Constructing Informative Priors using Transfer Learning

Rajat Raina RAJATR@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science Department, Stanford University, CA 94305 USA

Abstract

Many applications of supervised learning require good generalization from limited labeled data. In the Bayesian setting, we can try to achieve this goal by using an informative prior over the parameters, one that encodes useful domain knowledge. Focusing on logistic regression, we present an algorithm for automatically constructing a multivariate Gaussian prior with a full covariance matrix for a given supervised learning task. This prior relaxes a commonly used but overly simplistic independence assumption, and allows parameters to be dependent. The algorithm uses other "similar" learning problems to estimate the covariance of pairs of individual parameters. We then use a semidefinite program to combine these estimates and learn a good prior for the current learning task. We apply our methods to binary text classification, and demonstrate a 20 to 40% test error reduction over a commonly used prior.

1. Introduction

Classical supervised learning algorithms find good classifiers for a given learning task using labeled input-output pairs. When labeled data is limited and expensive to obtain, an attractive alternative is to use other data sources to improve performance. For example, semi-supervised learning uses unlabeled data for the given learning task.

In this paper, we use *transfer learning* (Baxter, 1997; Thrun, 1996; Caruana, 1997) to improve performance on the learning task at hand. Transfer learning utilizes labeled data from other "similar" learning tasks. It is inspired by the observation that humans do not receive tasks in isolation, but instead receive a sequence of learning tasks over their lifetimes. It appears intuitive that learning a sequence of related tasks should be easier than learning each of those tasks in isolation. For example, the visual system might find it easier to recognize a guava if it already knows how to recognize apples and oranges; it might be easier to learn French if one already knows English and Latin. Humans can probably discover some underlying structure in each of these domains, and can thus learn new but similar tasks quickly and easily.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

With this inspiration, we present a transfer learning algorithm that constructs an informative Bayesian prior for a given learning task. The prior encodes useful domain knowledge by capturing underlying dependencies between the parameters. The next section gives a brief overview of our approach.

2. Overview

For conciseness and clarity, in this paper we focus on binary text classification, even though our model can also be applied straightforwardly to other multiclass classification problems. To motivate our algorithm, let us first consider the traditional supervised learning setting for binary text classification.

In this setting, each input X is a text document and is assigned to a unique output label from the set $\{0,1\}$. A vocabulary of words $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ is given, and we assume every input document X is represented as a "bag-of-words" vector $X = (X_1, X_2, \ldots, X_{|\mathcal{W}|}) \in \{0,1\}^{|\mathcal{W}|}$, where X_i is 1 if word w_i occurs in document X and 0 otherwise. A labeled training set $M = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ is given, and the task is to predict the label y for a test document x.

A linear classifier for this setting can be defined using a parameter vector $\theta = (\theta_1, \theta_2, \ldots, \theta_{|\mathcal{W}|}) \in \mathbb{R}^{|\mathcal{W}|}$. For example, logistic regression makes predictions according to the rule $P(Y=1|X=x,\theta) = 1/\left(1+\exp(-\theta^T x)\right)$. Each parameter θ_i thus corresponds to a word w_i in the input vocabulary. Using the given training set M, the (discriminative) log-likelihood of a parameter setting θ can be computed as $\ell_M(\theta) = \sum_{i=1}^{m} \log P(Y = y^{(i)}|X = x^{(i)}, \theta)$. To avoid overfitting, usually one assumes a multivariate Gaussian prior $\mathcal{N}(0, \sigma^2 I)$ on the parameter vector θ (Nigam et al., 1999), and then finds the maximum-a-posteriori (MAP) estimate θ_{MAP} by maximizing the (penalized) log-likelihood of the training set M:

$$\theta_{\text{MAP}} \quad = \quad \arg\max_\theta \left(\ell_M(\theta) - \frac{1}{2\sigma^2}||\theta||_2^2 \right) \qquad (1)$$

When training data is extremely scarce ($m << |\mathcal{W}|$), the parameter values learnt in this way often produce poor performance on unseen test data. This is to be expected from a learning theoretic viewpoint—the prior distribution is only weakly informative as it assumes that the parameters θ_i are independent of each other and have equal prior variance.

However, many classification problems naturally display rich structure in their features. Text documents, for example, generally use many words drawn from a small set of topics. Thus, the occurrence of a word such as `moon` in a document with label y might make it more likely that words "similar" to `moon` (such as `rocket` or `astronaut`) will occur in other documents with the same label y. Further, there might be systematic trends making rare words more or less informative about a document label than common words. We aim to model these dependencies by placing a more informative prior over the parameters. In particular, we will construct a Gaussian prior $\mathcal{N}(0, \Sigma)$ where $\Sigma \in \mathbb{R}^{|\mathcal{W}|\times|\mathcal{W}|}$ is a (non-diagonal) covariance matrix. The off-diagonal entries of the matrix capture dependencies between parameters; further, unequal values on the diagonal allow the parameters to have different prior variance. For our example, if the prior covariance between the parameters for `moon` and `rocket` is highly positive, we can infer that `rocket` supports the label y even without observing this directly in training data. As a result, the algorithm can now correctly classify test documents containing words such as `rocket` that may not have occurred in the training set.

