Data Perturbation for Escaping Local Maxima in Learning

Gal Elidan and **Matan Ninio** and **Nir Friedman**
Hebrew University
{*galel,ninio,nir*}*@cs.huji.ac.il*

Dale Schuurmans
University of Waterloo
dale@cs.uwaterloo.ca

Abstract

Almost all machine learning algorithms—be they for regression, classification or density estimation—seek hypotheses that optimize a score on training data. In most interesting cases, however, full global optimization is not feasible and local search techniques are used to discover reasonable solutions. Unfortunately, the quality of the local maxima reached depends on initialization and is often weaker than the global maximum. In this paper, we present a simple approach for combining global search with local optimization to discover improved hypotheses in general machine learning problems. The main idea is to escape local maxima by perturbing the *training data* to create plausible new ascent directions, rather than perturbing hypotheses directly. Specifically, we consider example-reweighting strategies that are reminiscent of boosting and other ensemble learning methods, but applied in a different way with a different goal: to produce a *single* hypothesis that achieves a good score on training and test data. To evaluate the performance of our algorithms we consider a number of problems in learning Bayesian networks from data, including discrete training problems (structure search), continuous training problems (parametric EM, non-linear logistic regression), and mixed training problems (Structural EM)—on both synthetic and real-world data. In each case, we obtain state of the art performance on both training and test data.

Introduction

Training algorithms in machine learning are almost always optimization algorithms: that is, they search for a hypothesis that maximizes a score on the training data. This is true for regression, classification and density estimation, as well as most other machine learning problems. The most common scores used in machine learning are *additive* on training data, which means that the score of a hypothesis h on data $D = \{\mathbf{x}[1], ..., \mathbf{x}[M]\}$ is a sum of local scores on each individual example, plus an optional regularization penalty

$$\text{Score}(h, D) \quad = \quad \sum_m \text{score}(h, \mathbf{x}[m]) - \text{penalty}(h)$$

Such scores arise naturally in regression or classification problems, where the local score is typically negated prediction error, and in density estimation, where the local score is typically log likelihood. Although we will apply our techniques to more general non-additive scores below, it will be useful to keep additive scores as a simple example.

Even for simple optimization objectives, in interesting hypothesis spaces like decision trees, neural networks, and graphical models, the problem of finding a globally optimal hypothesis is usually intractable. This is true whether one is searching for an optimal combination of hypothesis structure and parameters (e.g., decision tree learning), or just optimizing the parameters for a given structure (e.g., neural network training). Therefore, most training algorithms employ local search techniques such as gradient descent or discrete hill climbing to find locally optimal hypotheses (Bishop 1995). The drawback is that local maxima are common and local search often yields poor results.

A variety of techniques have been developed for escaping poor local maxima in general search, including random restarts, TABU search (Glover & Laguna 1993) and simulated annealing (Kirpatrick, Gelatt, & Vecchi 1994). However, these techniques do not exploit the particular nature of the training problem encountered in machine learning. Instead, they alter hypotheses in an oblivious fashion until this happens to provide an escape from a local basin of attraction.

In this paper, we consider strategies for escaping local maxima that perturb the *training data* instead of perturbing the hypotheses directly. In particular, we consider simple strategies for *reweighting* training examples to create useful ascent directions in hypothesis space. To do this we augment the score so that it considers a probability distribution w on the training examples, thus yielding

$$\text{Score}(h, D, \mathrm{w}) = \sum_m M \cdot \mathrm{w}_m \text{score}(h, \mathbf{x}[m]) - \text{penalty}(h)$$

An intuition for why example reweighting is effective for escaping local maxima is that it can cause "informed" changes to be made to the current hypothesis, rather than arbitrarily damage it: When a local maximum is reached, each training example contributes differently to the score. If the hypothesis is poor, then some training examples which contribute strongly to the score are likely to be outliers that should be down-weighted, whereas other examples that do not contribute strongly should be up-weighted to reflect their true importance in the underlying distribution. That is, a poor

procedure **PerturbedSearch**$(D, \mathrm{w}^0, h^0, \tau^0, \tau_{final})$
$t \leftarrow 0$
while $\tau^t > \tau_{final}$ do
$\mathrm{w}^{t+1} \leftarrow reweight(\text{Score}, \mathrm{w}^t, \tau^t, h^t, D)$
$h^{t+1} \leftarrow optimize(\text{Score}, \mathrm{w}^{t+1}, h^t, D)$
$\tau^{t+1} \leftarrow reduce(\tau^t, t)$
$t \leftarrow t + 1$
return h^t

Figure 1: Outline of the generic search procedure.

hypothesis will tend to fit outliers but under-represent examples that are actually important. Understanding how the score is influenced by training examples can therefore suggest plausible perturbations to the data which favor superior hypotheses.

Below, we consider two basic techniques for perturbing example weights to escape local maxima: *random reweighting*, which randomly samples weight profiles on the training data, and *adversarial reweighting*, which updates the weight profile to explicitly punish the current hypothesis, with the intent of moving the search quickly to a nearby basin of attraction. In both cases the weight profile is annealed toward uniform weights over time to ensure that the search eventually focuses on producing good solutions for the original distribution of training data.

