Transfer learning by constructing informative priors

Rajat Raina
Computer Science Department
Stanford University
Stanford, CA 94305
rajatr@cs.stanford.edu

Andrew Y. Ng
Computer Science Department
Stanford University
Stanford, CA 94305
ang@cs.stanford.edu

Daphne Koller
Computer Science Department
Stanford University
Stanford, CA 94305
koller@cs.stanford.edu

Abstract

Many applications of supervised learning require good generalization from limited labeled data. In the Bayesian setting, we can use an informative prior to try to achieve this goal by encoding useful domain knowledge. We present an algorithm that constructs such an informative prior for a given discrete supervised learning task. The algorithm uses other "similar" learning problems to discover properties of optimal classifiers, expressed in terms of covariance estimates for pairs of feature parameters. A semidefinite program is then used 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% error reduction over a commonly used prior.

1 Introduction

For conciseness and clarity, we restrict our attention to binary text classification, although our model can be extended to handle other discrete multiclass classification problems.

In this task, we are given a vocabulary of words $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ and a set of output labels $\mathcal{Y} = \{0, 1\}$. We assume that every input document X is represented as a vector $X = (X_1, X_2, \ldots, X_{|\mathcal{W}|}) \in \{0,1\}^{|\mathcal{W}|}$, where X_i is 1 if w_i occurs in X and 0 otherwise. Each document X has a unique label $Y \in \mathcal{Y}$. A labeled training set $M = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$ is given, and the task is to predict label y for an unseen input instance x. Throughout this paper, we use logistic regression as the basic prediction model. Logistic regression uses a parameter vector $\theta = (\theta_1, \theta_2, \ldots, \theta_{|\mathcal{W}|}) \in \Re^{|\mathcal{W}|}$ to make predictions according to the rule: $P(Y = 1|X = x, \theta) = 1/\left(1 + \exp(-\theta^T x)\right)$. A common learning algorithm assumes a multivariate Gaussian prior $\mathcal{N}(0, \sigma^2 I)$ on the parameter θ, and then finds the maximum-a-posteriori (MAP) estimate θ_{MAP} by maximizing the (penalized) discriminative log-likelihood of the training set M:

$$\theta_{MAP} \quad = \quad \arg\max_\theta \left(\sum_{i=1}^{m} \log P(Y = y^{(i)}|X = x^{(i)}, \theta) - \frac{1}{2\sigma^2}||\theta||_2^2 \right) \tag{1}$$

When training data is limited ($m << |\mathcal{W}|$), the parameter values learnt often produce low 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 coherently 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 `space` 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

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

Table 1: Word pairs from the classification problem `rec.motorcycles` versus `comp.os.ms-windows.misc` that were estimated to have the most positive (left) or most negative (right) bootstrap-corrected parameter covariance using auxiliary learning problems.

prior $\mathcal{N}(0, \Sigma)$ where $\Sigma \in \Re^{|\mathcal{W}| \times |\mathcal{W}|}$ is a (non-diagonal) covariance matrix. For our example, if the prior covariance between the parameters for `moon` and `space` is highly positive, we can infer that `space` supports the label y even without observing this directly in training data.

2 Estimating the covariance between word parameters

We will first describe a method to estimate individual entries of this covariance matrix Σ using other labeled text classification problems, which we shall call the *auxiliary* learning problems. Let C be one of these auxiliary problems that also contains a word $w_i \in \mathcal{W}$ in its vocabulary. We can use C to estimate the prior variance $\mathrm{Var}(\theta_i)$ as follows: We generate several subproblems from C by sampling several smaller vocabularies (of size 5, say) each containing the word w_i, and apply logistic regression to each of these subproblems.[1] Each subproblem produces a final learnt value for parameter θ_i; the sample variance of these values gives an estimate for $\mathrm{Var}(\theta_i)$. However, since we are interested in estimating the parameter variance caused only by the word appearing in different problems, and not the variance caused by the choice of training set, we correct the computed variance estimate using a bootstrap correction (see, e.g., [1]). The prior covariances $\mathrm{Cov}(\theta_i, \theta_j)$ can be similarly estimated using the bootstrap from an auxiliary problem containing both words w_i and w_j.

For a particular classification problem constructed using the 20 newsgroups dataset [2], Table 1 shows some word pairs whose covariance was found to be the most positive or most negative when estimated using auxiliary problems constructed from the other 18 newsgroups. Words related to similar topics are estimated to have a highly positive covariance by this method; some of the word pairs with highly negative covariance entries also capture useful distinctions between the classes.