In this paper, we present an algorithm for learning the covariance matrix Σ using other labeled text classification problems, which we shall call the *auxiliary* learning problems. The original problem (i.e., the one we ultimately want to perform well on) will henceforth be called the *target* problem. For our running example, suppose that we observe from auxiliary problems that the parameters for words `moon` and `rocket` are positively correlated (i.e., in any auxiliary problem, they typically support the same label when they occur in a document). On average, these correlations are likely to provide good guesses for correlations in the target problem as well. We show that we can use such cues from auxiliary problems to efficiently construct a covariance matrix that also makes useful predictions for previously unseen words.

In the sequal, we will first describe a method for estimating a single covariance entry (such as $\text{Cov}(\theta_{\texttt{moon}}, \theta_{\texttt{rocket}})$) using auxiliary problems; then, we show how these individual estimates can be efficiently put together to obtain a complete covariance matrix Σ for the target problem.

3. Estimating the Covariance between Word Parameters

Suppose we want to compute the prior covariance between the parameters θ_i and θ_j of the target problem, corresponding to words w_i and w_j in its vocabulary.

Consider building a classifier for an auxiliary problem C using a random vocabulary that includes words w_i and w_j. For this vocabulary, assume that we know the optimal values for the parameters θ_i and θ_j—i.e., the values that produce the highest expected log-likelihood on test data for C. Let these optimal values be given by the random variables Θ_i^* and Θ_j^*, where the randomness is over the choice of the random vocabulary. The covariance of these random variables is given by $E[\Theta_i^*\Theta_j^*]$, since the class labels are symmetric and so $E[\Theta_i^*] = E[\Theta_j^*] = 0$. We will use this covariance as a surrogate for the desired prior covariance, and will estimate the former using labeled data for C. For our earlier example, $\Theta_{\texttt{moon}}^*$ and $\Theta_{\texttt{rocket}}^*$ might have similar values for most vocabularies, and thus their covariance $E[\Theta_{\texttt{moon}}^*\Theta_{\texttt{rocket}}^*]$ will be highly positive.

Concretely, we estimate the covariance $\text{Cov}(\theta_i, \theta_j)$ using Algorithm 1. The algorithm computes a Monte Carlo estimate for the expectation in $E[\Theta_i^*\Theta_j^*]$ by averaging over several vocabularies of a fixed size $K > 2$. Since the optimal parameters Θ_i^* and Θ_j^* are not known even for the auxiliary problem C, we also sample several training sets for each vocabulary. This produces several subproblems, each with a particular vocabulary and training set. For each subproblem, we use logistic regression with a weak prior to learn the parameters. With a small vocabulary size ($K = 5$, say) the learned parameters should be close to optimal. The sample covariance of these parameters over all subproblems gives an estimate for $E[\Theta_i^*\Theta_j^*]$.

The procedure described thus far has a drawback. The expectation in $E[\Theta_i^*\Theta_j^*]$ should be over the random choice of vocabulary only; the above procedure, however, also includes variance due to the random choice of the training set. It can be shown, for example, that when we compute variances by this procedure ($i = j$), the estimated variance *always* overestimates the true variance. To correct for the randomness due to the choice of the training set, we apply a bootstrap correction (Efron, 1979) to the above sample covariance. This correction term subtracts out an estimate for the covariance due to the choice of training set.

```
Algorithm 1 Estimate single covariance
Input: Words $w_i$ and $w_j$, auxiliary problem $C$, vocabulary size $K$.
Return: Estimate $\hat{\Sigma}_{ij}$ for the covariance $\mathrm{Cov}(\theta_i, \theta_j)$.
  Fix $V$ = number of vocabularies to sample.
  Fix $T$ = number of training sets to sample.
  for $v = 1$ to $V$ do
    Sample vocabulary $\mathcal{V}$ of size $K$, with $w_i, w_j \in \mathcal{V}$.
    for $t = 1$ to $T$ do
      Sample training set $\mathcal{T}$ from labeled data for $C$.
      $\theta^{(v,t)} \leftarrow$ Parameter vector learnt using logistic regression with vocabulary $\mathcal{V}$ on training set $\mathcal{T}$.
    end for
    $\bar{\theta}^{(v)} \leftarrow (1/T) \sum_t \theta^{(v,t)}$
    $C^{(v)} \leftarrow (1/T) \sum_t (\theta_i^{(v,t)} - \bar{\theta}_i^{(v)})(\theta_j^{(v,t)} - \bar{\theta}_j^{(v)})$
  end for
  $U \leftarrow (1/VT) \sum_{v,t} \theta_i^{(v,t)} \theta_j^{(v,t)}$    (Sample covariance)
  return $\hat{\Sigma}_{ij} = U - (1/V) \sum_v C^{(v)}$    (Bootstrap)
```

With this algorithm, each estimate can be computed using $O(VT)$ calls to a logistic regression routine in the inner loop; small values of V and T are found to be sufficient ($V = T = 4$ in our experiments). A similar algorithm can also be used to estimate the variance $\mathrm{Var}(\theta_i)$ corresponding to a word w_i.