Our basic approach has several benefits. First, these perturbation schemes are general and can be applied to a large variety of hypothesis spaces, either continuous or discrete. Second, our approach uses standard search procedures to find hypotheses, rather than employ the often wasteful "propose, evaluate, reject" cycle of simulated annealing approaches. Third, because a perturbation of the training data can generate a long chain of search steps in hypothesis space, a single reweighting step can result in a hypothesis that is very different from the one considered at the outset (although its score might not be that different). Finally, in the adversarial variant, the perturbations to the score are not arbitrary. Instead, they force the score to be more attentive to a subset of the training instances. Our experimental results show that substantial improvements are achieved in a variety of training scenarios.

Weight Perturbation for Escaping Maxima

Generic Search Procedure Our goal is to perturb the training data to allow local search to escape poor local maxima, under the constraint that we ultimately find a hypothesis that scores well on the original training distribution. Therefore, the perturbations should not move the training data too far from their original state, and eventually the data must be restored to its original form to ensure that the final hypothesis is optimized on the correct distribution. This suggests that we follow an annealing approach where we allow the weights to change freely early in the search, but then eventually "cool" the weights toward the original distribution.

Figure 1 outlines the generic search procedure we examine. The free parameters in this procedure are the annealing schedule ($reduce(\tau, t)$), the local search method ($optimize(\text{Score}, \mathrm{w}, h, D)$), and the example reweighting scheme ($reweight(\text{Score}, \mathrm{w}, \tau, h, D)$), each of which we instantiate below. For the annealing schedule, we follow a standard initialization with standard decay, starting with temperature τ^0 and setting $\tau^{t+1} = \delta\tau^t$, with $\delta = 0.95$ in most runs. For local optimization, one issue is to note that local search can be interleaved with the example reweighting in many ways. For example, one could perform full local optimization between each reweighting step, or perform only a partial optimization between reweights. We apply both of these interleaving strategies in specific cases below, depending on what appears to be most advantageous for the problem at hand. The final component of our search procedure is the reweighting method, for which we propose the following two main strategies.

Random Reweighting The first approach we consider is a randomized method motivated by *iterative local search* methods in combinatorial optimization (Codenotti *et al.* 1996) and phylogenetic reconstruction (Nixon 1999). Instead of performing random steps in the hypothesis space, we perturb the score by randomly reweighting each training example around its original weight. Candidate hypotheses are then evaluated with respect to the reweighted training set and we perform standard optimization on the perturbed score. After each iteration is complete, we repeat the process by independently sampling new example weights, reoptimizing the hypothesis, etc., until the magnitude of the weight perturbation approaches zero.

For convenience, we require the weights to be a probability distribution over the M data instances. Thus, we sample with a *Dirichlet* distribution with parameter β, so that $P(W = w) \propto \prod_m w_m^{\beta-1}$ for legal probability vectors (see, for example (DeGroot 1989)). When β grows larger, this distribution peaks around the uniform distribution. Thus, if we use $\beta = 1/\tau^t$ the randomly chosen distributions will anneal toward the uniform distribution, since the temperature τ^t decreases with the number of iterations t. We refer to this random perturbation approach as **Random**.

Adversarial Reweighting The second approach we consider is to update weights to directly challenge the current hypothesis. This approach is motivated by the exponential gradient search of (Schuurmans, Southey, & Holte 2001) for constrained optimization problems. Here, we combine their technique with an annealing process and modify it for a machine learning context. Intuitively, one can challenge a local maxima by calculating the gradient of the score with respect to the weights and then updating the weights to *decrease* the hypothesis' score. For example, on a training item $\mathbf{x}[m]$ one could consider the adversarial weight update $w_m^{t+1} \leftarrow w_m^t - \eta \frac{\partial \text{Score}}{\partial w_m}$ which would explicitly make the current hypothesis appear less favorable and hence less likely to remain a local maximum. In this way, $\text{Score}(h, D, \mathrm{w})$ behaves somewhat like a Lagrangian, in the sense that the local search attempts to maximize the score over h whereas the weight update attempts to *minimize* the score over w, in an adversarial min-max fashion.

This general approach still has to be adapted to our needs.

First, for reasons outlined above, we need to anneal the weight vector toward a uniform distribution. Therefore we add a penalty for divergence between w^{t+1} and uniform weights w^0. We use the Kullback-Leibler measure (Kullback & Leibler 1951) to evaluate the divergence between the distribution of the weights with respect to the original weights. We heighten the importance of this term as time progresses and the temperature is cooled down by evaluating $\beta\, KL(\mathrm{w}^{t+1} \| \mathrm{w}^0)$ where $\beta \propto 1/\tau^{t+1}$. Second, to maintain positive weight values we follow an exponential gradient strategy and derive a multiplicative update rule in the manner of (Kivinen & Warmuth 1997) where a penalty term for the for the KL-divergence between successive weight vectors w^{t+1} and w^t is added. All of these adaptations to our general schema lead us to use the following penalized score to guide our weight updates

$$\begin{aligned} L(h, \mathrm{w}^{t+1}) &= \mathrm{Score}(h, D, \mathrm{w}^{t+1}) \\ &+ \beta\, KL(\mathrm{w}^{t+1} \| \mathrm{w}^0) \\ &+ \gamma\, KL(\mathrm{w}^{t+1} \| \mathrm{w}^t) \end{aligned}$$

where $1/\beta$ and $1/\gamma$ are proportional to the temperature and enforce proximity to uniform weights and the previous weights respectively.

