Kernel-Based Inductive Transfer

Ulrich Rückert and Stefan Kramer

Institut für Informatik/I12, Technische Universität München, Boltzmannstr. 3,
D-85748 Garching b. München, Germany
{rueckert,kramer}@in.tum.de

Abstract. Methods for inductive transfer take advantage of knowledge from previous learning tasks to solve a newly given task. In the context of supervised learning, the task is to find a suitable bias for a new dataset, given a set of known datasets. In this paper, we take a kernel-based approach to inductive transfer, that is, we aim at finding a suitable kernel for the new data. In our setup, the kernel is taken from the linear span of a set of predefined kernels. To find such a kernel, we apply convex optimization on two levels. On the base level, we propose an iterative procedure to generate kernels that generalize well on the known datasets. On the meta level, we combine those kernels in a minimization criterion to predict a suitable kernel for the new data. The criterion is based on a meta kernel capturing the similarity of two datasets. In experiments on small molecule and text data, kernel-based inductive transfer showed a statistically significant improvement over the best individual kernel in almost all cases.

Keywords: kernels, inductive transfer, transfer learning, regularized risk minimization.

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.* [17] and Lanckriet *et al.* [14]. 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.

The task of using such related data to improve accuracy for the the learning problem at hand is known as *inductive transfer*. Here, the main idea is that a kernel (i.e., a bias) that has worked well on the related *transfer datasets* should also work well on the new data. One particularly pragmatic approach to inductive transfer is to build a range of classifiers with varying kernels and settings on the transfer data and to evaluate the predictive accuracy of those classifiers. The kernel that performed best on the transfer datasets could then be selected for the new data. While conceptually simple, this method has three disadvantages. First, classifier evaluation takes quite a lot of time, because evaluation methods like cross validation require the generation of many classifiers. Second, the method evaluates only single kernels and does not take into account the case where a combination of many kernels might perform better than each individual kernel. Third, it does not consider the fact that some transfer datasets are more similar to the learning problem at hand than others.

In this paper we would like to address these issues. As a first contribution we present a method that finds kernels, which generalize well on the existing transfer data without the need to resort to expensive evaluation methods. Having these "known good" kernels for the transfer data, 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 transfer datasets together with the corresponding good kernels, what would a good kernel for the data at hand look like? We propose to solve 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 propose the *histogram kernel*, a generic meta-kernel that is applicable for standard propositional datasets.

The paper is organized as follows. After a brief review of some related work in Section 2, we introduce the setting and describe our approach in Section 3. We start with the problem of finding good kernels for the transfer data in Section 3.1, present the histogram kernel in section 3.2 and discuss our approach to the meta learning problem in Section 3.3. Finally, we report on experiments in Section 4 and conclude in Section 5.

2 Related Work

The presented work is related to research in three areas. On one side, there has been considerable progress in learning kernels from data. The original formulation as a semi-definite optimization problem by Lanckriet *et al.* [14] has been extended in various directions [16,19]. Other techniques for kernel learning include hyper kernels [17] and boosting [4,12]. All these approaches aim at learning a good kernel from training data rather than from transfer data. On the other side, there is a long history of work on inductive transfer, see e.g. Baxter [2]. Of course, inductive transfer is related to multi-task learning [3], where the goal is to induce classifiers for many tasks at once. Multi-task learning with kernels