As a sanity check for this procedure, we performed the following experiment: we constructed 10 binary text classification problems using the 20 newsgroups dataset (Lang, 1995), as described later in Section 5. We isolated a random target problem ("Motorcycles" vs. "MS-Windows"), and used the other 9 problems as auxiliary problems. Table 1 shows the word pairs from this target problem's vocabulary whose covariance was found to be the most positive or most negative when estimated using the auxiliary problems. The word pairs with the highest positive covariance seem to be related to similar topics; some of the word pairs with the most negative covariance also capture some useful distinctions between the two classes. These relations between words were uncovered without looking at any labels for the target problem.

4. Constructing the Covariance Matrix

We could try to construct the desired covariance matrix Σ by computing covariance estimates for all word pairs and putting them together in a matrix. However, this would pose two major problems:

Table 1. Word pairs from the classification problem "Motorcycles" vs. "MS-Windows" estimated to have the most positive (left) or most negative (right) bootstrap-corrected parameter covariance using auxiliary learning problems.

Most positive covariance		Most negative covariance	
insurance	mile	wave	mouse
rear	mile	air	resource
honda	mile	wave	menu
brake	gear	air	server
meg	printer	ground	server
brake	wheel	object	ram
bmw	seat	battery	mouse
desktop	ram	low	server

1. It may not be feasible to get covariance estimates for all word pairs using the above algorithm. There might be previously unseen words—if a word pair has never been seen in any auxiliary problem data, Algorithm 1 cannot produce meaningful estimates for the parameter covariance. Further, the total number of covariance estimates needed would grow quadratically with the size of the vocabulary. For large vocabularies, this becomes computationally impractical.

2. A valid covariance matrix must be positive semidefinite (PSD). However, our covariance estimates will invariably have noise, and might arise from different auxiliary problems; the matrix formed by these estimates need not be PSD.

4.1. Learning Individual Covariances

To address the first problem, instead of learning the entries themselves, we propose to learn general transformations that generate these entries. In particular, we model covariance between a pair of parameters as a function of *features* of the corresponding word pair. For example, a feature of a word pair might check if these words are synonyms—if they are, there might be a high covariance between their parameters. Given such features, we can estimate a small fraction of the covariances directly using the auxiliary learning problems, and then learn a general transformation of the features that allows us to approximate all the missing entries in the covariance matrix. As we show later in the results section, a small number of simple features are sufficient to ensure good performance.

More formally, suppose we extract a feature vector $F_{ij} \in \mathbb{R}^S$ for each matrix position (i, j), such that every element of F_{ij} is a feature of the word pair (w_i, w_j) and the vocabulary $\mathcal{W}$. For example, a particular element of F_{ij} might be 1 if w_i and w_j are synonyms, and 0 otherwise. We then approximate each entry in the covariance matrix as a linear function of the corresponding feature vector—i.e., given a suitable parameter vector $\psi \in \mathbb{R}^S$, we construct a candidate matrix $\hat{\Sigma}$ by computing its $(i,j)^{th}$ element as follows:

$$\hat{\Sigma}_{ij} \quad = \quad \psi^T F_{ij} \tag{2}$$

If synonyms generally have parameters with highly positive covariance, we would want features such as the "synonym-check" feature to have a highly positive corresponding weight in the parameter vector ψ. The features we used are described in Section 4.4.

We now present an algorithm for learning ψ. Define the set $G = \{(i,j) \mid \text{Cov}(\theta_i, \theta_j)$ was directly estimated$\}$, and let e_{ij} be the value of the covariance estimate for position $(i,j) \in G$. Given these "desired" values e_{ij} for some entries in the covariance matrix, ψ might be learnt as a small supervised learning task—we could pick ψ so that the values $\hat{\Sigma}_{ij}$ generated using Equation (2) are as "close" as possible to the available desired values e_{ij}. For example, linear regression would minimize the following squared-error criterion:

$$\min_{\psi} \sum_{(i,j)\in G} (e_{ij} - \psi^T F_{ij})^2 \tag{3}$$

Once ψ has been chosen, the candidate matrix $\hat{\Sigma}$ can be filled in using Equation (2).