There are two ways to use this function to derive weight updates. The first is to explicitly minimize the penalized score by solving for w^{t+1} in $\nabla_{\mathrm{w}^{t+1}} L(h, \mathrm{w}^{t+1}) = 0$. If the score function is convex in w (as it often is) the solution can be quickly determined by iteration. A second, a more expedient approach is suggested by Kivinen and Warmuth (1997): Instead of doing full optimization, we heuristically fix the score gradient in $\nabla_{\mathrm{w}} L$ to its value at w^t and *analytically* solve for the minimum of the resulting approximation, up to a step size parameter η. This is tantamount to approximating the optimal update by taking a fixed step of size η in the exponential gradient direction from the current weight vector w^t. Omitting the details, we recover the multiplicative update formula

$$w_m^{t+1} = \alpha^{t+1} (w_m^0)^{\frac{\beta}{\beta+\gamma}} (w_m^t)^{\frac{\gamma}{\beta+\gamma}} e^{-\frac{\eta}{\beta+\gamma} \left(\frac{\partial \mathrm{Score}}{\partial w_m} |_{w_m^t} \right)}$$

where α^{t+1} is just a normalization constant. We refer to this approach as the **Adversary** strategy.

In sum, our second basic reweighting approach is to make adversarial weight updates by following the negative gradient in a well motivated function. This approach can be applied whenever the original weighted score is differentiable *with respect to the weights*, for any fixed hypothesis. Note this is a very weak requirement that is typically satisfied in machine learning scenarios. In particular, differentiability with respect to the weights has nothing to do with the discreteness or continuity of the hypotheses—it is a property of how the instance weights affect the score of a fixed hypothesis. Thus one could apply the adversarial reweighting approach to standard decision tree and neural network training problems without modification.

Although the adversarial strategy has many similar advantages to the randomized approach, one distinction is noteworthy: randomness is replaced by a guided methodology where weights are perturbed to minimize an intuitive function. This loses some of the flexibility of a random approach, which may reach the optimal solution by chance, but promises a far better average solution since it benefits from superior guidance.

Relation to ensemble reweighting There are interesting relationships between these reweighting techniques and ensemble learning methods like boosting (Schapire & Singer 1999). However, our techniques are not attempting to build an ensemble, and although they are similar to boosting on the surface, there are some fundamental differences. On an intuitive level, one difference is that boosting attempts to build a weighted ensemble of hypotheses that achieves a good score, whereas we are deliberately seeking a single hypothesis that attains this. On a technical level, boosting derives its weight updates by differentiating the loss of an entire ensemble (Mason *et al.* 2000), whereas our weight updates are derived by taking only the derivative of the score of the most recent hypothesis. Interestingly, although we do not exploit a large ensemble, we find that our methods still produce hypotheses that generalize well to unseen *test data*. Although surprising initially, this phenomenon is explained by the fact that example reweighting discovers hypotheses that obtain good scores while simultaneously being robust against perturbations of the training data, which confers obvious generalization benefits.

Learning Bayesian Networks from Data

To illustrate our approach on a general density estimation learning task we consider several problems in learning Bayesian networks from data: learning Bayesian network structure from complete data (structure search), optimizing Bayesian network parameters from incomplete data but given a fixed structure (parametric EM), and learning Bayesian network structure from incomplete data (Structural EM). Although the scores we encounter in these cases are not all additive, they are still differentiable and we can apply our methodology without modification.

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables. A *Bayesian network* is an annotated directed acyclic graph that encodes a joint probability distribution over $\mathcal{X}$. The nodes of the graph correspond to the random variables $X_1, \ldots, X_n$. Each node is annotated with a *conditional probability distribution* (CPD) that represents $P(X_i \mid \boldsymbol{U}_i)$, where $\boldsymbol{U}_i$ denotes the parents of X_i in G. A Bayesian network B specifies a unique joint probability distribution over $\mathcal{X}$ given by: $P(X_1, \ldots, X_n) = \prod_{i=1}^n P(X_i | \boldsymbol{U}_i)$.

Given a *training set* $D = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ we want to learn a Bayesian network B that *best matches* D, for each of the above scenarios. (See (Heckerman 1998) for a comprehensive overview of Bayesian network learning.) We explore each of the problems noted above in more detail in the subsequent sections.

Perturbing Structure Search

Structure Scores In this scenario, we search for a network structure B that best matches our *training set* D. In order to

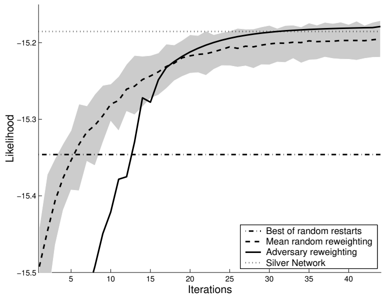


Figure 2: The progress of test set likelihood (log-loss/instance) during iterations while learning structure for the *Alarm* network. Compared are the golden model with parameters trained (**Silver** model), the best of random restarts search, the **Random** perturbation method and the **Adversary** method.

