Learning a Meta-Level Prior for Feature Relevance from Multiple Related Tasks

Su-In Lee SILEE@CS.STANFORD.EDU
Vassil Chatalbashev VASCO@CS.STANFORD.EDU
David Vickrey DVICKREY@CS.STANFORD.EDU
Daphne Koller KOLLER@CS.STANFORD.EDU
Computer Science, Stanford University, Stanford, CA 94305

Abstract

In many prediction tasks, selecting relevant features is essential for achieving good generalization performance. Most feature selection algorithms consider all features to be a priori equally likely to be relevant. In this paper, we use transfer learning — learning on an ensemble of related tasks — to construct an informative prior on feature relevance. We assume that features themselves have meta-features that are predictive of their relevance to the prediction task, and model their relevance as a function of the meta-features using hyperparameters (called *meta-priors*). We present a convex optimization algorithm for simultaneously learning the meta-priors and feature weights from an ensemble of related prediction tasks that share a similar relevance structure. Our approach transfers the meta-priors among different tasks, allowing it to deal with settings where tasks have non-overlapping features or where feature relevance varies over the tasks. We show that transfer learning of feature relevance improves performance on two real data sets which illustrate such settings: (1) predicting ratings in a collaborative filtering task, and (2) distinguishing arguments of a verb in a sentence.

1. Introduction

In many prediction tasks, we are faced with a huge number of features. The use of effective feature selection algorithms or regularization method is critical for achieving good performance. Much effort has been devoted to the topic of feature selection, and many approaches have been proposed (see Kaelbling (2003) for examples). Most feature selection algorithms treat all candidate features as equally likely, *a priori*, to be relevant, and use the data alone to select among them. In many cases, however, we may have reason to believe that some features may innately be more or less likely to be relevant. For example, consider a collaborative filtering task, where we predict a user's preference for one product p based on his observed preferences for others. Here, the preferences for certain products — e.g., those similar or related to p — are more likely to be relevant to p than others. In another example, consider inferring protein function from a set of motifs (short segments) in its sequence. Here, conserved motifs, or those on the surface of the protein, are more likely to be relevant.

In these examples, and in many others, a feature k can be characterized by a set of *meta-features* $\boldsymbol{f}_k$ that describe both the properties of the feature and its potential relationship to the prediction problem. In the collaborative filtering example, a product serving as a feature may be characterized by meta-features such as its price range or its similarity (along different dimensions) to the target product. In the protein prediction example, a motif may be characterized by meta-features such as its conservation score or the composition of hydrophilic amino acids. We may wish to utilize these meta-features to construct a more informed prior over feature relevance. In most cases, however, we do not have enough prior knowledge to determine exactly how much effect, or even in which direction, each of these meta-features has on feature relevance. Thus, a key problem is to learn the hyperparameters (a meta-level prior) that characterize our prior on the relevance of a feature in terms of its meta-features. While cross-validation is often used to estimate meta-parameters such as these, the number of meta-features in many applications can be quite large, rendering a standard cross-validation regime intractable.

In this paper, we describe a *transfer-learning* approach for estimating these hyperparameters. In the transfer learning paradigm, first proposed by Baxter (1997); Caruana (1997); Thrun (1996), one aims to achieve better general-

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

ization by considering multiple related learning tasks, and transferring information among them. Here, we use an ensemble of prediction tasks where similar types of features tend to be relevant. Although information about the actual values of parameters is commonly transferred in this setting, the problem of learning informed models of feature relevance has been largely unexplored; see Sec. 6 for a discussion of prior work.

Our approach applies to the class of generalized linear models (McCullagh & Nelder, 1989), where the predicted target is a function of a weighted linear combination of the features. Typically, all features are assumed to have a zero mean Gaussian prior with the same variance. In our case, we take the prior variance to be a weighted linear combination of the feature's meta-features, where the weights are the model hyperparameters (called meta-priors). These meta-priors are shared both across features and across tasks, allowing transfer of learned relevance information across different features, both within a task and between tasks. Thus, our approach can handle situations where tasks have non-overlapping features or where feature relevance can vary over the tasks. The objective function used in our formulation is jointly convex in the feature weights in each task and the meta-priors. Moreover, we show that two well-known feature selection methods, i.e., L_1-regularization (Tibshirani, 1996) and group Lasso (Yuan & Lin, 2006), are special cases of our general approach.

We apply our method to two real world data sets. The first is a collaborative filtering task, where we predict user ratings for movies in (a subset of) the Netflix data. The second is the natural language task of *semantic role labeling* (Gildea & Jurafsky, 2002), where we aim to identify which words in the sentence correspond to which semantic argument of a verb (e.g., for the verb "throw" who is the thrower and what was thrown). We show that, by tying together the feature selection decisions in our ensemble of tasks, we obtain better generalization to unseen test data in all of them. Even more interestingly, we show that our method also allows us to transfer prior knowledge on feature relevance, in the form of $\boldsymbol{\beta}$, to new prediction tasks, allowing us to achieve significantly better performance with small amounts of training data.