3 Constructing the covariance matrix Σ

In a perfect world, we could have constructed the desired covariance matrix Σ by putting together the above covariance estimates for all word pairs. However, this approach has two major problems:

1. Some word pairs from the current vocabulary $\mathcal{W}$ may not occur in any auxiliary task vocabularies. For a large vocabulary $\mathcal{W}$ it is also computationally impractical to generate all the $|\mathcal{W}|(|\mathcal{W}|+1)/2$ covariance estimates needed to construct Σ.
2. A valid covariance matrix must be positive semidefinite (PSD). The matrix formed by the noisy covariance estimates may not be PSD.

To solve the first problem, we model covariance between a pair of parameters as a function of *features* of the corresponding word pair. For example, if the words are synonyms of each other, there might be a high covariance between their parameters; so checking if the words are synonyms might be a useful feature. Given these features, we can estimate a small fraction of the covariances using the auxiliary classification tasks, and then learn a general transformation of the features that allows us to approximate all the missing entries in the covariance matrix.

More formally, suppose we extract a feature vector $F_{ij} \in \Re^S$ for each position (i, j) in the covariance matrix, such that every element of F_{ij} computes some feature of the word pair (w_i, w_j) and the vocabulary $\mathcal{W}$. We approximate each entry in the covariance matrix as a linear function of the corresponding feature vector–i.e., given a suitable parameter vector $\psi \in \Re^S$, we construct a candidate matrix $\hat{\Sigma}$ by computing its $(i, j)^{th}$ element as follows:

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

[1] The small vocabulary size ensures that each parameter estimate is close to optimal.

Let the set $G = \{(i,j) \mid \mathrm{Cov}(w_i, w_j)\text{was estimated}\}$ contain the positions of the available covariance estimates from the auxiliary tasks, and e_{ij} be the value of the estimate for position $(i,j) \in G$. We could then pick ψ so that the generated values $\hat{\Sigma}_{ij}$ are close to the available estimates e_{ij}, using linear regression, for example.

Unfortunately, the matrix $\hat{\Sigma}$ may not be PSD. We could project $\hat{\Sigma}$ onto the PSD cone to obtain a valid covariance matrix; however, $\hat{\Sigma}$ often turns out to highly indefinite, and the projected matrix is far from the matrix $\hat{\Sigma}$ originally learnt through ψ. This sequential procedure is unnecessary. If Σ is the final covariance matrix produced, we can make the PSD projection explicit by posing the following joint optimization problem over variables ψ and Σ:

$$\min_{\psi,\Sigma} \sum_{(i,j)\in G} (e_{ij} - \psi^T f_{ij})^2 + \lambda \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\mathcal{W}|} (\Sigma_{ij} - \psi^T f_{ij})^2 \qquad \text{s.t.} \quad \Sigma \succeq 0 \tag{3}$$

where Σ_{ij} denotes the $(i,j)^{th}$ element of matrix Σ. The first term in the objective function encourages ψ to better approximate the available covariance estimates e_{ij}; the second term encourages ψ to generate a matrix close to a PSD matrix Σ; λ represents the relative importance assigned to these two terms.

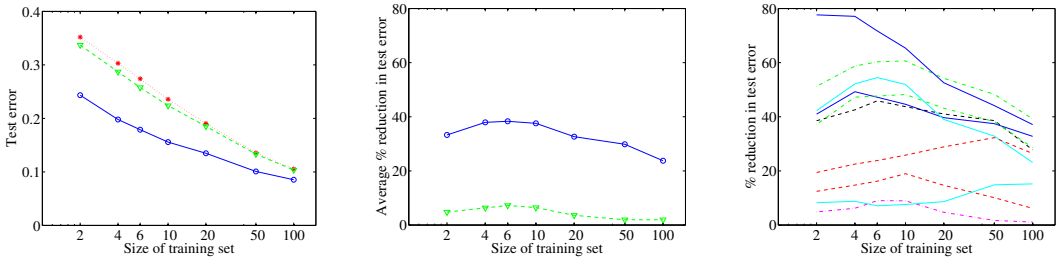
This optimization problem can be written as a semidefinite program (SDP), and can be solved directly using standard SDP solvers. A more scalable method is to perform optimization by coordinate-descent on ψ and Σ. Keeping Σ fixed, ψ can be optimized using a fast QP solver; keeping ψ fixed, Σ can be optimized via a fast projection onto the PSD cone. This alternating optimization method is guaranteed to find the global minimum.