4.2. Learning a Positive Semidefinite Matrix

The above method may learn useful patterns, but the final matrix $\hat{\Sigma}$ may still not be PSD. We could project $\hat{\Sigma}$ onto the PSD cone to obtain the closest valid covariance matrix; however, $\hat{\Sigma}$ often turns out to be highly indefinite, and so the projected matrix is "far" from the matrix $\hat{\Sigma}$ originally learnt through the parameters ψ. This makes the earlier method of learning ψ unsatisfactory, since the final PSD projection is oblivious to any underlying preferences and might "unlearn" some of the patterns learnt through the parameters ψ.

It turns out that this sequential procedure (i.e., first learn ψ, then find closest PSD matrix) is unnecessary. We can learn good values for ψ while also considering the final PSD constraint. If Σ is the final covariance matrix produced, we make the PSD constraint explicit by posing the following joint optimization problem over variables ψ and Σ:

$$\min_{\psi,\Sigma} \sum_{(i,j)\in G} (e_{ij} - \psi^T F_{ij})^2 + \lambda \sum_{i,j} (\Sigma_{ij} - \psi^T F_{ij})^2 \tag{4}$$
$$\text{s.t.} \quad \Sigma \succeq 0$$

where Σ_{ij} denotes the $(i,j)^{th}$ element of matrix Σ. The objective function provides a trade-off between two different goals. The first term encourages ψ to better approximate the available covariance estimates e_{ij}, as in Equation (3); the second term encourages ψ to generate a matrix close to a PSD matrix Σ. The positive number λ controls the relative importance assigned to these two terms. For example, in the limit of $\lambda \to 0$, this becomes equivalent to the previous sequential method.

Importantly, the optimization problem in (4) is convex *jointly* in ψ and Σ, and thus possesses a unique global optimum over these variables. In fact, the problem can be written as a semidefinite program (SDP), and can be solved directly using standard SDP solvers.

However, an even more scalable method for optimization can be derived using alternating minimization over the variables ψ and Σ. This procedure consists of two alternating steps, repeated until convergence. First, keep Σ fixed and optimize the objective function in (4) only over ψ; this problem reduces to a QP similar to (3) and can be solved with a fast QP solver. Then, fix ψ and optimize the objective only over Σ; this problem reduces to minimizing only the second term in the objective, subject to the PSD constraint. This is a particularly simple SDP—it involves projecting the matrix formed by $\psi^T F_{ij}$ onto the PSD cone, which can be done efficiently by finding its eigendecomposition and taking only the components with nonnegative eigenvalues. All of these steps can be performed efficiently for large problems. This alternating optimization method is guaranteed to converge to the global minimum since the objective function is convex.

4.3. Algorithm Details

The matrix Σ generated using the SDP (4) nicely captures the relative magnitudes of the covariances, but does not adequately capture their absolute scale. This is to be expected because the covariance entries were estimated on auxiliary problems with a fixed small vocabulary size ($K = 5$ in our case), whereas the target problem generally uses a different, larger vocabulary size where the parameters might have differently scaled magnitudes. We thus allow a single scaling parameter q, and use $q\Sigma$ as the final covariance matrix in the prior. The final results are fairly insensitive to the exact value of this scaling parameter, as long as it has the right order of magnitude. In practice, simply hard-coding a single (training set size specific) value of q for all problems leads to only minor differences in the final results. We will later describe a method to learn q from auxiliary data.

Algorithm 2 summarizes the complete method for estimating a covariance matrix for a target problem.

Table 2. List of feature functions used. The left half shows features for a single word w_i, for the diagonal entries of the covariance matrix. The right half shows features for a word pair (w_i, w_j), for the off-diagonal entries. Σ_{nemp} is a normalized empirical co-occurrence matrix, estimated from a large English corpus. $\Sigma_{\text{lowrank}} = \min_A ||A - \Sigma_{\text{nemp}}||_F$, where the minimization is over all $\mathcal{W} \times \mathcal{W}$ matrices of rank at most $\mathcal{W}/5$. Σ_{lowrank} can be efficiently computed using an SVD.

Constant (always 1)	Constant (always 1)
$(i,i)^{th}$ element of Σ^r_{nemp} for $r = 2, 3$	$(i,j)^{th}$ element of Σ^r_{nemp} for $r = 1, 2, 3$
$(i,i)^{th}$ element of Σ_{lowrank}	$(i,j)^{th}$ element of Σ_{lowrank}
Raw frequency of occurrence	Raw frequency of co-occurrence, log transform
Log transformed frequency of occurrence	Distributional similarity score computed by Infomap
	Check if words are synonyms or hypernyms using WordNet

```
Algorithm 2 Estimate covariance matrix
Input: Target vocabulary $\mathcal{W}$, set of auxiliary problems $\mathcal{C}$.
Return: Covariance matrix $\Sigma$ for a Gaussian prior.
  Fix $S$ = number of covariance entries to estimate directly using auxiliary problems.
  Initialize $G = \{\}$.
  for $s = 1$ to $S$ do
    Pick a word pair $w_i, w_j$ from vocabulary $\mathcal{W}$, such that $w_i, w_j \in$ some auxiliary problem $C \in \mathcal{C}$.
    $G \leftarrow G \cup (i, j)$.
    $e_{ij} \leftarrow$ Cov estimate using C for $(\theta_i, \theta_j)$. (Alg. 1)
  end for
  Compute all feature vectors $F_{ij}$ $(1 \leq i, j \leq |\mathcal{W}|)$.
  $\Sigma_{\text{SDP}} \leftarrow$ Optimal $\Sigma$ for SDP (4).
  Pick scaling parameter $q$ (or set to default value).
  return $\Sigma = q\Sigma_{\text{SDP}}$
```