2. Transfer Learning Formulation

Our problem formulation assumes the existence of an ensemble of R supervised learning tasks. Each task $r = 1, \ldots, R$ is associated with a *response variable* y_r and a set of K_r features, denoted $x_{r1}, \ldots, x_{rK_r}$. The features themselves may or may not be shared among the prediction problems. We use $\boldsymbol{x}_r$ to denote the feature vector associated with problem r.

We aim to learn the parameters of a probabilistic model that defines $P(y_r \mid \boldsymbol{x}_r)$. Our framework applies to the class of generalized linear models, where this conditional distribution is defined in terms of $g(\boldsymbol{w}_r^\top \boldsymbol{x}_r)$, for some vector of task-r parameters $\boldsymbol{w}_r = (w_{r1}, \ldots, w_{rK_r}) \in \mathbb{R}^{K_r}$ and some pre-selected gating function g. For example, we can consider the case where the y_r's are discrete binary-valued variables and g is the logit function:

$$P(y_r = 1 \mid \boldsymbol{x}_r, \boldsymbol{w}_r) = \frac{1}{1 + \exp(-\boldsymbol{w}_r^\top \boldsymbol{x}_r)}, \qquad (1)$$

where, for simplicity of notation, we ignore the bias (intercept) term, under the assumption that one of the features is always set to 1. Alternatively, y_r could be continuous with a Gaussian distribution whose mean is $\boldsymbol{w}_r^\top \boldsymbol{x}_r$. We can also consider the case where y_r is continuous and its distribution is a linear Gaussian whose mean is $\boldsymbol{w}_r^\top \boldsymbol{x}_r$.

As is typically done, we associate each feature weight w_{rk} with a a Gaussian prior with mean zero and variance γ: $P(w \mid \gamma) = \frac{1}{\sqrt{2\pi\gamma}} \exp(-\frac{w^2}{2\gamma})$. In most applications, all parameters in a model are taken to have the same prior, encoding a similar bias towards 0. In our setting, we allow a different γ_{rk} for each weight w_{rk}. In order to achieve effective generalization performance, we model γ_{rk} as a weighted linear combination of meta-features of the feature x_{rk}. More precisely, we assume that each feature x_{rk} is associated with a meta-feature vector $\boldsymbol{f}_{rk} \in \mathbb{R}^\ell$, which encodes certain characteristics of the feature that may be predictive of its relevance to the prediction task. The meta-features may depend either on the feature alone, or on the feature *and* on the prediction task.

Given a meta-feature vector $\boldsymbol{f}_{rk}$, we take the prior $P(w_{rk})$ to be a Gaussian distribution with variance $\gamma_{rk} = \boldsymbol{\beta}^\top \boldsymbol{f}_{rk}$ (constrained to be positive), where $\boldsymbol{\beta}$ is the set of *model hyperparameters*:

$$P(w_{rk} \mid \boldsymbol{\beta}, \boldsymbol{f}_{rk}) = \frac{1}{\sqrt{2\pi\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}}} \exp(-\frac{w_{rk}^2}{\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}}). \qquad (2)$$

We also define a prior distribution over the variance γ_{rk} $(= \boldsymbol{\beta}^\top \boldsymbol{f}_{rk})$ to be the gamma distribution:

$$P(\gamma_{rk}) \propto \gamma_{rk}^{D-1} \exp(-C\gamma_{rk}) \quad \gamma_{rk} \geq 0, \qquad (3)$$

for constants C and D whose selection we discuss below. This prior distribution over the γ_{rk}'s serves to bias their values towards zero, which helps prevent the overfitting that can occur when the variance of the weight prior is too high. We note that the meta-features should be chosen so that the feasible set $\{\boldsymbol{\beta} : \forall r, k \quad \boldsymbol{\beta}^\top \boldsymbol{f}_{rk} > 0\}$ is nonempty. We can guarantee this feasibility condition by restricting the meta-features to have non-negative values or by adding a "bias" meta-feature with a large enough positive value.

Now, consider a data set $\boldsymbol{X}, \boldsymbol{Y}$ consisting of M training instances for each of our R tasks (where we assume, purely for simplicity of notation, that all tasks have the

same number of instances). Thus, $\boldsymbol{X}$ is a set of vectors $\boldsymbol{x}_r[1], \ldots, \boldsymbol{x}_r[M]$, for $r = 1 \ldots, R$, where each $\boldsymbol{x}_r[m] \in \mathbb{R}^{K_r}$; and $\boldsymbol{Y}$ is a set of responses $y_r[1], \ldots, y_r[M]$. Denoting by $\boldsymbol{W}$ the vector of all of the parameters $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_R$, overall we can define a joint conditional distribution:

$$P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\beta} \mid \boldsymbol{X}, \mathbf{F}) = \prod_{r=1}^{R} \prod_{m=1}^{M} P(y_r[m] \mid \boldsymbol{x}_r[m], \boldsymbol{w}_r) \cdot \prod_{r=1}^{R} \prod_{k=1}^{K_r} P(w_{rk} \mid \boldsymbol{\beta}, \boldsymbol{f}_{rk}) P(\gamma_{rk}), \quad (4)$$

where $\gamma_{rk} = \boldsymbol{\beta}^\top \boldsymbol{f}_{rk}$. Here, the first term is our generalized linear model, and the two terms in the second product are defined in Eq. (2) and Eq. (3).

