
Kernel Learning and Meta Kernels for Transfer Learning

Ulrich Rückert

University of California, Berkeley
Department of Computer Science
750 Sutardja Dai Hall #1776
Berkeley, CA 94720-1776
rueckert@eecs.berkeley.edu

Abstract

In transfer learning, the task is to find a suitable learning bias for a new target dataset, given a set of related source datasets. In this study, we extend our previous work on kernel-based inductive transfer, where we aim at inducing a kernel from the source datasets, which leads to good predictive accuracy on the new target data. To find such a kernel, we proceed in three steps. First, we find a predictive kernel for each of the source datasets. Second, we induce a meta-classifier, which combines these source kernels into a new kernel for the target dataset. Third, we use the new target kernel to learn a successful classifier for the target data. The first step is a perfect application for kernel learning algorithms, because here we are more interested in finding a predictive kernel rather than a classifier. We present a new fast method based on semidefinite programming and a restricted classifier space to learn such source kernels. To combine these source kernels into a suitable kernel for the target learning task, we present three new meta kernels that quantify the similarity of two datasets. Preliminary experiments indicate that the approach performs well on datasets with structured data.

1 Introduction

It is well known that the success or failure of a supervised learning method depends on its bias. If the bias matches well with the underlying learning problem, the system will be able to construct predictive models. If the bias does not match very well, the generated classifier will perform poorly. One of the great advantages of kernel-based methods is the fact that the learning bias can be flexibly adjusted by choosing a customized kernel for the data at hand. However, building custom kernels from scratch for individual applications can be a tedious task. Recent research has dealt with the problem of learning kernels automatically from data, see e.g. the work by Ong *et al.* [1] and Lanckriet *et al.* [2]. In practice actual training data is often rare and in most cases it is better to invest it for the actual learning task than for kernel selection. Even though data from the same source may be rare, it is sometimes the case that data on related or similar learning problems is available. As an example, for text classification problems, plenty of related text data might be available on the internet. Similarly, for some problems from computational chemistry, research on related endpoints might lead to related datasets.

In a previous paper [3] we proposed a framework to address these issues. We performed kernel-based transfer learning in a three step process: First, we used an iterative projection-based method to find source kernels, which generalize well on the existing source data. Having such a “known good” kernel for each of the source datasets, we frame the problem of finding a good kernel for the new data at hand as a *meta learning* problem. Roughly, this learning problem can be stated as follows: given a set of source datasets together with the corresponding good kernels, what would a good kernel for

the target data look like? We solved this meta learning task using a strategy based on regularized convex loss minimization with a meta-kernel. For the case where the design of domain-specific meta-kernels is too tedious or impossible (perhaps due to lack of suitable background knowledge), we proposed the *histogram kernel*, a generic meta-kernel that is applicable for standard propositional datasets.

In this paper we extend our previous work in two directions. First, we describe a faster method for finding predictive kernels. The method in [3] makes use of an iterative procedure, which alternates between solving two convex optimization problems. On some datasets, convergence of this strategy can be slow. As an alternative, we present a modified optimization criterion, which can be expressed as a semidefinite program. Second, we present three new meta-kernels for datasets with varying characteristics. We perform preliminary experiments for two applications. In the first application, the task is to predict the biological activity of chemical compounds from the compounds molecular structure, given as a labeled graph. The second application is text categorization.