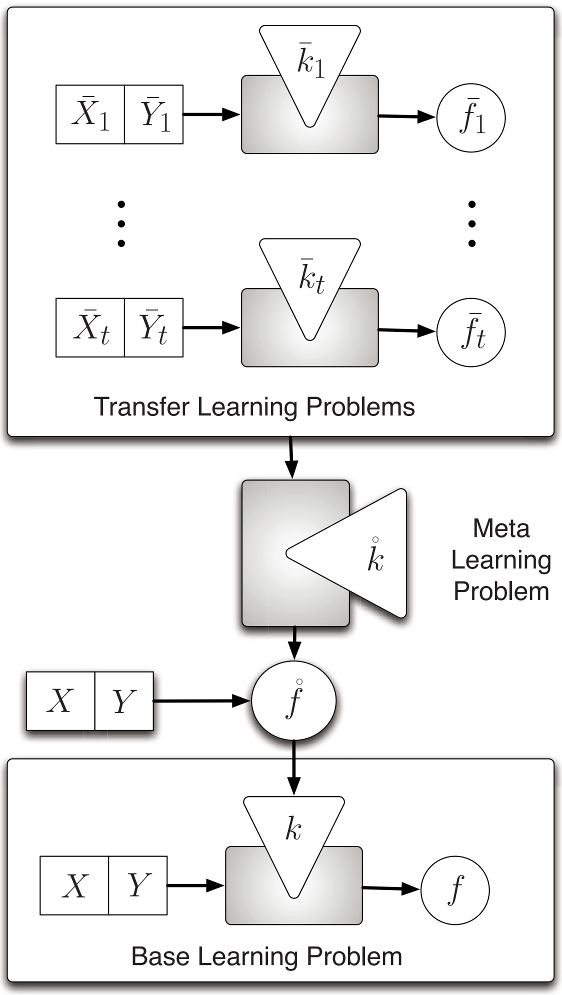

Fig. 1. We aim at finding a good kernel k for the base learning problem. To do so, we search for good kernels $\bar{k}_1, \ldots, \bar{k}_n$ for the n transfer learning problems. The meta learning problem deals with learning a predictor $\mathring{f}$, which outputs a good base kernel k for the base data.

has been the subject of research by Evgeniou *et al.* [6] and Argyriou *et al.* [1]. It is often approached using Bayesian methods [10,24,22]. This paper deals with a more asymmetric setting, where we use the older datasets only to increase predictive performance on the new data at hand. Similar settings have been the subject of work by Kaski and Peltonen [13] and Erhan *et al.* [5]. While Kaski and Peltonen consider the case where only a few instances in the transfer datasets are relevant, the study by Erhan *et al.* aims at generalizing from the transfer data only, so that no base data is necessary.

Finally, the presented work is also loosely related to research on *meta learning*. Here, the goal is to induce meta-classifiers, which, given a dataset and its characteristics, propose a learning algorithm that is supposed to perform well on the data at hand. We refer to Pfahringer *et al.* [18] for a meta learning study based on landmarking and a short overview of related work.

3 Kernel-Based Inductive Transfer

For the following discussion, we assume a setting, where a user is interested in finding a good classifier for some new data. Additionally, she has access to a range of older datasets, which are assumed to be similar to the new data in some regard. In the following, we call the older datasets *transfer datasets* and the new data *base dataset*. The main question we are dealing with is how to find a good kernel (i.e. bias) for the base learning problem, given the old transfer datasets. As illustrated in Figure 1, we frame this problem as a *meta learning* task. We proceed in three steps. First, for each transfer dataset, we generate a kernel that leads to a highly predictive classifier on this data. Then, from the transfer datasets and kernels, we learn a *meta classifier* $\mathring{f}$, which predicts a new base kernel when given a base dataset. The meta learning algorithm makes use of the *meta kernel* $\mathring{k}$ to compare two datasets. Finally, in the last step, the meta classifier is applied to the base data at hand, leading to a kernel k for the base learning problem. This kernel is then plugged into a standard SVM to construct a suitable classifier for the base data.