3. Optimization Algorithm

We choose to address our learning problem by finding the joint *maximum a posteriori (MAP)* assignment to all of the parameters $\boldsymbol{w}_r$, $r = 1, \ldots, R$ and the hyperparameters $\boldsymbol{\beta}$, in the objective Eq. (4). To simplify the objective function, we fix D in Eq. (3), so that γ_{rk}^{D-1} in Eq. (3) cancels out $1/\sqrt{2\pi\gamma_{rk}}$ in Eq. (2). With this assignment, we take the logarithm of Eq. (4), and obtain a joint log-likelihood function:

$$\begin{aligned} &\log P(\boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\beta} \mid \boldsymbol{X}, \mathbf{F}) \quad (5) \\ &= \sum_{r=1}^{R} \sum_{m=1}^{M} \log P(y_r[m] \mid \boldsymbol{x}_r[m], \boldsymbol{w}_r) \\ &- \sum_{r=1}^{R} \sum_{k=1}^{K_r} \left(\frac{w_{rk}^2}{\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}} + C\boldsymbol{\beta}^\top \boldsymbol{f}_{rk} + \textit{Const} \right), \end{aligned}$$

where *Const* does not depend on the optimization parameter. The other hyper-parameter, C, will be estimated using cross-validation.

Critically, this objective function is jointly convex over the optimization variables $\boldsymbol{W}$ and $\boldsymbol{\beta}$. Therefore, it can be solved using a range of efficient convex optimization algorithms, any of which is guaranteed to find the unique global optimum. We choose to use a coordinate ascent procedure over the two sets of parameters $\boldsymbol{W}$ and $\boldsymbol{\beta}$.

For optimizing $\boldsymbol{W}$ given the current $\beta^{(t)}$, we solve:

$$\arg\min_{\boldsymbol{W}} \sum_{r=1}^{R} \sum_{m=1}^{M} -\log P(y_r[m] \mid \boldsymbol{x}_r[m], \boldsymbol{w}_r) + \sum_{r=1}^{R} \sum_{k=1}^{K_r} \frac{w_{rk}^2}{\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}}. \quad (6)$$

This equation has the same form as a generalized linear model with a weighted L_2 regularization penalty, and can be solved using standard gradient methods (or factorization methods in the case of linear regression).

To optimize $\boldsymbol{\beta}$ given $\boldsymbol{W}^{(t)}$, we solve:

$$\arg\min_{\boldsymbol{\beta}} \sum_{r=1}^{R} \sum_{k=1}^{K} \left(\frac{{w_{rk}^{(t)}}^2}{\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}} + C\boldsymbol{\beta}^\top \boldsymbol{f}_{rk} \right) \quad (7)$$
$$\text{subject to } \boldsymbol{\beta}^\top \boldsymbol{f}_{rk} > 0.$$

This objective is convex in $\boldsymbol{\beta}$, and can therefore be optimized efficiently using standard methods.

4. Sparse Norm Equivalence

A certain special case of our approach is equivalent to a standard L_1-regularization (Tibshirani, 1996), and another to group Lasso regularization (Yuan & Lin, 2006). As a reminder, group Lasso regularization partitions the set of features into mutually exclusive and exhaustive sets $\mathcal{F}_n$, for $n = 1, \ldots, N$. The regularization term then takes the form $\sum_n \sqrt{\sum_{i \in \mathcal{F}_n} w_i^2}$. This has the effect of driving the parameters in each group towards 0 *together*.

Group Lasso arises as a special case in our framework if we associate a single shared meta-feature for each group of features $\mathcal{F}_n$, which (for simplicity) takes the value 1. Letting $\boldsymbol{\beta}_n$ be the meta-prior corresponding to $\mathcal{F}_n$, the weight prior for the features from Eq. (5) is now $\sum_n \sum_{(r,k) \in \mathcal{F}_n} (\frac{w_{r,k}^2}{\beta_n} + C\beta_n)$. Applying the optimality condition for $\boldsymbol{\beta}$, we can show that the above weight prior term is equivalent to: $\sum_n 2\sqrt{C|\mathcal{F}_n|} \sqrt{\sum_{(r,k) \in \mathcal{F}_n} w_{r,k}^2}$. In the extreme case, when each feature is modeled with a unique identity meta-feature, i.e. $|\mathcal{F}_n| = 1$ for all n, the prior reduces to a standard L_1-regularization.

5. Experimental Results

We train our model over an ensemble of R prediction tasks, jointly learning both the weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_R$ and the hyper-parameters $\boldsymbol{\beta}$. We then evaluate the results relative to two different types of learning setups. In the **Gen-Test** setup, we evaluate the generalization of the learned weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_R$ to new instances from the training tasks. This setup tests the ability of our approach to utilize the meta-level information on the features in order to select more relevant, better generalizing features. In the **Trans-Test** setup, we use the $\boldsymbol{\beta}$ learned on these R tasks as a prior for new, previously unseen tasks. We then learn only $\boldsymbol{W}$ on each new *test task*, using $\boldsymbol{\beta}$ as a prior, and evaluate generalization performance on new test-task instances. This setup evaluates whether information learned from previous learning tasks can be used to allow learning of new tasks using a lot less data.