2 Kernel-Based Inductive Transfer

As explained in the introduction, we proceed in three steps to find a kernel, which is likely to perform well on the target data. First of all, we compute for each source dataset (\bar{X}_i, \bar{Y}_i) a kernel \bar{k}_i^* that generalizes well from (source) training data to (source) test data. Then, in the second step, we tackle the problem of finding a good kernel k for the target learning data (X, Y) . We frame this as a meta learning problem. In particular, we make use of a *meta kernel* $\bar{k} : ((\bar{X}, \bar{Y}), (\bar{X}', \bar{Y}')) \mapsto r \in \mathbb{R}$, defined on the space of (source) datasets, to induce a meta model, represented by a coefficient vector $\bar{\alpha}$ and a threshold \bar{b} . Given a (target) dataset, the meta model outputs a kernel that is likely to work well for learning on the target data. For notational clarity, we mark all parameters and variables that deal with the meta learning task (as opposed to the source and target learning tasks) with a circle (e.g., we write \bar{k} to represent the meta kernel). Finally, in the last step, we apply the meta model to the training data (\bar{X}, \bar{Y}) at hand, yielding a target kernel k . This kernel is then employed in the target SVM to build a final classifier.

2.1 Learning Kernels from Transfer Data

In the first step, we would like to discover which bias works well for each of the k source data sets. Since we are working with SVMs, this essentially boils down to the question of what kernel should be chosen in each case. Recall that the soft-margin SVM optimizes the regularized hinge loss of classifier \bar{f} on the training set. More formally, let \bar{k} be some positive semi-definite kernel, $l_h(\bar{y}, \bar{f}(\bar{x})) = \max(0, 1 - \bar{y}\bar{f}(\bar{x}))$ denote the *hinge loss*, and let $C > 0$ be a tuning parameter. Then, the SVM minimizes the following functional over all linear classifiers $\bar{f} \in \mathcal{H}_{\bar{k}}$ in the Hilbert space produced by kernel \bar{k} , where $\|\cdot\|_{\bar{k}}$ denotes the norm in this space: $S(\bar{X}, \bar{Y}, \bar{f}, \bar{k}) := C \sum_{(\bar{x}, \bar{y}) \in (\bar{X}, \bar{Y})} l_h(\bar{y}, \bar{f}(\bar{x})) + \|\bar{f}\|_{\bar{k}}$.

The standard SVM computes the optimal classifier \bar{f}^* by minimizing $S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$ over \bar{f} , while keeping the kernel \bar{k} fixed: $\bar{f}^* := \operatorname{argmin}_{\bar{f} \in \mathcal{H}_{\bar{k}}} S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$. Since the hinge loss can be seen as a robust upper bound of the zero-one loss, it is a sensible strategy to select not only the classifier \bar{f} , but also the kernel \bar{k} by minimizing $S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$. In other words, to find a good kernel for a given dataset, one can solve $(\bar{f}^*, \bar{k}^*) := \operatorname{argmin}_{\bar{f} \in \mathcal{H}_{\bar{k}}, \bar{k} \in \mathcal{K}} S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$, where \mathcal{K} denotes some pre-defined space of possible kernels. If \mathcal{K} is a convex set, this enlarged optimization problem is still convex and can thus be solved efficiently [2].

Unfortunately, minimizing $S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$ over a source data set (\bar{X}, \bar{Y}) does not necessarily lead to a kernel that generalizes well on new data. This is for two reasons. First, by optimizing \bar{k} and \bar{f} at the same time, one finds a kernel \bar{k}^* that works well together with the optimal \bar{f}^* . However, when one applies the kernel later on new data, the SVM might induce a \bar{f} , which might differ from \bar{f}^* and does therefore not match well with the \bar{k}^* . In other words, we are not looking for a \bar{k}^* that works well with \bar{f}^* , but a kernel that works well with an \bar{f} that is typically chosen by an SVM on new data. Second, for some classes of kernels the kernel matrix has always full rank. This means that there is always a subspace $H_0 \subseteq \mathcal{H}_k$ whose classifiers $\bar{f} \in H_0$ achieve $\sum_{i=1}^n l_h(\bar{y}_i, \bar{f}(\bar{x}_i)) = 0$. This is also true for practically relevant classes of kernels, for instance, radial basis kernels. In those

cases, minimizing $S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$ focuses almost exclusively on the regularization term $\|\bar{f}\|_{\bar{k}}$ and the data-dependent term is largely ignored (because it is zero in most cases). In other words, a kernel matrix of full rank leads to overfitting in the sense that the optimization procedure prefers kernels that match well with the regularization criterion rather than kernels that catch the bias inherent in the data.

