, 'half rotated' - the same setting as 'T 100 S 20' with a random rotation of half of the features, and 'full rotation' - a random rotation of all the features.

	T 100 S 20	half rotation	full rotation
H	95.40 ± 0.17	90.37 ± 0.61	78.49 ± 0.16
L1Reg	92.54 ± 0.17	86.70 ± 0.59	73.01 ± 0.09
L12Reg	91.49 ± 0.2	85.56 ± 0.62	78.49 ± 0.16
NoReg	72.88 ± 0.19	72.81 ± 0.12	73.03 ± 0.10

C. Real Data

C.1. ILSVRC2010 Baseline Comparison

In Fig 9 we show the error rates for the Top-1 scenario, considering a single classification. We show results when training using all the data Fig 9-left, and when using only 100 examples per each task Fig 9-right. The results are shown as a function of the number of repetitions of the algorithm over all the data.

At convergence we see an improvement of 1.67% in accuracy when using the cascade with 7 levels, 28.96% compared to 27.29%. (Zhao et al., 2011) obtained an improvement of 1.8% in accuracy when comparing their approach to their own baseline, 22.1% vs. 20.3%. We obtained a similar rate of improvement using much less information (not knowing the hierarchy) for a higher range of accuracies.

We note that our baseline approaches converge after 20 repetitions when using all the data, (for clarity we show only up to 15 repetitions in the left plot of Fig 9). This effectively means the same runtime, as the cascade runtime is linear in the number of levels where each level has the same complexity of the baseline approaches. On the other hand the online cascade algorithm 2 can be trivially parallelized where as the repetitions over the data for a single baseline cannot. Thus, in a parallel setting the gain in runtime would be linear in the number of levels of the cascade. A trivial parallelization can be implemented by running each level of the online cascade on a time stamp shifted by l thus the first level of the cascade will see at time t the t sample while level l will see sample $t-l$.

D. Knowledge Transfer

Batch Method The batch knowledge-transfer method is described below in Algorithm 3. The projection matrix $\mathbf{P}^l$ is defined by the first z columns of the orthonormal matrix $\mathbf{U}^l$, where $svd(\mathbf{W}^l) = \mathbf{U}^l \Sigma \mathbf{V}^l$.

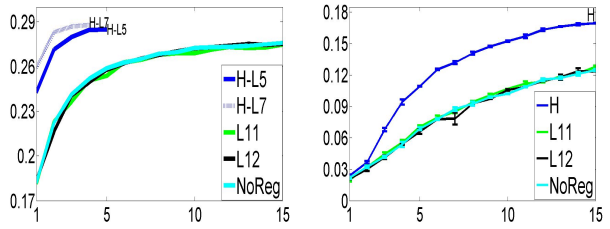


Figure 9. Real data, showing performance of Top-1 classification on the ILSVRC(2010) challenge (Berg et al., 2010) using all examples (left plot) or only 100 examples per each category (right plot). Here the 'X'-axis corresponds to repetitions over the training data. In the left plot 'H-L5' and 'H-L7' denote our hierarchical algorithm with 5 and 7 levels respectively. In the right plot 'H' corresponds to 5 levels in our hierarchical algorithm. The error bars in correspond to the standard error given 3 different choices of 100 Examples.

Online Method The online knowledge-transfer algorithm is described below in Algorithm 4; it succeeds $\mathbf{t}_{old}$ iterations of Algorithm 2. The input to the algorithm is the same set of parameters used for Algorithm 2 and its intermediate calculations - the cascade $\{\mathbf{W}^l_{old}\}_{l=1}^L$ and the set of final average subgradients $\{\bar{\mathbf{U}}^l_{old}\}_{l=1}^L$. These are used to approximate future subgradients of the already learnt tasks, since Algorithm 4 receives no additional data-points for these tasks. The parameters of Algorithm 2 are used because cross-validation for parameter estimation is not possible with small sample.

Below we denote the vector corresponding to the mean value of each feature as $mean(\mathbf{W}^l_{old})$. We denote the concatenation of columns by $\circ$. In order to account for the difference between the old time step $\mathbf{t}_{old}$ to the new time step $\mathbf{t}$ we consider $h(\mathbf{W})$ to be the squared l_2 norm applied to each column separately. We calculate the inner product of the resulting vector with $\frac{\gamma}{\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}}$ in step 1(a).(iv); $\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}$ denotes a vector derived by the concatenation of $\mathbf{k}$ times $\mathbf{t}_{old}$ with $\mathbf{t}$.