5.1. Collaborative Filtering

We applied our algorithm to the task of collaborative filtering for movie ratings. Specifically, we considered the problem of predicting ratings assigned to movies by viewers in the Netflix movie rating dataset.The full dataset consists of nearly 100 million discrete ratings from 1 to 5 that

480189 users assigned to 17770 movies. For our experiments, we selected the 5000 users with the highest number of ratings, as well as the 600 movies with the highest number of ratings, and normalized the ratings for each movie so that they have 0 mean and unit variance.

As often done in collaborative filtering (Marlin, 2004), we view each movie as a separate prediction task, in which the instances are users, and the features are the ratings that the user has assigned to other movies that he has seen. More precisely, in the prediction task for movie m, user u's ratings are mapped to a feature vector $\boldsymbol{x}_m[u]$, in which the ith entry is either the (normalized) rating the user gave to the ith movie, or 0 if the user has not rated the ith movie which is equivalent to treating a missing rating as the mean rating over the movie since movie ratings have mean 0. The response variable is the rating $y_m[u]$ that the user gave to movie m. We performed feature selection for each prediction task by discarding feature movies whose ratings have a Pearson correlation coefficient with absolute value of less than 0.20 with the target movie's ratings.

We use a linear regression model: $y_m \sim N(\boldsymbol{w}_m^\top \boldsymbol{x}_m, \epsilon)$. For our baseline model, we add a Gaussian prior over the weights $\boldsymbol{w}_m$ with mean 0 and variance γ, resulting in a standard ridge regression model that treats all features as equally likely to be relevant (we also experimented with L_1-regularized regression, but the performance was slightly worse, so we omit those results). While our feature representation ignores the information present in the mere presence of movie ratings, in our experience with the full Netflix dataset, the ridge regression model is competitive with other collaborative filtering approaches such as memory-based methods and generative models. Note, however, that we do not aim to compete with all state-of-the-art collaborative filtering algorithms, but rather to evaluate our approach on this real-world data set.

Table 1. Meta-features used in our collaborative filtering domain

1.**Genre:** Whether the movies share a particular genre.
2.**Decade:**Whether both movies were released in 2000s, 1990s, 1980s, 1970s or a previous decade.
4.**Actors/Directors:** How many and which actors/directors the movies have in common.
5.**Keywords:** How many and which keywords the movies have in common.

In this domain, the features for rating one movie m are ratings for other movies m'. Intuitively, the rating for m' is more likely to be relevant to predicting m when m' and m are similar, in some sense. To capture this intuition, we model the relevance of features using meta-features that are based on shared attributes of the feature movie and prediction task movie. We associated each Netflix movie title with a corresponding entry in the Internet Movie Database (IMDB),[1] and extracted a rich set of attributes: movie genre, actors, director, and many descriptive keywords about the movie (such as "cult movie", "violence", "organized crime" etc.). We discarded actors with fewer than 25 movies in the full Netflix dataset, directors with fewer than 5 movies, and keywords which appear in fewer than 80 movies. We defined a meta-feature vector $\boldsymbol{f}_{mm'}$ for each prediction movie m and feature movie m' pair, as shown in Table 1. Notice that these meta-features represent properties of the *combination* of a feature and a prediction task movie, and thus illustrate the ability of our model to have a feature's relevance vary depending on the prediction task. Altogether, this set consists of around 1400 meta-features, from which selected the 200 most correlated with the magnitude of the weights of a ridge regression baseline model learned over a random subset of the training data. In addition, for each task we also include a unique "bias" meta-feature whose value is always 1.

In our **Gen-Test** experiment, we used all 600 movies in our dataset as tasks, each with a training set of fixed size M (M ranging from 100 to 2000). We split the remaining ratings for each movie into a validation set, consisting of 10% of the ratings, and a test set. We trained the feature weights $\boldsymbol{W}$ and meta-feature weights $\boldsymbol{\beta}$ on the training set, and used the validation set to select the optimal ridge penalty parameter for the baseline model. The reported results are averaged over 5 trials. Our measure of error is root mean squared error (RMSE).

The meta-feature weights learned by the model are intuitive: the meta-features with the highest weights are those corresponding to specific popular actors such as Adam Sandler, Arnold Schwartzeneger and many others. Certain directors such as Quentin Tarantino and Richard Donner also have meta-features with high weights suggesting that viewers who liked some of their movies tend to like all. From the shared descriptive keyword meta-features, some of the ones with the highest weights are whether the two movies contain vulgarity or whether they involve sword-fights or warriors. Among the shared genre meta-features the most predictive ones are the Sports and Musical genres.