To avoid the two problems, we proposed an iterative method in [3], which alternates between optimizing the kernel and optimizing the linear classifier. Unfortunately, the method can be slow to converge on some datasets. As a faster alternative, we propose the following modified optimization criterion. As in the original approach, we split (\bar{X}, \bar{Y}) into two parts and modify the optimization criterion, so that \bar{f} depends mainly on the first part of the data, whereas the kernel $\bar{k} \in \mathcal{K}$ is evaluated on the whole dataset. More precisely, let (\bar{X}', \bar{Y}') be some subset of (\bar{X}, \bar{Y}) . To reduce the number of parameters learned for the linear classifier, we assume that the classifier is contained only in the Hilbert space induced by \bar{X}' . This is in contrast to the standard SVM setting, where f is taken from the Hilbert space for the whole training set. This means we can represent the classifier using a weight α_i for each instance in \bar{X}' :

$$\bar{f}(x) := \sum_{\bar{x}' \in \bar{X}'} \alpha_i \bar{k}(\bar{x}', x)$$

We would like to find \bar{f} which optimizes the soft-margin criterion on \bar{X} . Thus, we have the following optimization problem:

$$\begin{aligned} & \underset{\alpha, b, \xi, \bar{k}}{\text{minimize}} && C \sum_{i=1}^n \xi_i + \left\| \sum_{\bar{x}' \in \bar{X}'} \alpha_i \bar{k}(\bar{x}', x) \right\|_{\bar{k}}, \\ & \text{subject to} && y_j \left(\sum_{\bar{x}' \in \bar{X}'} \alpha_i \bar{k}(\bar{x}', x_j) + b \right) \geq 1 - \xi_j \text{ for all } j \end{aligned} \quad (1)$$

In this way, f^* is chosen from a much smaller and more restricted space of classifiers. Using the Karush-Kuhn-Tucker conditions, one can show that the solution α^* must satisfy

$$\alpha^* = \bar{K}'^{-1} \sum_{\bar{x}_i \in \bar{X}} \beta_i y_i \bar{k}(\bar{x}_i, \bar{X}')$$

Here, the β_i are the Lagrange multipliers of the convex dual of the original optimization problem, \bar{K}' is the kernel matrix for \bar{X} and $\bar{k}(\bar{x}_i, \bar{X}')$ denotes the vector of kernel values $k(\bar{x}_i, \bar{x}')$ for all $\bar{x}' \in \bar{X}'$. Plugging this into the usual dual form of the SVM criterion, we get the following optimization problem: minimize $\beta_{\bar{k}} \beta^T \mathbf{K} \beta - \sum_i y_i \beta_i$ subject to $0 \leq \beta \leq C, \beta \geq 0$ with

$$\mathbf{K} = \begin{pmatrix} \bar{K}_{11} & \bar{K}_{12} \\ \bar{K}_{12}^T & \bar{K}_{12}^{-1} \bar{K}_{12} \end{pmatrix}$$

where $\bar{K}_{11} = [k(x_1, x_2)]_{x_1, x_2 \in \bar{X}'}$ is the kernel matrix for the instances in \bar{X}' and $\bar{K}_{12} = [k(x_1, x_2)]_{x_1 \in \bar{X}, x_2 \in \bar{X}'}$ contains the kernel values for the instances in \bar{X} and \bar{X}' . Thus, in order to optimize (1), we can compute \mathbf{K} and plug it into the semidefinite program described in [2]. This program computes the optimal kernel k^* and classifier f^* , which is taken only from the Hilbert space induced by the examples in \bar{X}' .