In our experiments, we observed that the matrix Σ 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 particular small vocabulary size (5 in our case), while our experiments consider covariances with a much larger vocabulary size. We thus multiply in a scaling parameter q, and use $q\Sigma$ as the final covariance matrix in the prior. We use the auxiliary problems to search for a good value of q via a simple multiplicative coordinate-ascent procedure that locally optimizes likelihood of training sets drawn for those auxiliary problems.

4 Results

We present our results on the 20 newsgroups dataset [2]. The data was preprocessed by stemming and removing stopwords. The newsgroups were randomly paired to construct 10 binary classification problems. The vocabulary for each pair was constructed by picking the 250 most frequent words from each newsgroup. We then performed a hold-out procedure, taking each of these 10 problems in turn and letting the other 9 be auxiliary problems; covariance estimates were generated from these auxiliary problems[2] and were used to solve the SDP for the held-out problem; a scaling parameter was estimated using the auxiliary problems, and the SDP-solution with the scaling parameter was evaluated on the held-out problem using several randomly sampled balanced training sets. The class labels on the held-out problem were hidden throughout. The final test error was averaged over all these hold-out iterations. We compare against a baseline that uses a uniform diagonal prior, with a scaling parameter that was chosen using exactly the same procedure as for the SDP-generated covariance matrices. To verify the advantage of using a non-diagonal covariance, we also solved Eqn. 3 constraining Σ to be diagonal. Figure 1 shows the final results. The SDP-generated covariance matrices reduce error by 20-40% over the baseline; they continue to be better than the baseline even for a training set of 100 documents. In fact, on all 10 tests, the SDP-result were better than baseline over the full range of training set sizes. The SDP-generated diagonal covariance matrix performs only marginally better than baseline, showing that the off-diagonal entries capture crucial dependencies.

[2] In each case, we used the auxiliary problems to get covariance estimates for about 75% of the diagonal entries and 20% of the off-diagonal entries. About 25% of the words in each vocabulary do not occur in any auxiliary problem, and are "novel." Some features used WordNet (`http://wordnet.princeton.edu`) and Infomap (`http://infomap.stanford.edu`). For optimization, we used the SeDuMi (`http://sedumi.mcmaster.ca`) and Yalmip (`http://control.ee.ethz.ch/~joloef/yalmip.php`) packages.



(a) (b) (c)

Figure 1: Classification results on the 20 newsgroups dataset. Training set size is graphed on a log-scale. (a) Average test error for different training set sizes. Blue circles are for our SDP-based method, green triangles for SDP with the diagonal covariance constraint, red stars for the baseline diagonal prior. (b) Average percentage reduction in test error over the baseline: Blue circles are for the SDP-based method, green triangles for SDP with the diagonal covariance constraint. (c) Percentage reduction in test error over the baseline for each of the 10 runs of the SDP-based method. [Colors where available.]

Figure 2: Graphical depiction of the final SDP-generated covariance matrix generated for the classification problem `rec.motorcycles` vs. `comp.os.ms-windows.misc`. [Description in text.]

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.[3] From the full covariance matrix, we then extracted the rows and columns corresponding to these words only, and arranged them in a matrix so that all words picked from a class are together. The left half of Figure 2 shows this matrix, with brighter positions representing higher values. On the right, Figure 2 shows the same matrix with all entries above the median entry colored white and the rest black. These figures show a rough block structure, demonstrating that the learnt covariance matrix is able to discover good word dependencies for the classification task at hand *without* any knowledge of the labels.

5 Related and Future Work

Our algorithm is most similar in spirit to [3], where the authors use a different setup to learn the properties of good classifiers from multiple learning tasks. We are currently working on a method that views the prior covariance as a hyperparameter in a hierarchical Bayesian model and directly learns it using auxiliary learning problems.

References

[1] B. Efron. Bootstrap methods: Another look at the jackknife. In *The Annals of Statistics*, volume 7, pages 1–26. 1979.

[2] Ken Lang. Newsweeder: learning to filter netnews. In *Proceedings of the Ninth European Conference on Machine Learning*, 1997.

[3] Rie K. Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. Technical Report RC23462, IBM T.J. Watson Research Center, 2004.

[3] We picked words with the highest weighted pointwise mutual information with each class label.