From this discussion, it is clear that not all prediction tasks in this domain are associated with high-weight meta-features. For example, when a prediction task movie does not have any of these significant actors, directors, or keywords, its associated meta-features are likely to have fairly low weights. In this case, the range of relevance weights for different features will be fairly narrow, giving rise to an undifferentiated relevance prior. Indeed, Figure 1 shows the relative RMSE improvement the meta-prior model gives over the baseline model (when training over 100 examples), as a function of the range of learned relevance of its features. When the learned relevance of the features lies in a narrow range, we see little improvement. Importantly, however, the meta-level prior model is robust and never leads to a significant reduction in RMSE. When the fea-

[1] Downloadable in text format from `www.imdb.com`.

ture relevance range is wider, we see that the improvement of the model is most dramatic, reaching as high as 10% for some movies. We therefore focused the remainder of our evaluation to those tasks which exhibit an average feature relevance range greater than 0.005.

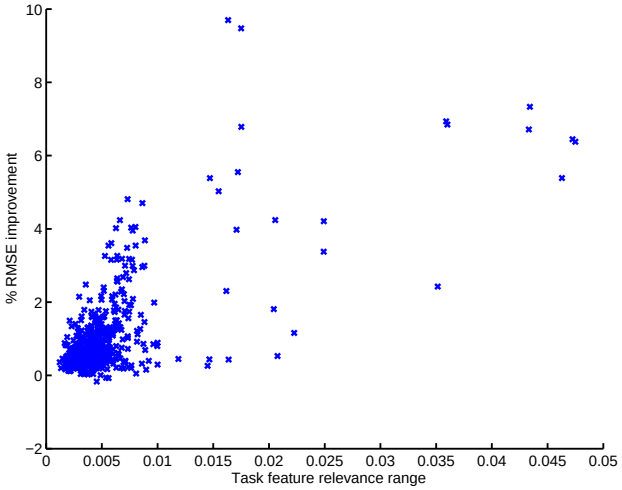

Figure 1. Relative RMSE improvement over a ridge regression baseline of the meta-level prior model with a training set of 100 examples from each movie. Each point is a movie task m. The x-axis is the range of the learned feature relevances for each movie: $\max_{m'} \boldsymbol{\beta}^\top \boldsymbol{f}_{m,m'} - \min_{m'} \boldsymbol{\beta}^\top \boldsymbol{f}_{m,m'}$. The results are averaged over 5 random splits of training and test sets.

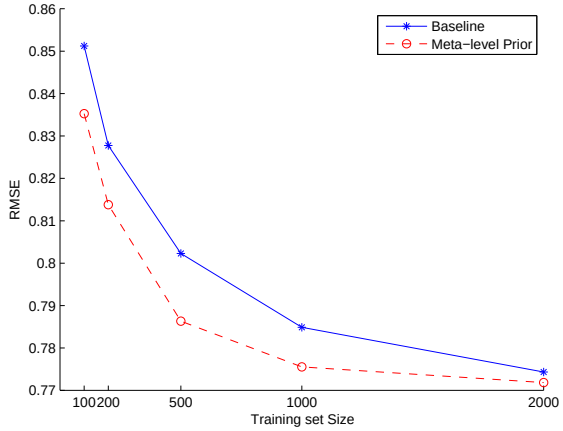

Figure 2. Improvement in RMSE of the meta-prior model over the baseline for the **Gen-test** with varying training set sizes.

In Figure 2, we show the effect of training set size on generalization performance in the **Gen-Test** experiment. As more training examples are available, the performance gains over the baseline model diminish. This is to be expected since, in such cases, there is usually sufficient information in the training set, reducing the importance of regularization.

We also performed a **Trans-Test** experiment, in which we used 60% of the movies in our dataset and 500 training

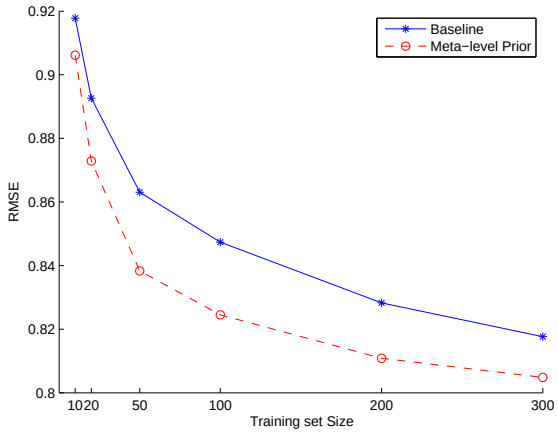

Figure 3. Improvement in RMSE of the meta-prior model over the baseline for the **Trans-test** with varying training set sizes.

examples per movie, to learn the meta-feature weights $\boldsymbol{\beta}$. We then tested the relevance transfer ability of the meta-prior on the remaining 40% of the movies, by using the learned meta-feature weights $\boldsymbol{\beta}$ to construct the feature relevance prior for each unseen movie, and estimating the weights $\boldsymbol{w}$ from the movie's training set. We set the weight for the bias meta-feature for the unseen task to the optimal baseline inverse ridge penalty, as determined using the validation set. Figure 3 shows the generalization ability of the meta-prior model for training set sizes ranging from 10 to 300. We can see that the transferred prior improves performance significantly. The improvement peaks at a training set size of 50; this indicates that, when using fewer examples, the noise in the training set overwhelms the benefit of the transferred prior, whereas for very large numbers of training instances, the baseline model does an adequate job of selecting relevant features, even without an informed prior. This result validates the usefulness of our algorithm in the transfer learning setting, where a new prediction task has a relatively small number of training instances.