D.1. Experiments

In this section we evaluate our algorithms for knowledge transfer in small sample scenarios. We start by comparing the different methods on controlled synthetic data in Section D.1.1. We then test the performance of our method using several real data datasets employing different image representation schemes in two different settings: *medium size*, with several tens of classes and a dimensionality of 1000 features as image representation in Section D.1.2; and *large size*, with hundreds of classes and an image representation of 21000 features in Section D.1.3. We also compared the performance in a '1-vs-rest' setting and in a setting with a common negative class (as in clutter).

Algorithm 3 Knowledge-Transfer with shared features projections

Input :

L number of levels

$\{\mathbf{P}^l\}_{l=1}^L$ Set of projections matrices learnt from the k pre-trained tasks

Output :

$\mathbf{W}$

1. $\mathbf{W}^0 = 0$
2. for $l = 1$ to L
   (a) Projection:
      i. $\hat{\mathbf{x}} = \mathbf{P}^{l^t} * \mathbf{x}$, $\forall i \in [1..\mathbf{k}]$ and $\forall \mathbf{x} \in \mathbf{S}^i$
      ii. $\hat{\mathbf{w}}^{l-1} = \mathbf{P}^{l^t} * \mathbf{w}^{l-1}$
   (b) $\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\{\hat{\mathbf{S}}^i\}_{i=1}^{\mathbf{k}}, \mathbf{W} + \hat{\mathbf{W}}^{l-1})$
   (c) Backprojection: $\mathbf{w} = \mathbf{P}^l * \hat{\mathbf{w}}$
   (d) $\mathbf{w}^l = \mathbf{w}^{l-1} + \mathbf{w}$
3. $\mathbf{w} = \mathbf{w}^L$

The methods used for comparison are the following:

Batch methods

- *KT-Batch-H*: corresponds to Algorithm 3 where knowledge-transfer is based on the projection matrices extracted from the batch cascade learning.
- *KT-Batch-NoReg*: here knowledge transfer corresponds to Algorithm 3 with $L = 1$ and $\phi = 0$, where information is transferred from the previously learnt models which were learnt in a single level with no regularization and no incentive to share information.

Online methods

- *KT-On-H*: corresponds to Algorithm 4 where we transfer information given the full cascade of regularization functions.
- *KT-On-L12*: corresponds to Algorithm 4 with $L = 1$ and $\lambda = 0$, where a single level of informa-

Algorithm 4 Online Knowledge-Transfer learning cascade

Input :

L, $\{\lambda^l\}_{l=1}^L$, $\{\phi^l\}_{l=1}^L$, γ set of parameters as in Algorithm 2

$\{\bar{\mathbf{U}}_{old}^l\}_{l=1}^L$ the average subgradient of the last iteration of Algorithm 2

$\{\mathbf{W}_{old}^l\}_{l=1}^L$ the set of parameters learnt by Algorithm 2

$\mathbf{t}_{old}$ number of temporal iterations of Algorithm 2

Initialization:

$\hat{\mathbf{W}}_0^l = \mathbf{W}_{old}^l \circ mean(\mathbf{W}_{old}^l),\ \bar{\mathbf{U}}_0^l = \bar{\mathbf{U}}_{old}^l \circ 0\ \forall l \in \{1..L\}$

$\mathbf{W}_t^0 = 0\ \forall t$

1. for t = 1,2,3,... do
   (a) for $l = 1$ to L
      i. Given the function $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$, compute a subgradient $\mathbf{U}_{t,new}^l \in \partial \mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$
      ii. $\bar{\mathbf{U}}_{t,new}^l = \frac{t-1}{t}\bar{\mathbf{U}}_{t-1,new}^l + \frac{1}{t}\mathbf{U}_{t,new}^l$
      iii. $\bar{\mathbf{U}}_t^l = \bar{\mathbf{U}}_{old}^l \circ \bar{\mathbf{U}}_{t,new}^l$
      iv. $\hat{\mathbf{W}}_t^l = \underset{\mathbf{W}}{\operatorname{argmin}}\ \bar{\mathbf{U}}_t^l\mathbf{W} + \psi^l(\mathbf{W}) + < \frac{\gamma}{\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}}, h(\mathbf{W}) >$
      v. $\mathbf{W}_t^l = \mathbf{W}_{old}^l \circ (\mathbf{W}_{t,new}^{l-1} + \hat{\mathbf{W}}_{t,new}^l)$
   (b) $\mathbf{W}_t = \mathbf{W}_t^L$

tion transfer is based on models trained to share information between all tasks equally.