4.4. Features Used

We used two different parameters ψ_1 and ψ_2 to construct the diagonal and off-diagonal entries of the covariance matrix; this is natural as different features are useful in these two cases. The joint optimization problem (4) can be used with minor changes. The features used for diagonal and off-diagonal entries are listed in Table 2. Most features used a normalized empirical word co-occurence matrix derived from a large English corpus: if the empirical word co-occurence matrix is Σ_{emp}, we used the matrix $\Sigma_{\text{nemp}} = D^{-1/2}\Sigma_{\text{emp}}D^{-1/2}$ where D is a diagonal matrix of the same size as Σ_{emp} with each diagonal entry containing the corresponding row-sum from Σ_{emp}.[1] We used higher powers of this co-occurence matrix and low-rank approximations to it, as they might capture higher-order word correlations beyond simple co-occurence (e.g., `astronaut` co-occurs often with `moon`, which co-occurs often with `cosmonaut`; but `astronaut` and `cosmonaut` might co-occur very rarely). To provide basic linguistic features, we used a distributional similarity score computed by Infomap (`http://infomap.stanford.edu`) and checked if the words are related as synonyms or hypernyms in WordNet (Miller, 1995).

[1]This matrix is often called the normalized Laplacian in spectral graph theory (Chung, 1997; Ng et al., 2002).

5. Data and Methods

We used the standard 20 newsgroups dataset (Lang, 1995). This dataset contains documents from 20 document classes, each derived from postings to a separate newsgroup. The text data was preprocessed by stemming and removing stopwords. The newsgroup classes were randomly paired to construct 10 binary classification problems. The vocabulary for each of these classification problems was constructed by picking the 250 most frequent words from each constituent newsgroup. To construct a transfer learning setup, we conducted the following hold-out experiment using these 10 classification problems: each of the 10 problems was in turn treated as the target problem, and the remaining 9 were considered auxiliary problems; in each case, covariance estimates were generated from the auxiliary problems using Algorithm 1 and were used to solve the SDP for the held-out target problem.[2] This produced a covariance matrix $\Sigma_{\text{P,SDP}}$ for every held-out problem P. A scaling parameter was chosen, and the test error on P was then evaluated with the learnt prior using training sets of different sizes drawn from the labeled data for P.

For the sake of reproducibility, we describe the procedure used to pick q in the remainder of this section. Note that cross-validation on the target problem is not a practical procedure to pick q here, as it would not work well with very limited training data. The description below involves several details and the reader can skip to the next section without loss of continuity.

[2]For each target problem, we used the auxiliary problems to estimate about 75% of the diagonal entries and 20% of the off-diagonal entries. About 25% of the words in each target vocabulary do not occur in any auxiliary problem, and are "novel." For the QP, we used the SeDuMi (`http://sedumi.mcmaster.ca`) and Yalmip (`http://control.ee.ethz.ch/~joloef/yalmip.php`) packages for Matlab.

For our hold-out experiment, we estimate q using the auxiliary problems. In general, suppose we have access to a set of auxiliary problems $\mathcal{A}$ to estimate the scaling parameter for target problem P with a fixed training set size m, and that a covariance matrix $\Sigma_{\mathrm{C,SDP}}$ has already been computed by solving the SDP (4) for every $C \in \mathcal{A}$. For any single auxiliary problem $C \in \mathcal{A}$, we define a "goodness" function $g_C(q)$ for scaling parameter q as $E_{M,T}\ell_T(\theta_{\mathrm{MAP}}(q, M))$, where the expectation is over the random choice of labeled test example $T = (x, y)$ and labeled training set M of size m drawn from the labeled data for C; $\ell_T(\cdot)$ is the log-likelihood function as in Section 2; and $\theta_{\mathrm{MAP}}(q, M)$ is the MAP estimate for θ using training set M and the prior $\mathcal{N}(0, q\Sigma_{\mathrm{C,SDP}})$. The expectation can be computed by averaging over several training sets and test examples from the labeled data for C, solving a MAP problem for each sampled training set. Then, for target problem P, we can define the overall goodness $\mathcal{G}_{P,\mathcal{A}}(q)$ of scaling parameter q as $\mathcal{G}_{P,\mathcal{A}}(q) = \sum_{C\in\mathcal{A}} g_C(q)$. With this definition, we pick $q = q_{\mathrm{FINAL}}$ for target problem P as a local maximum of this overall goodness function $\mathcal{G}_{P,\mathcal{A}}(\cdot)$ by performing a local search using multiplicative coordinate-ascent. Then, the prior covariance matrix actually used for problem P is $q_{\mathrm{FINAL}}\Sigma_{\mathrm{P,SDP}}$.