5.2. Verb Argument Classification

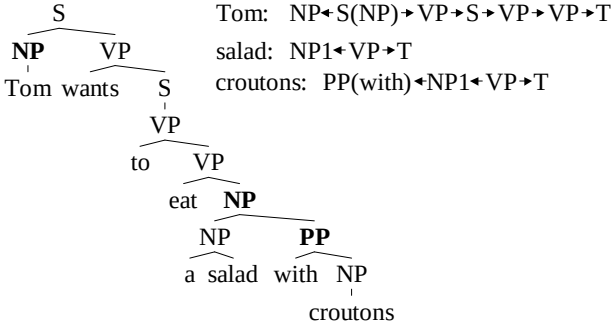

Figure 4. Example of parse tree features for semantic role labeling

We applied our algorithm to the natural language task of

semantic role labeling. Given a sentence containing a target verb, we want to label the semantic arguments, or roles, of that verb. For example, for the verb "eat", a correct labeling of the sentence "Tom wants to eat a salad with croutons" is {ARG0(Eater)="Tom",ARG1(Food)="salad"}. In the full semantic role labeling task, every phrase in the sentence must be labeled as a being an argument to the verb or not, and then labeled with a role from a set of approximately 30 different roles. In these experiments we tested on a somewhat simpler task: we first extracted all phrases which were either ARG0 (Agent) or ARG1 (Patient/Theme), and then attempted to classify each phrase as either ARG0 or ARG1.

We train and test on the PropBank corpus (Kingsbury et al., 2002), which labels the arguments of every verb in approximately 50000 sentences. Hand-labeled parses are also available for this corpus; we take these parses as given. As usual, we considered each verb (frame) as a different task. There were a total of 4216 verbs with at least one argument labeled either ARG0 or ARG1. We filtered out tasks that did not have at least 3 instances, resulting in 2610 tasks. For each task, we randomly split the instances into a training set, validation set, and test set, ensuring that each set had at least one instance for each set.

A variety of features of the sentence and the phrase being classified are used for semantic role labeling (Pradhan et al., 2005). The most important features are the head word of the phrase (e.g., "Tom" or "salad" above), and the path from the verb to the phrase being classified; these features are illustrated in Figure 4 for our example sentence.

For our results we consider a set of features that attempts to take advantage of structure within the paths from the verb to the phrase. In Figure 4, "Tom" has a fairly complicated path to the verb. In the sentence "Tom ate a salad." the path from "Tom" to "eat" is simpler, $NP \leftarrow S(NP) \rightarrow VP \rightarrow T$. However, in both cases the path contains the segment $NP \leftarrow S(NP) \rightarrow VP$. This example suggests that we take subsequences of the path as features in order to generalize between different paths with common elements. To keep the number of features from becoming too large, we restricted to segments containing exactly 3 nodes. Note that previous work (Moschitti, 2004) has used string or tree kernels to find common structure between paths; our feature set is somewhat similar to the implicit feature set encoded in these kernels. The difference here is that we will regularize each feature separately, which kernels are not able to do.

In order to increase the discriminative power of these features, we also include a number of generalizations/specializations of these features. Specifically, for each node we allow three choices: completely general ("*"), part-of-speech only (e.g., "NP"), or part-of-speech plus head word ("NP:Tom"). For each edge, we have two choices: general ("–") or specific (e.g. "→"). Thus, an example feature is "NP:Tom – * → VP". We generate all possible combinations of these choices; for a total of 3*2*3*2*3 = 108 different representations for a given sequence of 3 nodes (and 2 edges). Note that this set of features includes the head word of the phrase being classified, using features of the form "NP:Tom – * – *". This task illustrates the situation where there is very little feature sharing over the tasks: 80% of the 1154673 features appear in <10 tasks and 21% appear in only one task. We have one meta-feature for each feature, which specifies which of the 108 types it is, and therefore its degree of specificity. Thus, the information being transferred between tasks is not the relevance of individual features, but rather the appropriate level of abstraction in this complex feature space.

We compared two models: a baseline model — logistic regression with input features as described; and our meta-feature model using the same feature set and the meta-features described above. Both models include, for each feature, a global, unregularized mean parameter μ_k which allows feature means to be shared across tasks; we modify Eq.(2),

$$P(w_{rk} \mid \boldsymbol{\beta}, \mu_k, \boldsymbol{f}_{rk}) = \frac{1}{\sqrt{2\pi\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}}} \exp\left(-\frac{(w_{rk}-\mu_k)^2}{\boldsymbol{\beta}^\top \boldsymbol{f}_{rk}}\right).$$

Thus, we learn for each feature two pieces of information: the bias of the feature towards ARG0 vs. ARG1, and how relevant the feature is for distinguishing ARG0 from ARG1. This allows us to distinguish between features that always indicate a particular class (for example, some words are usually ARG0) and thus have high global mean but low variance; and features that are very relevant for deciding ARG0 vs. ARG1, but have different means for different tasks. For example, whether a phrase is the subject of the sentence is highly relevant for determining whether the phrase is the ARG0 or ARG1; but the subject can map to different arguments for different verbs: for "say", the subject is nearly always ARG0, while for "increase" it is often ARG1 ("stocks increased"). The meta-features allow us to choose these variances by, for example, indicating that features which include the direction of the edges tend to be more informative about whether the phrase is the subject of the sentence, and thus have high variances.