Let us introduce some notation before we delve into the details. As usual, we assume that the training data is given as a set of labeled examples (X, Y), where the rows $x_1, \ldots, x_n$ of the training matrix $X \in \mathbb{R}^{n \times m}$ constitute the examples, and the $y_1, \ldots, y_n \in \{-1, 1\}$ represent the class labels. The goal is to induce a classifier from (X, Y) that predicts well on new, previously unseen test instances. Since we are dealing with the *inductive transfer* setting, we are given an additional set of t *transfer datasets* $(\bar{X}_1, \bar{Y}_1), \ldots, (\bar{X}_t, \bar{Y}_t)$, which are supposed to pose similar learning problems as the one given by (X, Y). To allow for a concise description, we mark all parameters and variables referring to the transfer problems with a bar. For instance, given a particular transfer dataset $(\bar{X}, \bar{Y})$, we let $(\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_n, \bar{y}_n)$ denote its examples. Similarly, if an SVM is applied on this data, we denote the coefficient vector and threshold output by the SVM with $\bar{\alpha} := (\bar{\alpha}_1, \ldots, \bar{\alpha}_n)^T$ and $\bar{b}$.

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

3.1 Learning Kernels from Transfer Data

In the first step, we would like to discover which bias works well for each of the k transfer 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.

$$S(\bar{X}, \bar{Y}, \bar{f}, \bar{k}) := C \sum_{(\bar{x},\bar{y}) \in (\bar{X},\bar{Y})} l_h(\bar{y}, \bar{f}(\bar{x})) + \|\bar{f}\|_{\bar{k}} \tag{1}$$

The standard SVM computes the optimal classifier $\bar{f}^*$ by minimizing (1) 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 $(\bar{f}^*, \bar{k}^*) := \arg\min_{\bar{f} \in \mathcal{H}_{\bar{k}}, \bar{k} \in \mathcal{K}} S(\bar{X}, \bar{Y}, \bar{f}, \bar{k})$, 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 [14].

Unfortunately, minimizing (1) over a transfer 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.

Algorithm 1. An iterative procedure to find a kernel that generalizes well on the dataset $(\bar{X}, \bar{Y})$

```
procedure FindGoodKernel($\bar{X}$, $\bar{Y}$, $C$)
    select subset $(\bar{X}', \bar{Y}')$ from $(\bar{X}, \bar{Y})$
    select initial $\bar{k}^{(0)} \in \mathcal{K}$
    $i \leftarrow 0$
    repeat
        $i \leftarrow i + 1$
        $\bar{f}^{(i)} \leftarrow \operatorname{argmin}_{\bar{f} \in \mathcal{H}_{\bar{k}}} S(\bar{X}', \bar{Y}', \bar{f}, \bar{k}^{(i-1)})$
        $\bar{k}^{(i)} \leftarrow \operatorname{argmin}_{\bar{k} \in \mathcal{K}} S(\bar{X}, \bar{Y}, \bar{f}^{(i)}, \bar{k})$
    until $S(\bar{X}, \bar{Y}, \bar{f}^{(i)}, \bar{k}^{(i)}) \geq S(\bar{X}, \bar{Y}, \bar{f}^{(i-1)}, \bar{k}^{(i-1)})$
    return $(\bar{k}^{(i)}, \bar{f}^{(i)})$
end procedure
```

To avoid the two problems, we split $(\bar{X}, \bar{Y})$ into two parts and modify the optimization criterion, so that $\bar{f}$ depends only on the first part of the data, whereas the kernel $\bar{k} \in \mathcal{K}$ is evaluated on the whole dataset. In this way, f^* is chosen from a much smaller and more restricted space of classifiers. Consequently, the optimization procedure needs to better adjust $\bar{k}^*$ to ensure that even a $\bar{f}^*$ from the rather restricted subspace generalizes well to the remaining instances. This setup is similar to classifier evaluation with a hold-out set, where a classifier is induced from the training set and then evaluated on a separate test set. More formally, let $(\bar{X}', \bar{Y}')$ be some subset of $(\bar{X}, \bar{Y})$. We use the standard SVM regularized risk functional (1) to rate a classifier $\bar{f}$ for a fixed kernel $\bar{k}$ on this subset:

$$\bar{f}^*_{\bar{k}} := \operatorname*{argmin}_{\bar{f} \in \mathcal{H}_{\bar{k}}} S(\bar{X}', \bar{Y}', \bar{f}, \bar{k}) \tag{2}$$