Note that $\Sigma_{\mathrm{C,SDP}}$ must be computed above for all $C \in \mathcal{A}$ without using *any* labeled data for P, as we are constructing the prior for P itself. Thus, we compute $\Sigma_{\mathrm{C,SDP}}$ for $C \in \mathcal{A}$ using covariance estimates from all available problems except P (i.e., all problems in $\mathcal{A} - \{C\}$). For the whole hold-out process, it is sufficient to precompute 100 different covariance matrices: one each for each problem leaving out estimates from none or one of the other problems. In this way, no labeled data for the target problem is used for estimating q. Later, we also present results using the same fixed value of q for a large number of target problems.

6. Experiments

We compare our algorithm against a baseline that uses a uniform diagonal prior; for fairness, we also allow a scaling parameter on this baseline prior, and choose it by adapting the exact procedure that was used to choose the scaling parameter for the SDP-generated covariance matrices. To verify the advantage of using a non-diagonal covariance matrix, we also solved Equation (4) constraining Σ to be diagonal.

Figure 1(a-j) shows our test error results. The SDP-generated matrices produce lower average test error than baseline over the full range of training set sizes; they reduced error by 20-40% over the baseline and continue to be better than baseline even for a training set of 100 documents. The SDP-generated diagonal covariance matrix performs only marginally better than baseline, showing that the off-diagonal entries capture crucial dependencies.

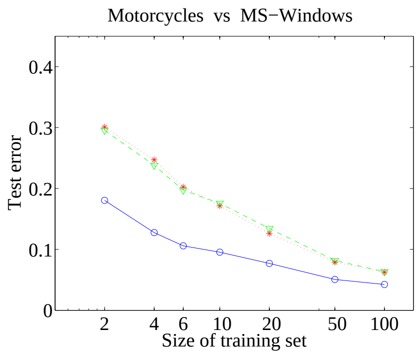
To graphically examine the final covariance matrices, we performed the following experiment: For each classification task, we used all the available training data to estimate the 50 "most informative" words per class. We picked the words with the highest weighted pointwise mutual information (WPMI) with each class label.[3] For example, for the classification problem "Motorcycles" vs. "MS-Windows", words such as `bike`, `ride` and `bmw` were chosen for the "Motorcycles" class, while words such as `version`, `file` and `program` were chosen for the "MS-Windows" class. From the full covariance matrix, we extracted the rows and columns corresponding to these words only, and formed a matrix so that all words picked from a class are together. Figure 1(k) shows this matrix, with brighter positions representing higher values. A rough block structure is evident, with words informative of the same class being assigned higher positive covariance on average. This demonstrates that the learnt prior is able to discover good word dependencies.

Several authors have noted that transfer learning can sometimes lower performance on the target problem (e.g., Caruana, 1997). Negative transfer was not observed on the 20 newsgroups dataset. We expect performance to depend strongly on the "relatedness" of the auxiliary problems to the target problem. Table 3 lists the classification problems that achieved the highest or lowest boost in performance due to transfer learning (as measured by percentage test error reduction over the diagonal covariance baseline). The two classification problems that achieved least transfer (bottom two rows) present ambiguity somewhat different from that in the auxiliary problems—for example, the parameters for the words `code` and `computer` are positively correlated in the general 20 newsgroups dataset, but should probably have negative prior covariance for the task "Cryptography" vs. "Graphics."

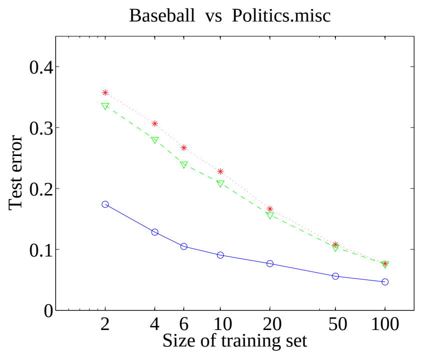
Finally, we tested transfer to an entirely different dataset. We created 50 random text classification problems using webpages under the DMOZ Open Directory Project hierarchy,[4] with 10 problems each from the Arts, Business, Health, Recreation and Sports cat-

[3] The WPMI between word w_i and label k is defined as: $P(w_i \text{ occurs, label is } k)\log_2 \frac{P(w_i \text{ occurs, label is } k)}{P(w_i \text{ occurs})P(\text{label is } k)}$ where the probabilities are over the choice of a random document from the dataset. Note that maximizing mutual information between the involved random variables would not give label-specific words; maximizing pointwise mutual information alone would be similarly unsatisfactory as it would generally prefer very rare words.