Figure 5 shows the results of these two models on a **Gen-Test** experiment, applied to a random subset of 500 of the 2610 tasks. The tasks are grouped into buckets based on the number of training examples. For tasks with few training examples, our model significantly improves over the baseline, decreasing error by 25% for verbs with 1-2 training examples. As we would expect, as the amount of training data increases the effect of the prior decreases, and the meta-features have less effect. However, as labeled training data is scarce in most language problems, improvement for sparse-data cases can be quite important in practice.

Figure 6 shows the results for a **Trans-Test** regime,

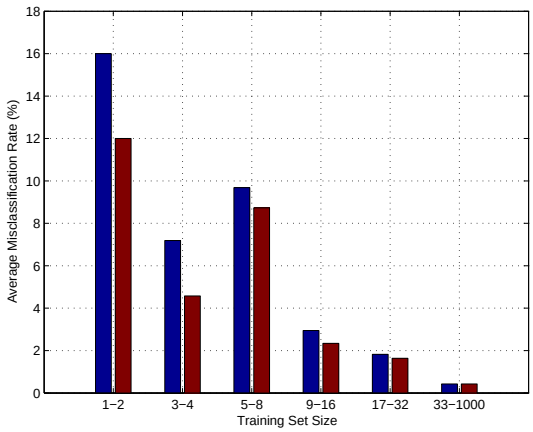

Figure 5. **Gen-test** for semantic role labeling. Blue(left) is baseline, Red(right) is the meta-feature model.

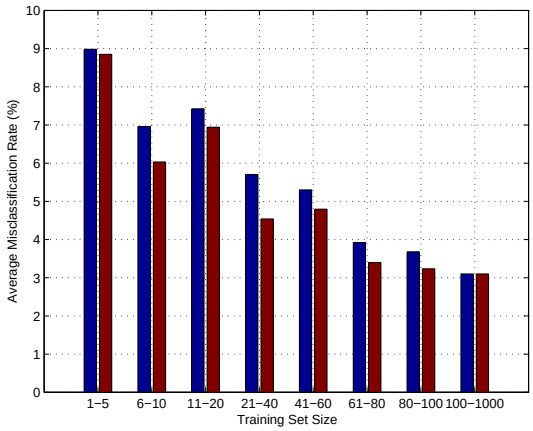

Figure 6. **Trans-test** for semantic role labeling

where the model is trained on the same 500 tasks above, and the learned $\boldsymbol{\beta}$ is then used for each of the remaining 2110 tasks. We again improve over the baseline model, although here the largest improvements are for tasks with a moderate amount of training data.

6. Related Work

This paper focuses on estimating the relevance of input features from an ensemble of related tasks, with the underlying assumption that similar features have similar relevance. Our algorithm lies at the intersection of two lines of research: feature selection and multi-task (or transfer) learning.

Traditionally, a prior for feature relevance is selected by hand, or via cross-validation. This approach is feasible only for a very small number of parameters, and hence, in most cases, all features are taken to be equally relevant, a priori. The best-known approach for automatically inferring the relevance of features is automatic relevance determination (ARD) (Neal, 1995; MacKay, 1992). As in our model, it models the distribution over a feature weight w_i to be a Gaussian with mean 0 and variance α_i. Given a classification data set $(\boldsymbol{x}_m, y_m)_{m=1..N}$, it chooses feature relevances α that maximize the marginal likelihood $p(\mathbf{y}|\alpha)$. ARD-based methods also aim to estimate the variance of each feature, while there are significant differences with our method. Our approach jointly optimizes the feature weights and the metaprior from multiple related tasks using the meta-features, while the ARD-based models estimate the relevance based on the marginal likelihood integrated over the weights with non-informative prior for a single task. Thus, our model identifies relevant features by looking for features with different biases for different tasks. The learned $\boldsymbol{\beta}$'s for the meta-features can also give insight about the domain. Additionally, our formulation is a convex optimization problem over both the weights and the metaprior, which allows efficient optimization techniques guaranteed to converge to the global optimum.

The concept of transfer learning was introduced by Baxter (1997), Caruana (1997) and Thrun (1996). Multiple approaches to transfer learning have been defined, including (Heskes, 2000; Evgeniou et al., 2005; Baxter, 2000; Teh et al., 2005). They vary in the model they use for how different tasks are related, which induces different types of information transfer between the tasks.

Somewhat related to our approach is the transfer of similarity between actual feature weights, as commonly done, for example, in a hierarchical Bayesian framework (McCallum et al., 1998). Most simply, the weights are asserted to be similar between classes, but some work defines the weights in different tasks to be a linear combination of the same set of components. For example, Taskar et al. (2003) and Fink et al. (2006) both define the weights associated with a feature (albeit using very different models and learning algorithms) as a linear combination of the feature's meta-features. Zhang et al. (2005) uses a similar decomposition of the feature weights, but automatically infers the components in the weight decomposition using techniques based on independent component analysis (ICA); this approach avoids the need for a set of pre-defined meta-features, but (conversely) does not take advantage of this prior knowledge when available. All of these approaches focus on modeling the similarity between the values of the feature weights, whereas we focus on transferring information regarding their relevance. More formally, in our probabilistic setting, we learn a model for the *variance* of a weight rather than its mean.