Then, we choose the optimal kernel so that it performs best with $\bar{f}^*_{\bar{k}}$ on all examples. More precisely, the desired optimal kernel $\bar{k}^*$ is

$$\bar{k}^* = \operatorname*{argmin}_{\bar{k} \in \mathcal{K}} S(\bar{X}, \bar{Y}, \operatorname*{argmin}_{\bar{f} \in \mathcal{H}_{\bar{k}}} S(\bar{X}', \bar{Y}', \bar{f}, \bar{k}), \bar{k}) \tag{3}$$

This criterion contains two nested optimization problems, one to determine the best classifier, the other, which depends on the first one, to compute the optimal kernel. Generally, the functional is not convex when optimizing over $\bar{k}$ and $\bar{f}^*_{\bar{k}}$ at the same time. However, on their own, the two single parts are convex optimization problems: The first part $\operatorname{argmin}_{\bar{f} \in \mathcal{H}_{\bar{k}}} S(\bar{X}', \bar{Y}', \bar{f}, \bar{k})$ is the standard SVM criterion, that is, a quadratic program. The second part $\operatorname{argmin}_{\bar{k} \in \mathcal{K}} S(\bar{X}, \bar{Y}, \bar{f}^*_{\bar{k}}, \bar{k})$ is a convex criterion, if one sets $\bar{f}^*_{\bar{k}}$ to a fixed value and optimizes only over $\bar{k}$. This naturally leads to an optimization procedure that alternates between optimizing (2) with a fixed $\bar{k}$ and optimizing (3) with a fixed $\bar{f}^*_{\bar{k}}$. The loop is terminated as soon as neither of the two steps improves on the score anymore. The approach is outlined in Algorithm 1.

Of course, it is impractical to deal with the explicit representations of $\bar{f}^*_{\bar{k}}$ in a high-dimensional Hilbert space. Fortunately, the representer theorem ensures that the optimal $\bar{f}^*_{\bar{k}}$ is always contained in the linear span of $\{k(\bar{x}', \cdot)|\bar{x}' \in \bar{X}'\}$. Therefore, a classifier $\bar{f}$ can be conveniently represented by a coefficient vector $\bar{\alpha}$ and a threshold $\bar{b}$, so that $\bar{f}(x) = \sum_{i=1}^{n'} \bar{y}'_i \bar{\alpha}_i \bar{k}(x, \bar{x}'_i) + \bar{b}$. With this, the first criterion (2) can be equivalently stated as follows:

$$\underset{\bar{\alpha} \in \mathbb{R}^n_+, \bar{b} \in \mathbb{R}}{\operatorname{argmin}} \; C \sum_{i=1}^{n'} l_h(\bar{y}_i, [\bar{K}' \bar{D}' \bar{\alpha}]_i + \bar{b}) + \bar{\alpha}^T \bar{D}' \bar{K}' \bar{D}' \bar{\alpha} \tag{4}$$

Here, $\bar{\alpha}$ and $\bar{b}$ are the coefficient vector and threshold of the linear classifier to be found, n' is the number of instances in $(\bar{X}', \bar{Y}')$, $\bar{K}'$ denotes the $n' \times n'$ kernel matrix with $\bar{K}'_{ij} = \bar{k}(\bar{x}_i, \bar{x}_j)$, $\bar{D}'$ is a $n' \times n'$ diagonal matrix whose diagonal contains the class labels $\bar{D}'_{ii} = \bar{y}'_i$, and $[x]_i$ denotes the ith component of vector x.