Baseline methods, without Knowledge Transfer:

- *NoKT-Batch*: corresponds to the multi-task batch Algorithm 1 with $L = 1$ and $\phi = 0$.
- *NoKT-On-NoReg*: corresponds to the multi-task online Algorithm 2 with $L = 1$ and $\phi = 0$.

D.1.1. Synthetic Data

To test Algorithms 3 and 4 we trained 99 tasks using the multi-task batch and online algorithms with only 99 tasks, keeping the remaining task aside as the unknown novel task. Each known task was trained with 50 examples, with 10 repetitions over the data for the online Algorithm 2. After this multi-task preprocessing had finished, we trained the left out task using Algorithms 3 and 4 with either 1-10, 20 or 50 examples (with 100 repetitions in the online Algorithm 4). This was done 100 times, leaving out in turn each of the original tasks. In the batch knowledge-transfer we chose the rank of each projection matrix to keep 99.9% of the variance in the data. In the hierarchical knowledge-transfer this resulted in approximately 10 dimensions at the top level of the cascade, and 90 dimensions at the lowest level of the cascade.

As can be seen in Fig. 10a, our Knowledge-Transfer methods based on shared multi-task models achieve the best performance as compared to the alternative methods. The online knowledge-transfer method achieves the best results on this dataset. Note that with very small samples, the method which attempts to share information with all tasks equally - *KT-On-L12* - achieves the best performance. As the sample increases to 50, Algorithm 4 is able to perform better by exploiting the different levels of sharing given by the hierarchical approach *KT-On-H*. For both online Knowledge-Transfer options we see a significant improvement in performance as compared to the online with no knowledge transfer approach, *NoKT-On-NoReg*.

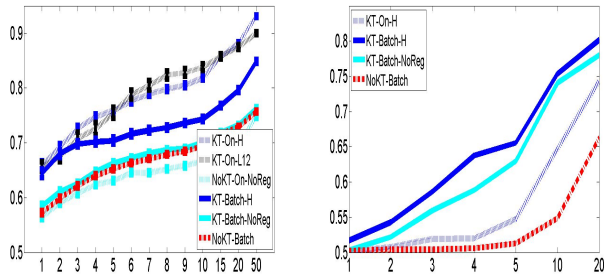
Looking at the batch method we see that Knowledge-Transfer based on sharing information between the original models, *KT-Batch-H*, outperforms significantly the knowledge-transfer based on no sharing of information in the original model training, *KT-Batch-NoReg*. The *KT-Batch-NoReg* actually performs no better than the no knowledge-transfer approach *NoKT-Batch*.

It is also interesting to note that the difference in average performance between the novel task to the pretrained tasks is less then 0.5% for 50 training examples when using the hierarchical knowledge-transfer. This indicates that in this experiment our hierarchical knowledge-transfer method reaches the potential of sharing as in the multi-task method, which outperforms all other methods on this synthetic data.

D.1.2. Medium size

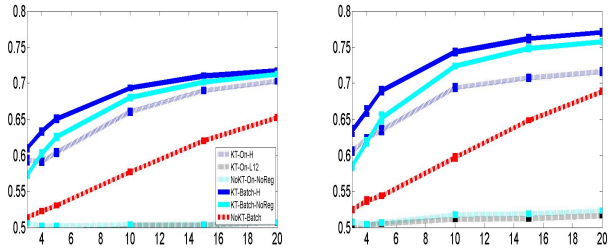
We tested our method with real data, starting with a moderate problem size and only 31 classes. For these experiments we used a subset of the ILSVRC(2010) challenge (Berg et al., 2010), which is an image dataset organized according to the WordNet hierarchy. From this huge dataset we chose 31 classes (synsets)[3] for the

[3] quail, partridge, hare, Angora rabbit, wood rabbit, indri, Madagascar cat, orangutan, chimpanzee, gorilla, fire engine, garbage truck, pickup truck, trailer truck, police wagon, recreational vehicle, half track, snowmobile, trac-



(a) (b)

Figure 10. Accuracy comparison. In all plots the 'Y'-axis corresponds to the average accuracy over all tasks, and the 'X'-axis to the sample size. (a) Results for Synthetic data experiment. (b) results for the large size imagenet experiment.