guide the search procedure, a scoring function for evaluating Bayesian network structures is used. A commonly used scoring function is the *BDe* score (Heckerman, Geiger, & Chickering 1995). A crucial property of the *BDe* (as well as other commonly used scores) is that it is a function of simple *sufficient statistics* of the data. For models in the exponential family, these sufficient statistics have a canonical form as a sum of functions applied to particular instances. Thus, if S is a sufficient statistic of interest, then

$$S(D) = \sum_m s(\mathbf{x}[m])$$

where $s()$ is a function of a particular training instance. For example, if $S(D)$ counts the number of times an event occurred in the data, then $s(\mathbf{x})$ is an indicator function that returns 1 if $\mathbf{x}$ satisfies the event, and 0 otherwise. When we perturb the score, we simply need to reweight the contribution of each instance. Thus, the perturbed statistic is

$$S(D, \mathrm{w}) = \sum_m M \cdot w_m \cdot s(\mathbf{x}[m])$$

Although the BDe score itself is not additive, it is nevertheless defined by sufficient statistics of the data and can therefore be easily adapted to the weighted case. (Details of the BDe score are given in the appendix.)

Once we specify the scoring function, the structure learning task reduces to a problem of searching over the combinatorial space of structures (i.e., DAGs) for the structure that maximizes the score. Since there are a super-exponential number of structures, an exhaustive search is infeasible. One common approach is to greedily follow local steps (add, remove or reverse an edge) using some search strategy (e.g. greedy hill-climbing) and incorporate a change into the current structure if it improves the score. Local search usually continues until convergence to a local maximum.

Note that the derivative of the *BDe* score with respect to w_m is straightforward (see appendix), and therefore both perturbation methods proposed above can be applied to this problem without change.

Experimental Evaluation We start by evaluating methods on the synthetic *Alarm* network (Beinlich *et al.* 1989) where we can compare our results to the "golden" model that has the additional prior knowledge of the true structure. We compare our methods to a greedy hill-climbing procedure that is augmented with a TABU-search mechanism and performs several random restarts to try to improve the quality of the results.[1] We apply our perturbation methods following the outline specified in Figure 1 and allow the search procedure to fully optimize with respect to the perturbed weights at each iteration.

It is possible to evaluate the results both in terms of scores on training data and generalization performance on test data (average log-loss per instance). In all of our experiments the two measures correlate closely, and therefore due to lack of space we report only test set performance. Figure 2 shows the progress of the likelihood during iterations of the perturbed runs. Shown are the average performance of 100 runs of the **Random** perturbation method (with the 20/80% margin in gray) and the **Adversary** method, compared to the best of random re-starts with similar running times. Several conclusions are notable. First, both perturbation methods solidly outperform the random re-starts method. In fact, both methods are able to outperform the golden model with parameters retrained on the training set (**Silver** model). Second, the best model overall is found by **Random**. However, **Adversary** is significantly better than **Random**'s average performance. This allows one to either invest a lot of time and achieve a superior model by performing many random perturbation runs or obtain a near optimal structure with a single **Adversary** run.

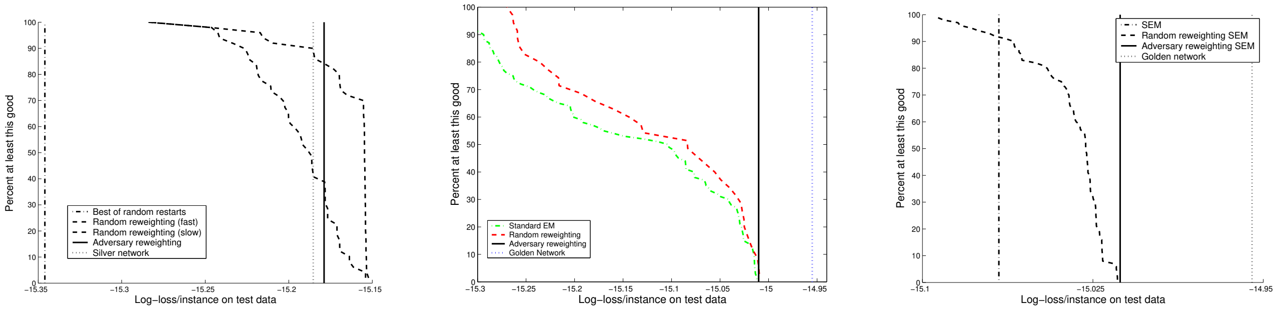
To emphasize this point, Figure 3(a) shows the cumulative performance of **Random** for different temperatures and cooling factors. The less favorable line has a similar running time to **Adversary** while the superior **Random** takes an order of magnitude longer for each run. These runs often reach what appears to be the achievable global maximum.

We also evaluated the performance of our methods for structure search on a real-life data set. *Stock Data* (Boyen, Friedman, & Koller 1999) is a dataset that traces the daily change of 20 major US technology stocks for several years (1516 trading days). As shown in Table 1, a significant improvement in the models learned by the perturbed method.

Perturbing Parametric EM

Learning with Incomplete Data In many real life datasets learning is complicated by the fact that some of the variables are not always observed. In such cases we say that there are *missing values* in the data. An even more difficult situation is when certain variables are never observed, we call such variables *hidden* or *latent* variables. In these scenarios, the m'th training instance is not a complete instance $\mathbf{x}[m]$, but only a partial instance, which we denote by $\mathbf{o}[m]$.