The exact form of the second criterion depends on the structure of $\mathcal{K}$. For the experiments in section 4 we set $\mathcal{K}$ to the space of positive linear combinations of l main kernels $\bar{k}_1, \ldots, \bar{k}_l$. This means that a kernel $\bar{k} \in \mathcal{K}$ can be represented by a vector $\bar{\mu} \in \mathbb{R}^l_+, \|\bar{\mu}\| \leq 1$ of linear coefficients, because $\bar{k}(\bar{x}_1, \bar{x}_2) = \sum_{i=1}^{l} \bar{\mu}_i \bar{k}_i(\bar{x}_1, \bar{x}_2)$. With this, criterion (3) becomes:

$$\underset{\bar{\mu} \in \mathbb{R}^n_+, \|\bar{\mu}\| \leq 1}{\operatorname{argmin}} \; C \sum_{i=1}^{n} l_h(\bar{y}_i, [\bar{M}\bar{\mu}]_i + \bar{b}) + \bar{r}^T \bar{\mu} \tag{5}$$

Here, $\bar{M} \in \mathbb{R}^{n \times l}$ with $\bar{M}_{ij} = \bar{y}_i \sum_{k=1}^{n'} \bar{y}_k \bar{\alpha}_k \bar{k}_j(\bar{x}_i, \bar{x}_k)$, and $\bar{r} \in \mathbb{R}^l$ with $\bar{r}_k = \sum_{i=1}^{n'} \sum_{j=1}^{n'} \bar{y}_i \bar{y}_j \bar{\alpha}_i \bar{\alpha}_j \bar{k}_k(\bar{x}_i, \bar{x}_j)$. Of course, the overall transfer learning scheme works also with other kernel spaces $\mathcal{K}$. For instance, one could choose $\mathcal{K} = \{\bar{k}_p | p \in \mathbb{R}\}$, where $\bar{k}_p$ is a kernel function parameterized with p. If the resulting kernel space $\mathcal{K}$ is a convex set, the corresponding transfer optimization criterion (3) and the meta learning problem (6) (see section 3.3) can be cast as convex optimization problems and are thus solvable with standard methods.

3.2 The Histogram Kernel for the Meta Learning Task

In the preceding section we described a method to find a kernel $\bar{k}^*_i$ that encodes the bias inherent in each transfer dataset $(\bar{X}_i, \bar{Y}_i)$. Now, we address the main question of the paper: How can we make use of this transfer information when dealing with the base learning problem, where we wish to learn a classifier for the base dataset (X, Y)? In particular, given the $\bar{k}^*_i$, what should a "good" base kernel k for the base data look like? Since we assume that the transfer learning problems are similar to the base 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 transfer learning problems are more similar to the base setup than others. Thus, it makes sense to frame the task of finding a good base kernel k as a *meta learning* problem. Formally, this problem can be stated as follows: Given a set of t transfer datasets $(\bar{X}_1, \bar{Y}_1), \ldots, (\bar{X}_t, \bar{Y}_t)$ and the corresponding t known good kernels $\bar{k}^*_1, \ldots, \bar{k}^*_t$, we would like to predict a new kernel k that performs as good as possible on the base data (X, Y).

We tackle this meta learning problem using a kernel-based approach. As a first step, we need a meta kernel $\mathring{k} : (\mathcal{X}, \mathcal{Y}) \times (\mathcal{X}, \mathcal{Y}) \to \mathbb{R}$. 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 setting one could use information about the features of the transfer and base datasets to construct informative kernels. For instance, if the learning task at hand deals with the classification of text documents, a meta kernel could compare two datasets by counting the tokens that are shared in the two documents. As a default kernel for the case where no usable meta-information is available for the construction of a custom kernel, we propose the *histogram kernel*.