(a) (b)

Figure 11. Mid-size experiment accuracy results, with (a) common negative set, and (b) 1-vs-rest. In all plots the 'Y'-axis corresponds to the average accuracy of all tasks, and the 'X'-axis to the sample size.

set of pre-trained known classes with many training examples. This group of classes was chosen heuristically to contain varying levels of relatedness among classes, grouping together various terrestrial, aerial and sea vehicles, buildings, sea animals etc. For the novel classes with small sample we considered 30 randomly chosen classes from the remaining 969 classes in the dataset. The set of features used to describe images in this data set is based on the Sift features quantized into a codebook of 1000 words, which was tf-idf normalized.

We considered binary learning tasks where each chosen class, either pre-trained or novel, is contrasted with a set of images (negative examples) chosen from a group of different classes. The negative set was constructed in two ways: In the *1-vs-rest* condition the negative set of classes, the *Rest*, was defined as the group of 31 classes from which knowledge is transferred. In the second condition the negative set included 31 different classes sampled randomly from the original dataset excluding the small sample classes and the set of pre-trained classes. In this second condition all classes, both pre-trained and novel, had the exact same set of negative examples. This condition resembles previous experiments with knowledge-transfer (Zweig & Weinshall, 2007), where all tasks share the same negative set.

In both conditions the pre-trained models were trained using 480 positive examples and 480 negative examples. For the positive set of the small sample classes, we considered a sample size in the range 1-20. For the negative set we used all examples from each negative group (480 examples per class in the 1-vs-rest condition and 480 in total in the second condition). Examples were weighted according to sample size. For the pre-trained models we used a validation set of 60 examples per class; we used 100 examples per each class as its test set.

We considered each of the novel 30 classes separately. The experiment was repeated 8 times with different random splits of the data, for both the pre-trained and novel tasks. In the batch knowledge transfer methods we set the projection rank to maintain 99.9% of the variance in the original models learnt.

Results for the condition with shared negative set are shown in Fig. 11a. Results for the 1-vs-rest condition are shown in Fig. 11b. We see that knowledge-transfer methods achieve improved performance as compared to the alternative. In both conidtions the best performer is the hierarchical batch approach for Knowledge-transfer, *KT-Batch-H*. The poor performance of the *KT-Online-L12* can be explained by the fact that the regularization coefficient ϕ chosen by cross-validation during the multi-task pre-learning phase of the pre-trained models was chosen to be very low, indicating that a single level of sharing is not sufficient for this data.

D.1.3. Large size

As the ultimate knowledge transfer challenge, we tested our method with real data and large problem size. Thus we used all 1000 classes from the ILSVRC(2010) challenge (Berg et al., 2010). Each image was represented by a vector of 21000 dimensions, following the representation scheme used by (Zhao et al., 2011). 900 classes were chosen randomly as pre-trained classes, while the remaining 100 classes were used as the novel classes with small sample. We con-

tor, tricycle, fiddler crab, king crab, silver salmon, rainbow trout, striper, airliner, warplane, lifeboat, catamaran, boathouse and church building.

sidered *1-vs-rest* tasks as explained above. Pre-trained tasks were trained using 500 examples from the positive class and 500 examples chosen randomly from the remaining 899 classes. We used the online Algorithm 2 with 2 repetitions over the training data to train pre-trained tasks.

During test, the set of negative examples in the *1-vs-rest* condition was chosen randomly from all of the dataset, total of 999 classes. We used the labeled test set provided by (Berg et al., 2010). As small sample we considered 1-20 examples per class. Due to the original large image representation, in the batch knowledge-transfer methods we fixed the projection rank to maintain only 80% of the variance in the original models learnt.

We note that once the pre-trained models are computed, each step of training with the projected batch approach is faster than each step of the online approach as the online Algorithm 4 needs at each step to consider all the pre-trained parameters in order to compute the regularization value, while the batch Algorithm 3 considers these parameters only once during the projection phase. Using a big image representation as we do the online methods becomes computationally expensive if repeating the experiment for each of the novel small samples classes separately.

Results are shown in Fig. 10b. Clearly all methods inducing information sharing outperformed significantly the batch and online learning with no sharing. The *NoKT-on-NoReg* method performed poorly similarly to *NoKT-batch* and was omitted for brevity. *KT-on-L12* also preformed poorly due to the very low regularization parameter ϕ automatically chosen.