[1] We also tried standard simulated annealing, as described by Heckerman *et al.* (1995). However, using the parameters proposed by Heckerman *et al.* we get worse results then the baseline search method. This is consistent with Chickering (1996), who showed that using multiple re-starts greedy hill-climbing is more effective than simulated annealing for this task.



(a) Structure search runs (b) Parametric EM runs (c) SEM runs

Figure 3: Cumulative performance on test data of the *Alarm* network. The x-axis shows test-set likelihood (log loss/instance), the y-axis shows percent of runs that achieve that likelihood or higher. Compared are 100 runs each of baseline learning method, computationally intensive **Random** perturbations and **Adversary**. (a) Shows results for structure search with complete data. (b) Shows results for parameter estimation for data set with 4 central variables hidden, where the structure is fixed to the true structure. (c) Shows results for estimation of the parameters as well as the structure from the same dataset.

Since we do not have complete instances, we cannot estimate sufficient statistics and learning becomes more complicated. A common method to cope with such situations is to use the *expectation-maximization* (EM) algorithm (Dempster, Laird, & Rubin 1977). In this method, in each iteration, we use the previously found model $P_0(X_1, \ldots, X_n)$ to compute the *expected sufficient statistics*

$$E[S(D) \mid P_0] = \sum_m \sum_{\mathbf{x}[m]} s(\mathbf{x}[m]) P_0(\mathbf{x}[m] \mid \mathbf{o}[m])$$

where $P_0(\mathbf{x}[m] \mid \mathbf{o}[m])$ is the probability, according to the model P_0, of the complete instance $\mathbf{x}[m]$ given the partial observation $\mathbf{o}[m]$. Once we have computed these expected sufficient statistics, we can evaluate the score with respect to them. This score is referred to as the *expected score*. The main EM theorem shows that the improvement of the expected score between P_0 and the new model is a lower bound on the improvement between the (true) scores of the two models.[2] The simplest application of EM in graphical models is for parameter learning (Lauritzen 1995; Heckerman 1998). In this case our maximization objective is just the likelihood of the model on the training data. To do this we maximize the expected likelihood at each iteration of the EM algorithm.

Escaping Local Maxima One of the additional benefits of the ideas for perturbing the weights suggested above is that they are readily applicable for this problem as well. Instead of using expected sufficient statistics, we compute reweighted expected sufficient statistics using our current weight vector

$$E[S(D, \mathrm{w}) \mid P_0] = \sum_m M \cdot w_m \sum_{\mathbf{x}[m]} s(\mathbf{x}[m]) P_0(\mathbf{x}[m] \mid \mathbf{o}[m])$$

It is clear that the maximum point of the expected score is not the maximum point of the true score (for otherwise one iteration suffices to get to the global maximum). Thus, the expected score is biased. In general, this bias is toward models that are in some sense similar to the one with which we computed the expected sufficient statistics. This suggests that we do not necessarily want to find the optimum of the expected score within EM iterations. Instead, we apply a limited number of EM iterations (i.e., one) within each *optimize* step of the **PerturbedSearch** procedure (Figure 1) and then reweight the instances. The general perturbation scheme is otherwise unchanged.

Experimental Evaluation We start by evaluating methods on the synthetic *Alarm* network. Figure 3(b) compares 100 runs of standard parametric EM, computationally intensive **Random** perturbation, and an **Adversary** run. Because of the limited number of EM iterations, **Adversary** takes only about 15 times longer than a single parametric EM run and **Random** takes around 50 times longer. We can clearly see the advantage of the **Adversary** method, which achieves what appears to be the global maximum. This maximum is reached by only a few of the random re-starts and **Random** perturbation runs, and is not far from the golden model that generated the test data.

Perturbing Structural EM

A more complex application of EM is the *Structural EM* (SEM) procedure for learning structures (Friedman 1997; 1998). Our problem is now two-fold: at each iteration we need to find an optimal structure and then optimize the parameters with respect to that structure. In order to do this, at each stage we compute the expected sufficient statistics for different structures, and use the structure score to compare them. By performing structure search *within* the EM iteration, the procedure attempts to optimize (or at least to improve) the expected score. Like standard EM, this procedure is guaranteed to improve at each iteration and typically converges to a local maxima. However, in practice the optimization problem now is much more complex with many local maxima, especially when *hidden* variables abound.

[2] This statement of course depends on the score. It is true for the likelihood score (Dempster, Laird, & Rubin 1977; Lauritzen 1995) and for the MDL/BIC scores (Friedman 1997), and holds approximately for Bayesian structure scores (Friedman 1998).

Experimental Evaluation The setting is identical to the one used in the EM runs, but we also attempt to learn the topology of the network. The starting point of the search is a structure where all the hidden variables are parents of all the observables. Figure 3(c) shows cumulative results 100 runs for the synthetic *Alarm* example. The standard structural EM (**SEM**) runs have a very low variance and are worse than over 90% of the **Random** perturbation runs. As with parametric EM, but more markedly, the **Adversary** method dominates random reweighting. Note that it halves the distance from the baseline performance to the golden model performance.