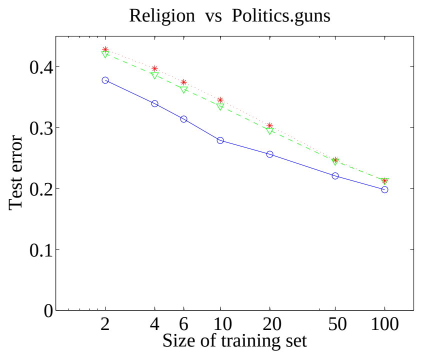
[4] `http://dmoz.org`



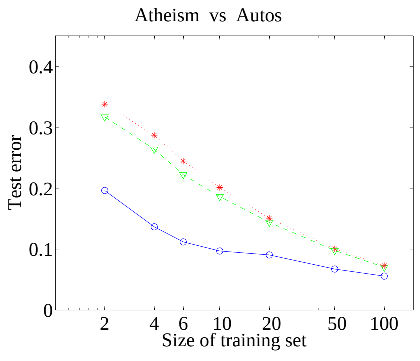
(a)



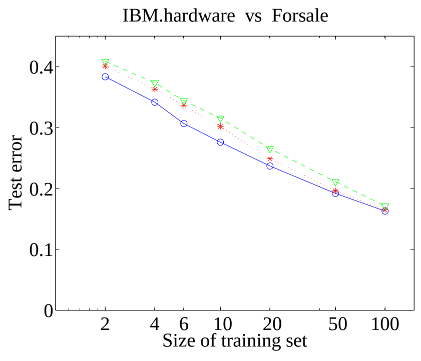
(b)



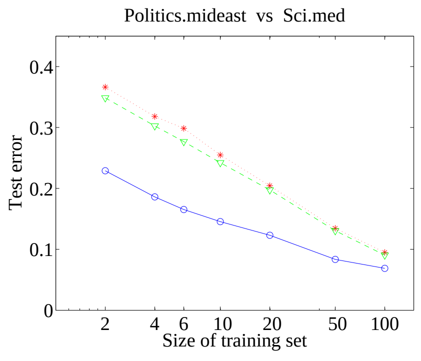
(c)



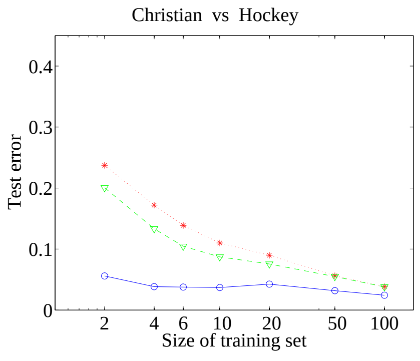
(d)



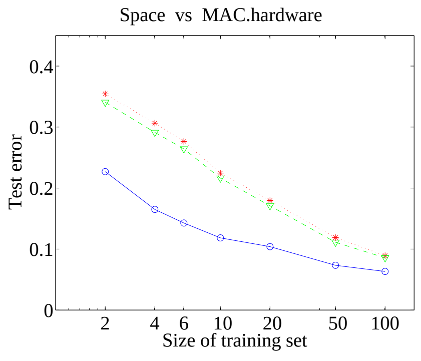
(e)



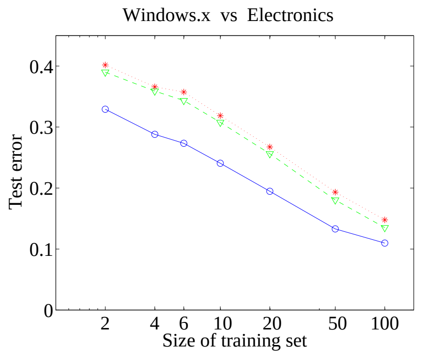
(f)



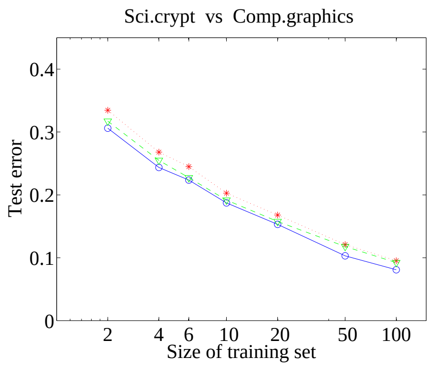
(g)



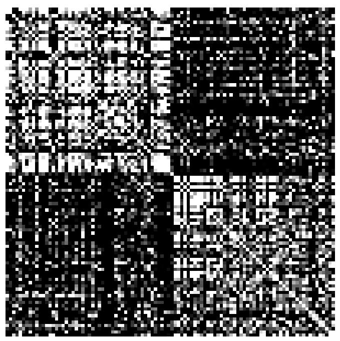
(h)