Given two datasets $(\bar{X}, \bar{Y})$ and $(\bar{X}', \bar{Y}')$, where $\bar{X} \in \mathbb{R}^{n \times m}$ and $\bar{X}' \in \mathbb{R}^{n' \times m'}$ are training matrices, the histogram kernel is the sum of the radial basis kernel applied on the difference of the two histograms of feature averages and the radial basis kernel on the difference of the histograms of instance averages. More precisely, the kernel is computed as follows. Assume one is given the two training datasets X and X'. First, we compute the averages $h_c(\bar{X}) = \frac{1}{n} e_n^T X$ and $h_r(\bar{X}) = \frac{1}{m} X e_m$ for $\bar{X}$ and $\bar{X}'$ (e_n denotes the vector of n ones) over the rows and columns of those two datasets. Let h_c, h_c', h_r and h_r' denote the average vectors for $\bar{X}$ and $\bar{X}'$ respectively. In the next step we sort the components in the four vectors by descending size. With that, each vector represents the distribution of feature and instance averages in the datasets, similar to a histogram. Unfortunately, we can not compare the corresponding vectors directly, because the number of columns and rows may differ between X and X' so that the histogram vectors differ in the number of components. Thus, we normalize the histograms h_c, h_c' and h_r, h_r' by duplicating components evenly in the smaller of the two vectors until the two histograms have the same number of components. Finally, we compute the absolute differences between the two corresponding histograms $d_c := \frac{1}{m} \sum_{j=1}^{m} |[h_c - h_c']_j|$ and $d_r := \frac{1}{n} \sum_{j=1}^{n} |[h_r - h_r']_j|$. The final kernel value is then $\mathring{k}((\bar{X}, \bar{Y}), (\bar{X}', \bar{Y}')) := \frac{1}{2}(\exp(-d_c) + \exp(-d_r))$. It is easy to see that $\mathring{k}$ is positive semi-definite, because d_c and d_r constitute the one-norm of the histogram differences. While the histogram kernel is designed as a generic meta kernel for the case where no application-specific choice is available, it appears to work quite well in the experiments in section 4.1. It is an open question, whether it could also be applied in other transfer or meta learning schemes, and which other application-specific meta kernels could provide successful alternatives.

3.3 The Meta Learning Procedure

With this, we have the building blocks to address the meta learning problem. Recall that we wish to learn a meta-model that relates the transfer datasets $(\bar{X}_1, \bar{Y}_1), \ldots, (\bar{X}_t, \bar{Y}_t)$ and the corresponding "known good" kernels $\bar{k}_1^*, \ldots, \bar{k}_t^*$. The model is later used to predict a suitable base kernel for the base dataset. Also recall from section 3.1 that each kernel $\bar{k}_i^*$ is a linear combination $\bar{k}_i^* = \sum_{j=1}^{l} \bar{\mu}_j \bar{k}_j$ of some main kernels $\bar{k}_j$. Thus, the transfer kernels $\bar{k}_1^*, \ldots, \bar{k}_t^*$ are actually represented by the corresponding coefficient vectors $\bar{\mu}_1, \ldots, \bar{\mu}_t$.

For the meta learning problem, we chose a regularized loss minimization approach that resembles a Leave-One-Out-SVM [21]. First of all, it is clear, that the zero-one or hinge losses are not applicable in this setting, because the quantity to predict is a whole vector rather than a binary variable. As a natural replacement, we select the 2-norm to measure the loss between the predicted and true coefficient vector: $l_2(\bar{\mu}, \mathring{f}(\bar{X}, \bar{Y})) := \|\bar{\mu} - \mathring{f}(\bar{X}, \bar{Y})\|_2$. Following the approach of the SVM, we now aim at finding a coefficient vector $\mathring{\alpha} \in \mathbb{R}^t$ and a threshold $\mathring{b} \in \mathbb{R}^l$, which minimize the loss of the kernel classifier $\mathring{f}(X, Y) := \sum_{i=1}^{t} \mathring{k}((\bar{X}_i, \bar{Y}_i), (X, Y))\mathring{\alpha}_i \bar{\mu}_i + \mathring{b}$. For the training of this classifier, though, we follow the philosophy of the Leave-One-Out-SVM and do not consider the contribution of a training example for its own classification. More precisely, when evaluating the classifier given by $(\mathring{\alpha}, \mathring{b})$ on training instance $(\bar{X}_i, \bar{Y}_i)$ during the learning step, we measure the 2-norm loss of the modified classifier $\mathring{f}^{\setminus i}(X, Y) := \sum_{j \neq i} \mathring{k}((\bar{X}_j, \bar{Y}_j), (X, Y))\mathring{\alpha}_j \bar{\mu}_j + \mathring{b}$, which does not incorporate the contribution of the instance it is evaluated on. This ensures that the induced classifier $(\mathring{\alpha}, \mathring{b})$ focuses more on generalization from similar training instances, rather than the rote-learning of dataset-weight associations. In other words, the approach encourages stronger generalization, which is helpful for the typically quite small meta datasets. Altogether, the optimization procedure for the meta-learning task is given by:

$$\underset{\mathring{\alpha} \geq 0, \mathring{b}}{\operatorname{argmin}} \, C \sum_{i=1}^{t} l_2(\bar{\mu}, \mathring{f}^{\setminus i}(\bar{X}, \bar{Y})) + \mathring{\alpha}^T \mathring{D} \mathring{\alpha} \tag{6}$$

Here, $\mathring{D}$ is the meta kernel matrix normalized with the main kernel weight vectors: $\mathring{D}_{ij} = \mathring{k}((\bar{X}_i, \bar{Y}_i), (\bar{X}_j, \bar{Y}_j))\bar{\mu}_i^T \bar{\mu}_j$.

Finally, after the meta classifier is induced from the transfer datasets, we can apply it to the base data (X, Y) to obtain a kernel k. This kernel, in turn, can be expected to work well for the data at hand, because it was derived from the related transfer datasets. In the next section we report results on experiments with the described transfer learning scheme.

4 Experiments

We evaluated the described approach to inductive transfer in two experiments. The first one deals with the prediction of biological activity of molecules, the second with text categorization.

4.1 Structure-Activity Relationship Data

In this experiment, we evaluated kernel-based inductive transfer on the task of learning the biological activity of compounds given their molecular structure as a graph. Learning challenges of this sort are known as structure-activity relationships (SARs), and are common in medicinal chemistry and pharmaceutical research. For the transfer and base data, we chose six datasets from the literature.

Table 1. Results: estimated predictive accuracy of the classifiers on the structure-activity relationship (SAR) data

	Induct. Trans.	Kernel Learn.	Lin 100	Quad 100	RBF 100	Lin 500	Quad. 500	RBF 500	Lin all	Quad all	RBF all
bloodbarr	75.6	70.6●	75.9	75.7	72.8●	72.7●	74.1●	71.3●	71.7●	73.6●	71.3●
factorxa	95.7	93.9●	90.9●	94.9●	78.9●	94.0●	95.0●	71.7●	94.7●	95.5	68.1●
hia	81.8	77.0●	76.7●	77.4●	70.6●	79.1●	79.4●	67.2●	77.3●	79.2●	66.3●
mutagenesis	77.1	71.9●	72.0●	73.7●	67.7●	72.0●	73.6●	57.8●	71.5●	72.8●	56.3●
nctrer	81.6	77.6●	79.3●	78.2●	65.0●	80.0●	79.9●	62.8●	78.5●	80.1●	61.1●
yoshida	72.5	64.9●	63.6●	65.9●	68.8●	67.4●	67.5●	61.0●	67.7●	67.7●	61.0●

The bloodbarr dataset classifies 415 molecules according to the degree to which they can cross the blood-brain barrier [15]. The HIA (Human Intestinal Absorption) dataset contains 164 molecules from various sources, rated according to their oral bio-availability [20]. For the FactorXa set, the task is to discriminate between factor Xa inhibitors of high and low affinity [8]. The NCTRER dataset [7] deals with the prediction of binding activity of small molecules at the estrogen receptor, while the mutagenesis dataset [11] deals with the mutagenicity of compounds. Finally, the Yoshida dataset [23] consists of 265 molecules classified according to their bio-availability.