Experiments with real-life data Finally, we applied our perturbation methods to several real-life datasets: From the UCI machine learning repository, we used the *Soybean* (Michalski & Chilausky 1980) disease database that contains 35 variables relevant for the diagnosis of 19 possible plant diseases, and the *Audiology* data set (Bareiss & Porter 1987) which explores illnesses relating to audiology dysfunctions. Both data sets have many missing values and comparatively few examples. We also used data from Rosetta's compendium (Hughes *et al.* 2000), using the preprocessing of (Pe'er *et al.* 2001), consisting of 300 examples over 6000 *Saccharomyces cerevisiae* genes. We chose 37 genes which participate in the *stationary phase* stress response.

For each data set we performed 5-fold cross validation and compared the log-loss performance on independent test data. Table 1 summarizes the results. Shown are results for best of multiple random restarts SEM, average and 80% values of **Random** perturbation runs, and the **Adversary** method. We can see that, as for the synthetic *Alarm* data, both perturbation methods achieve superior results to standard random restarts SEM. Similar to what was observed in structure search, it is sometimes possible to reach a superior model to **Adversary** by performing many **Random** perturbation runs.

As we see, in all domains, the perturbation methods improve over the baseline. Although the improvement per instances seems small, we stress that as one gets closer to the optimum, achieving additional improvements becomes more difficult. The results on synthetic data suggest that our methods are often very close to the global maximum achievable on the given training data.

Table 1: Summary of results on independent test data for several data sets for the structure search and structural EM problems. Shown are log-loss/instance of improvement in performance with respect to the **Best** of random restarts baseline. Compared are the mean of the **Random** perturbation method (along with the **80%** mark) and the **Adversary** method.

	Domain	**Random**	**80%**	**Adv**
Search	Stock	-0.02	+0.01	+0.03
	Alarm	+0.15	+0.18	+0.17
SEM	Rosetta	-0.05	+0.27	+0.09
	Audio	0	+0.39	+0.23
	Soybean	+0.19	+0.32	+0.19
	Alarm	+0.254	+0.31	+0.33

Learning Sequence Motifs

All of our case studies so far have addressed unsupervised density estimation problems. We now examine a different learning situation where we consider discriminative learning. The problem here is to perform a non-linear logistic regression to find *regulatory motifs* in DNA promoter sequences; *i.e.*, short subsequences that regulate the expression of genes. In particular, a motif is defined as a relatively short signature of about 8-20 nucleotides (DNA letters) that appears somewhere in the DNA sequence—the exact location of which is unknown and can vary from sequence to sequence. An example of a motif might be the sequence `ACGCGT` for instance. Unfortunately, most known motifs are not preserved perfectly in DNA, and one generally has to allow for substitutions in one or several positions. Accordingly, the common representation of a motif is as a *weight matrix* (Durbin *et al.* 1998) which describes the weight of each of the four possible DNA letters for each position within the motif. Intuitively, a subsequence that has a large sum of letter weights is said to match the motif. We use the notation $w_i[x]$ to denote the weight of the letter x in the i'th position.

Following (Barash, Bejerano, & Friedman 2001; Segal *et al.* 2002) we define the basic training problem in discriminative terms. Given N promoter sequences $\mathbf{s}_1, \ldots, \mathbf{s}_N$, where the n'th sequence consists of K letters $s_{n,1} \ldots, s_{n,K}$, and a set of of training labels $l_1, \ldots, l_N$, where l_i is 1 if the sequence is regulated by the motif and 0 if it is not (these labels can be obtained from different biological experiments), we wish to maximize the log-loss $\sum_i P(l_n \mid s_n)$ where

$$\begin{aligned} & P(l_n = 1 \mid S_{n,1}, \ldots, S_{n,K}) \\ & \quad = \quad \mathit{logistic}\left(\log\left(\frac{v}{K}\sum_j \exp\{\sum_i w_i[S_{n,i+j}]\}\right)\right) \end{aligned}$$

and $\mathit{logistic}(x) = \frac{1}{1+e^{-x}}$ is the logistic function and v is a threshold parameter; see Segal *et al* (2002) for more details.

Segal *et al.* (2002) address this problem in two stage. First, they search for high scoring seeds by considering all short words of length 6 using methods of Barash *et al* (2001). Then, for each seed they constructed a weight matrix of 20 positions that embodies the seed consensus sequence in the middle positions (the weights in flanking positions were initialized to 0), and then using conjugate gradient ascent (Price 1992) to maximize the log-likelihood score.

We adopt the same procedure augmented with our weight perturbation methods. After each weight perturbation, we perform a line search (Price 1992) in the direction of the gradient of the likelihood with respect to the reweighted samples. After the end of cooling schedule, we apply a final conjugate gradient ascent to find the local maxima in the vicinity of the final point of the search.

We applied this procedure to the 9 training sets generated during the analysis that Segal *et al.* performed on DNA-binding experiments of Simon *et al.* (2001). We report the

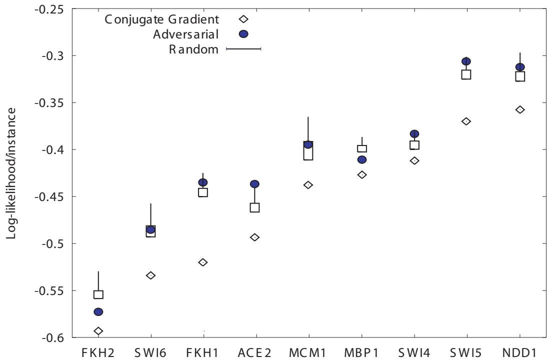


Figure 4: Performance of different methods in the motif finding task for 9 data sets. The x-axis corresponds to the different datasets, and the y-axis reports training log-likelihood per instance. We report the performance of the baseline conjugate ascent method, Adversarial reweighting, and Random reweighting. The box plot show the range between 20% to the 80% of 50 Random reweighting runs, and the narrow lines on top of the box show the best result of these 50 runs.

results in Figure 4. As one can see, both Random and Adversarial reweighting are consistently better than the baseline approach. In some cases (ACE2, SWI4, SWI5) the Adversarial reweighting achieves scores better than all the random runs, in others (FKH1, NDD1) it is better than at least 80% of the random runs, and only in two (FKH2, MBP1) it is worse than 80% of the random runs.