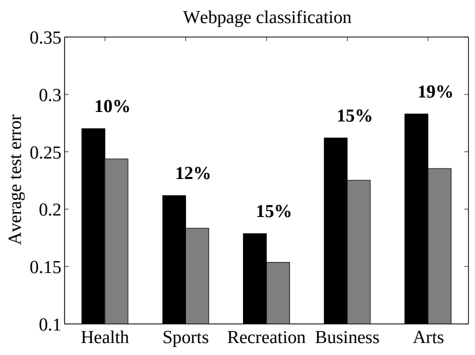
(i)



(j)



(k)



(l)

Figure 1. (a-j) 20 newsgroups results. Training set size is plotted on a log-scale. Each plot shows test error for a target problem for varying training set size. Blue circles are for our SDP-based method, green triangles for SDP with the diagonal covariance constraint, red stars for the baseline diagonal prior. [Colors where available.] (k) Graphical depiction of an SDP-generated covariance matrix.[See text.] (l) Performance on a DMOZ webpage classification task using the 20 newsgroups dataset as auxiliary data. Each group represents the average over 10 problems from a DMOZ subcategory; the subcategories are sorted from left to right in increasing order of fraction of direct covariance estimates used. (E.g., Health:1.1%, Arts:2.8%.) The black bars represent average test error for the uniform diagonal covariance baseline; the gray bars are for the SDP-generated covariance matrices. The avg % error reduction over baseline is listed per subcategory.

Table 3. Classification tasks achieving the most (top two rows) and least (bottom two rows) boost in performance because of the transfer learning setup.

Most transfer	`Christian` vs. `Hockey`
	`Atheism` vs. `Autos`
Least transfer	`IBM.hardware` vs. `Forsale`
	`Sci.crypt` vs. `Comp.graphics`

egories. For each classification problem, we used a vocabulary of 400 words and a training set of 10 documents. Using the 20 newsgroups dataset as auxiliary data, we estimated only 1-3% of all covariance entries using Algorithm 1, and solved the SDP (4) to fill in the covariance matrix. We fixed the scaling parameter q to the average value from the earlier experiments ($q = 3$). Figure 1(l) shows the test error for problems within each subcategory, averaged over the 10 problems and several training sets for each problem. The full covariance matrix reduces test error substantially over the baseline; the error reduction is higher when more covariance estimates are used.

7. Related Work

The initial foundations for transfer learning were laid by Thrun (1996), Caruana (1997) and Baxter (1997), among others. Several authors have since provided theoretical justification for transfer learning (Ben-David & Schuller, 2003; Ando & Zhang, 2005).

In this paper, we have presented an algorithm for constructing the covariance matrix for an informative Gaussian prior. The algorithm uses other "similar" learning problems to learn a good underlying mapping from word pair features to word parameter covariances. Ando & Zhang (2005) use a different setup to learn the properties of good classifiers from multiple learning tasks.

A different hierarchical Bayesian viewpoint might assume that the auxiliary learning problems arise from the same prior distribution as the target problem. In that setting, the prior covariance matrix is a hyperparameter in the model, and can be learnt using auxiliary learning problems. To deal with novel words, such a method could pose a generative model using hyperparameters (similar to our ψ parameters), and learn them instead. Several authors have used hierarchical Bayesian modeling to propose multi-task learning algorithms. For example, Lawrence & Platt (2004) and Yu et al. (2005) use multiple learning problems to learn the parameters of a Gaussian process.

The proposed algorithm shows promising transfer learning results using a small number of auxiliary problems. It is possible to transfer "knowledge" from newsgroup data to webpage data; this opens the possibility of training a single generic covariance matrix for frequently used English words, and then reusing it on a broad class of English text classification problems. Also, while our experiments have focused on text data, similar models can be used in other classification settings where correlations between individual parameters are likely, and can be usefully predicted by observable features of the parameters.

Acknowledgments

We give warm thanks to Chuong Do for providing the DMOZ data, and Penka Markova for useful comments. This work was supported by the DARPA transfer learning program under contract number FA8750-05-2-0249.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Baxter, J. (1997). A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, *28*, 7–39.

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for multiple task learning. *COLT*.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Chung, F. (1997). Spectral graph theory. *Regional Conference Series in Mathematics, American Mathematical Society*, *92*, 1–212.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. In *The Annals of Statistics*, vol. 7, 1–26.

Lang, K. (1995). Newsweeder: learning to filter netnews. *ICML*.

Lawrence, N. D., & Platt, J. C. (2004). Learning to learn with the informative vector machine. *ICML*.

Miller, G. A. (1995). Wordnet: A lexical database for English. *Commun. ACM*, *38*, 39–41.

Ng, A. Y., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *NIPS*.

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. *IJCAI Workshop on Machine Learning for Information Filtering*.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *NIPS*.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. *ICML*.