2.2 Meta Kernels

In the preceding section we described a method to find a kernel \bar{k}_i^* that encodes the bias inherent in each source dataset (\bar{X}_i, \bar{Y}_i) . Now, we address the second step: How can we make use of this source information when dealing with the target learning problem, where we wish to learn a classifier for the target dataset (X, Y) ? In particular, given the \bar{k}_i^* , what should a “good” target kernel k for the target data look like? Since we assume that the source learning problems are similar to the target learning problem, choosing the average over the \bar{k}_i^* appears to be a good option. On the second sight, though, it is clear that some source learning problems are more similar to the target setup than others. Thus, it makes sense to frame the task of finding a good target kernel k as a *meta*

learning problem. Formally, this problem can be stated as follows: Given a set of t source datasets $(\bar{X}_1, \bar{Y}_1), \dots, (\bar{X}_t, \bar{Y}_t)$ and the corresponding t known good kernels $\bar{k}_1^*, \dots, \bar{k}_t^*$, we would like to predict a new kernel k that performs as good as possible on the target data (X, Y) .

We tackle this meta learning problem using a kernel-based approach. As a first step, we need a meta kernel $\hat{k} : (\mathcal{X}, \mathcal{Y}) \times (\mathcal{X}, \mathcal{Y}) \rightarrow r \in \mathbb{R}$. As it is the case with all kernel-based classification systems, it is best to apply a kernel that incorporates domain specific knowledge about the problem at hand. In our experiments, we work with the following approaches:

- The *histogram kernel*. This kernel is described in our previous paper [3]. It compares the sorted sequences of feature and example vectors in the two training matrices and is designed for cases where no usable meta-information is available.
- The *Gaussian meta kernel*. Here, we assume that both datasets use the same features. We compute the sample means μ_1 and μ_2 and covariance matrices Σ_1 and Σ_2 from the two data samples. The kernel value between the two datasets is then given by $\hat{k}((\mu_1, \Sigma_1), (\mu_2, \Sigma_2)) := \int_{\mathcal{X}} f(x; \mu_1, \Sigma_1) f(x; \mu_2, \Sigma_2) d\mu(x)$, where the $f(x; \mu, \Sigma)$ is the probability density function of the normal distribution with parameters μ and Σ . This quantity can be interpreted as the probability that the two samples are taken from the same distribution under the assumption that they are normally distributed. One can show that

$$\hat{k}((\mu_1, \Sigma_1), (\mu_2, \Sigma_2)) = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma_1 + \Sigma_2|^{\frac{1}{2}}} \exp \left[-\frac{1}{2} (\mu_1 - \mu_2)^T (\Sigma_1 + \Sigma_2)^{-1} (\mu_1 - \mu_2) \right]$$

- The *Gaussian mixture meta kernel*. This is similar to the Gaussian meta kernel, but we model the data distribution as a mixtures of two Gaussians, one for each class.
- The *feature meta kernel*. This is well suited for settings where the features in the two datasets are homogenous and each feature can be described by a feature description vector f . This is the case for instance, when the feature test the occurrence of substructures in the (structured) training examples. The kernel $\hat{k}(F_1, F_2) = \text{trace}(F_1^T F_2)$ is defined to be the Frobenius product between the two feature description matrices F_1 and F_2 .

3 Conclusion

In this paper we extend our previous work [3] in two directions. First, we describe a faster approach to kernel learning for inductive transfer. The method is based on a semidefinite program, which restricts the linear classifier to be contained in a smaller Hilbert space. Second, we describe three new meta kernels for combining the source kernels into a predictive target kernel for the target learning task. Preliminary experiments (omitted here due to space constraints) indicate that the new kernel learning method is considerably faster and that the new meta kernels can be applied successfully to transfer learning biases. In future work we plan to extend the described framework to settings, where the examples are nodes in large networks and the usual i.i.d. assumption does not hold.

References

- [1] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *J. Mach. Learn. Res.*, 6:1043–1071, 2005.
- [2] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.
- [3] Ulrich Rückert and Stefan Kramer. Kernel-based inductive transfer. In *ECML PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 220–233, Berlin, Heidelberg, 2008. Springer-Verlag.