Discussion and Future Work

In this paper we proposed an annealing like method for escaping local maxima. The essence of our approach is to perturb the problem rather than the solution, and look for optimal solutions in the perturbed problems. As we show, such perturbations allow one to overcome local maxima in several learning scenarios. On both synthetic and real-life data, this approach seems to lead to significantly improved models in learning structure with complete data, learning parameters with hidden variables, and learning both parameters and structure with incomplete data. The improvements are particularly impressive for the complex problem of learning both structure and parameters from missing data. The Random reweighting approach we introduce here has been applied in Friedman *et al.* (2002) and Barash and Friedman (2002) for learning phylogenetic trees and context-specific clustering models, respectively. Both papers report dramatic improvements with this approach.

The perturbation of instance weights is particularly attractive for learning problems. It is easy to find reweighted versions of standard learning scores. Moreover, example weights are easily incorporated into non-trivial learning procedures such as EM. First, one can exploit the expected sufficient statistics for efficient search, and second, randomize the expected score to often find better scoring models.

In this paper, we compared two strategies for generating the sequences of weights during the annealing: *randomized reweighting* and *adversarial reweighting*. Our results show that both approaches dominate the straw-man of multiple restart search. Randomized reweighting can sometimes achieve better performance, but this might require performing several annealing runs. The deterministic adversarial strategy has the advantage of achieving similar performance in a single run.

One class of approaches that might be related to the ones we describe here are the *deterministic annealing* methods (Rose 1998). These methods are similar in that they change the score by adding a "free energy" component. This component serves to smooth out the score landscape. Deterministic annealing proceeds by finding the (local) maxima at each iteration and then moves to a "colder" score that recovers more of the structure of the score of interest. Local ascent is then used to trace the maxima starting from the hypothesis of the previous iteration. At the outset the rationale for the scores used in deterministic annealing is quite different then our weight perturbation. It is unclear if there are deeper connections between the two methods.

One avenue that we did not explore in this paper is the combination of a randomized element within the adversarial strategy. It is also clear that for realistic applications, we need to tune the implementation to reduce the number of iterations. This can be done by incorporating more sophisticated cooling strategies from the simulated annealing literature (see (Laarhoven & Aarts 1987) for a review). It is also worth exploring improved ways to interleave the maximization and the reweighting steps. Finally, the empirical success of these methods raises the challenge of providing a better theoretical understanding of their effectiveness. This is particularly intriguing for the adversarial reweighting strategy. Although this strategy has similarities to boosting and multiplicative update strategies, the analysis of these methods does not seem to directly apply to our setting.

Acknowledgments The authors thank Yoseph Barash, Daphne Koller, Itsik Pe'er, Tal Pupko, and, in particular, Noam Lotner for useful discussions and comments on previous drafts of the paper. This research was supported in part by ISF grant 244/99 and Israeli Ministry of Science grant 2008-1-99. N. F. was supported by Alon Fellowship. D. S. was supported by NSERC and MITACS.

Appendix: Gradient of the Lagrangian

The Lagrangian used in the section for **Adversarial Reweighting** is of the form

$$\begin{aligned} L(h, \mathrm{w}^{t+1}) &= \mathrm{Score}(h, D, \mathrm{w}^{t+1}) \\ &+ \beta\, KL(\mathrm{w}^{t+1} \| \mathrm{w}^0) \\ &+ \gamma\, KL(\mathrm{w}^{t+1} \| \mathrm{w}^t) \end{aligned}$$

The method suggested in that section requires the calculation of the derivative of the Lagrangian with respect to each instance weight.

The derivative of $KL(w^{t+1} \| w^0)$ is simply

$$\frac{\partial \sum_m w_m^{t+1} \log \frac{w_m^{t+1}}{w_m^0}}{\partial w_m^{t+1}} = \log \frac{w_m^{t+1}}{w_m^0} + 1$$

and similarly for the derivative of $KL(w^{t+1}\|w^t)$.

For Bayesian network structure learning, we use the BDe Score (Heckerman, Geiger, & Chickering 1995). This require defining a prior distribution P^0 over the domain, and a prior strength parameter N^0. For each event $\mathbf{x}$, we define $\alpha(\mathbf{x}) = N^0 \cdot P^0(x)$, and $\alpha'(\mathbf{x}) = \alpha(\mathbf{x}) + N(\mathbf{x})$, where $N(\mathbf{x})$ is the number of occurrences of $\boldsymbol{X} = \mathbf{x}$ in the training data. The BDE score is defined as

$$\text{Score}_{\text{BDe}} = \sum_i \sum_{pa_i} \left(\log \frac{\Gamma(\alpha(pa_i))}{\Gamma(\alpha'(pa_i))} + \sum_{x_i} \log \frac{\Gamma(\alpha'(x_i, pa_i))}{\Gamma(\alpha(x_i, pa_i))} \right)$$

where x_i is an assignment to the i'th variable, pa_i is an assignment to the parents of the i'th variable, and $\Gamma(x)$ is the *gamma function*.