Some recent works considered the variance in multiple related tasks. Yu et al. (2005) proposed an EM-based algorithm for learning a Gaussian process from multiple related tasks based on a hierarchical Bayesian framework. Argyriou et al. (2006) learned the covariance matrix of a Gaussian prior over the features. However, none of these approaches generalize the relevance of the features,

and thus are not applicable to problems where tasks have (largely) non-overlapping features or where the relevance varies across the tasks.

Closest to our approach is the work of Raina et al. (2006), who proposed the use of transfer learning for constructing a multivariate Gaussian prior with a full covariance matrix for a given supervised learning task. Their task formulation is similar to ours, in that a parameter prior, involving both variances and covariances, is learned as a linear function of a set of meta-features. However, their approach first estimates covariances empirically, using a computationally expensive bootstrap procedure, then learns a prior from those estimates, and only then uses the prior for prediction. By contrast, in our approach, the prior is learned as part of a single, coherent objective, which encompasses both the data likelihood and the prior, and jointly optimizes over both parameters and hyperparameters. Moreover, this objective is convex, allowing efficient optimization and convergence to a unique global optimum.

7. Discussion

In this paper, we propose a probabilistic approach for learning an informed prior about feature relevance from an ensemble of related tasks. Specifically, our approach learns a variance for each feature as a function of its meta-features. Therefore, our framework can be viewed as a "paired" prediction problem: (1) predicting relevance of features using meta-features and (2) predicting the tasks using features. Our approach allows tranfer between tasks with completely different feature sets and allows the feature relevance to vary over different tasks.

Our work raises several interesting directions for future work. First, the framework we defined applies without change to learning relationships between parameter values in related tasks, by modeling their mean as a linear function of meta-features. As we discussed above, other approaches with the same goal have been proposed; but ours is a simple and efficient convex formulation, which may have benefits in practice. In a different direction, one could follow the work of Raina et al. (2006), and jointly learn a prior of both the variances and the covariances of the weights, thereby modeling not only their relevance, but also the relationships between them. We believe that our formulation could easily be extended to cover this case, without losing convexity. Finally, it would be interesting to generalize our methods to apply to cases where we do not have (enough) pre-defined meta-features, by inducing factors in a decomposition; this approach was taken by Zhang et al. (2005) when modeling a prior over the mean of the weights, but the application to modeling variance is far from obvious.

References

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. *Proceeding of NIPS*. Cambridge, MA: MIT Press.

Baxter, J. (1997). A bayesian/information theoretic model of learning to learn viamultiple task sampling. *Mach. Learn.*, *28*, 7–39.

Baxter, J. (2000). Model for inductive learning. *J. of Artificial Intelligence Research*.

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

Evgeniou, T., Micchelli, C., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*

Fink, M., Shwatz-Shalev, S., Singer, Y., & Ullman, S. (2006). Online multiclass learning by interclass hypothesis sharing. *Proc. 23rd International Conference on Machine Learning*.

Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles.

Heskes, T. (2000). Empirical bayes for learning to learn. *Proc. 17th International Conference on Machine Learning*.

Kaelbling, L. (2003). JMLR special issue on variable and feature selection.

Kingsbury, P., Palmer, M., & Marcus, M. (2002). Adding semantic annotation to the penn treebank. *Proceedings of the Human Language Technology Conference (HLT'02)*.

MacKay, D. (1992). Bayesian interpolation. *Neural Computation*, *4*, 415–447.

Marlin, B. (2004). Collaborative filtering: A machine learning perspective.

McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. .

McCullagh, P., & Nelder, J. (1989). *Generalized linear models*. London: Chapman and Hall.

Moschitti, A. (2004). A study on convolution kernels for shallow statistic parsing. *ACL*.

Neal, R. (1995). *Bayesian learning for neural networks*. Doctoral dissertation. Adviser-Geoffrey Hinton.

Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J. H., & Jurafsky, D. (2005). Support vector learning for semantic argument classification. *Machine Learning*, *60*, 11–39.

Raina, R., Ng, A., & Koller, D. (2006). Transfer learning by constructing informative priors. *Proc. 21st International Conference on Machine Learning*.

Taskar, B., Wong, M., & Koller, D. (2003). Learning on the test data: Leveraging unseen features. *Proc. 20th International Conference on Machine Learning*.

Teh, Y., Seeger, M., & Jordan, M. (2005). Semiparameteric latent factor models. *Workshop on Artificial Intelligence and Statistics 10*.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems* (pp. 640–646). The MIT Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B*.

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

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

Zhang, J., Ghahramani, Z., & Yang, Y. (2005). Learning multiple related tasks using latent independent component analysis. *Advances in Neural Information Processing Systems 17*.