The only expressions that depend on the weights are $N(pa_i) = \sum_m w_m \cdot P(Pa_i = pa_i \mid e_m)$ and $N(x_i, pa_i)$, where e_m is the evidence of the m'th instance. Using the *digamma function* $\Psi(x) \equiv \frac{\Gamma'(x)}{\Gamma(x)}$ (DeGroot 1989), the derivative of $\text{Score}_{\text{BDe}}$ (for a specific value of pa_i) with respect to a specific weight w_m is given by

$$\frac{\partial \text{Score}_{\text{BDe}}}{\partial w_m} = \sum_i \sum_{pa_i} \sum_{x_i} (\Psi(\alpha'(x_i, pa_i)) - \Psi(\alpha'(pa_i))) P(x_i, pa_i \mid e_m)$$

which can readily be evaluated using a numerical approximation to the digamma function.

References

Barash, Y., and Friedman, N. Context-specific Bayesian clustering for gene expression data. *J. Comp. Bio.* 9:169–191.

Barash, Y.; Bejerano, G.; and Friedman, N. 2001. A simple hyper-geometric approach for discovering putative transcription factor binding sites. In *Algorithms in Bioinformatics: Proc. First International Workshop*, 278–293.

Bareiss, R., and Porter, B. 1987. Protos: An exemplar-based learning apprentice. *Proceedings of the 4th International Workshop on Machine Learning* 12–23.

Beinlich, I.; Suermondt, G.; Chavez, R.; and Cooper, G. 1989. The ALARM monitoring system. In *Proc. 2'nd European Conf. on AI and Medicine*.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*.

Boyen, X.; Friedman, N.; and Koller, D. 1999. Learning the structure of complex dynamic systems. In *UAI '99*.

Chickering, D. M. 1996. Learning equivalence classes of Bayesian network structures. In *UAI '96*. 150–157.

Codenotti, B.; Manzini, G.; Margara, L.; and Resta, G. 1996. Perturbation: An efficient technique for the solution of very large instances of the TSP. *INFORMS Journal on Computing* 8(2):125–133.

DeGroot, M. H. 1989. *Probability and Statistics*.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B 39:1–39.

Durbin, R.; Eddy, S.; Krogh, A.; and Mitchison, G. 1998. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*.

Friedman, N.; Ninio, M.; Peer, I.; and Pupko, T. 2002. A structural EM algorithm for phylogenetic inference. *J. Comp. Bio.* 9:331–353.

Friedman, N. 1997. Learning belief networks in the presence of missing values and hidden variables. In *ICML '97*. 125–133.

Friedman, N. 1998. The Bayesian structural EM algorithm. In *UAI '98*.

Glover, F., and Laguna, M. 1993. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20:197–243.

Heckerman, D. 1998. A tutorial on learning with Bayesian networks. In *Learning in Graphical Models*.

Hughes, T. R.; Marton, M. J.; *et al* 2000. Functional discovery via a compendium of expression profiles. *Cell* 102(1):109–26.

Kirpatrick, S.; Gelatt, Jr., C.; and Vecchi, M. 1994. Optimization by simulated annealing. *Science* 220:671–680.

Kivinen, J., and Warmuth, M. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation* 132:1–63.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *Annals of Mathematical Statistics* 22:76–86.

Laarhoven, P., and Aarts, E. 1987. *Simulated Annealing: Theory and Applications*.

Lauritzen, S. L. 1995. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis* 19:191–201.

Mason, L.; Baxter, J.; Bartlett, P.; and Frean, M. 2000. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*.

Michalski, R., and Chilausky, R. 1980. Learning by being told and learning from examples. *International Journal of Policy Analysis and Information Systems* 4(2).

Nixon, K. C. 1999. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics* 15:407–414.

Pe'er, D.; Regev, A.; Elidan, G.; and Friedman, N. 2001. Inferring subnetworks from perturbed expression profiles. *Bioinformatics* 17(Suppl 1):S215–24.

Price, W. H. 1992. *Numerical Recipes in C*.

Rose, K. 1998. Deterministic annealing for clustering, compression, classification, regression and related optimization problems. *Proc. IEEE* 80:2210–2239.

Schapire, R. E., and Singer, Y. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning* 37.

Schuurmans, D.; Southey, F.; and Holte, R. 2001. The exponentiated subgradient algorithm for heuristic Boolean programming. In *Proceedings IJCAI-01*, 334–341.

Segal, E.; Barash, Y.; Simon, I.; Friedman, N.; and Koller, D. 2002. From promoter sequence to expression:a probabilistic framework. In *RECOMB '02*.

Simon, I.; Barnett, J.; Hannett, N.; Harbison, C.; Rinaldi, N.; Volkert, T.; Wyrick, J.; Zeitlinger, J.; Gifford, D.; Jaakkola, T.; and Young, R. 2001. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell* 106:697–708.