To transform the graph data into an attribute-value representation, we determined for each dataset the subgraphs that occur in more than a fixed fraction p_{min} of graphs in the set. We lowered the threshold p_{min} for each dataset until around 1,000 non-redundant subgraphs[1] were found and used these subgraphs as binary features ("subgraph occurs/does not occur"). For the resulting datasets, we applied the method described in section 3.1. We randomly chose half of the instances for the estimation of $\bar{f}^*_{\bar{k}}$ and ran Algorithm 1 with $C = 1$ to compute good transfer kernels $\bar{k}^*$. Each such kernel is a linear combination of nine main kernels. For the main kernels, we chose a linear kernel, a quadratic kernel, and a Gaussian kernel with $\sigma = 1$. Since the subgraph features are sorted by size (i.e. number of edges), and since it is known that smaller subgraphs tend to be more informative, we applied the three kernels first on the first 100 features, then on the first 500 features, and finally on all features. The $\bar{k}^*_i$ for each transfer dataset are then linear combinations of these nine main kernels.

Finally, we set one dataset as base learning problem aside and kept the remaining datasets as transfer data. We generated the meta model from this transfer data using the histogram kernel outlined in section 3.2 and the optimization criterion described in section 3.3. The application of the meta model on the base data yields the desired base kernel, which was used in an SVM with $C = 1$ on the base data to learn a classifier. An evaluation of this classifier is given in

[1] Here, "non-redundant" means that we omitted features whose occurrence vector agreed exactly with an already exiting subgraph feature. This step is necessary to avoid the combinatorial explosion, which takes place when a large amount of slight subgraph variants occur frequently in a dataset.

Table 2. Results: estimated predictive accuracy of the classifiers induced on text categorization data

Dataset	Induct. Trans.	Kernel Learn.	Lin 25%	Quad 25%	RBF 25%	Lin 50%	Quad 50%	RBF 50%	Lin all	Quad all	RBF all
10341-14525	55.9	52.2•	49.9•	49.2•	56.6	50.2•	49.2•	57.2	50.1•	49.2•	53.4•
1092-1110	78.5	63.3•	74.6•	51.7•	61.3•	74.6•	51.8•	61.7•	74.9•	51.8•	61.9•
114202-190888	69.3	55.3•	64.6•	52.6•	50.2•	64.6•	52.6•	48.8•	64.9•	52.8•	50.8•
1155181-138526	85.2	64.8•	85.2	57.5•	57.0•	85.2	58.4•	56.1•	85.1	58.3•	55.9•
123412-17899	68.7	66.7	66.3•	64.6•	63.5•	66.3•	64.5•	63.7•	66.3•	64.2•	64.4•
14271-194927	67.1	56.9•	65.3	52.7•	49.4•	65.1	52.4•	49.8•	64.9	52.0•	49.1•
14630-20186	73.4	56.1•	64.7•	44.6•	59.7•	64.6•	45.2•	59.9•	67.5•	45.8•	54.8•
173089-524208	79.8	68.0•	73.5•	63.4•	62.5•	73.7•	63.9•	64.5•	73.8•	63.9•	64.2•
17360-20186	69.5	56.0•	62.8•	51.6•	53.2•	62.6•	51.7•	53.9•	63.1•	51.9•	51.2•
186330-314499	59.3	56.0	52.3•	49.0•	57.4•	52.5•	49.0•	57.3•	52.7•	49.0•	55.9•

Table 1. For comparison, we state the accuracy of a classifier that was induced by kernel learning without the use of the transfer data. To obtain this classifier, we split the training data into two parts of equal size, learn a kernel from the first part of the data and a classifier from the second part. We also give the accuracies of the classifiers induced by the nine main kernels. The estimates are averages over ten ten-fold cross-validation runs. An accuracy estimate for the comparison classifiers is marked with a bullet ("•"), if it is significantly worse than the one for the inductive transfer method according to a paired Wilcoxon rank test at the 5% significance level. The classifiers generated by the presented inductive transfer method were never significantly worse than those induced by the main kernels and by kernel learning. Inductive transfer outperforms kernel learning on all six datasets and it is better than all main kernels on five of them (significantly so on four datasets). Only on the bloodbarr dataset, the classifier constructed by the linear kernel proved to be competitive with the inductive transfer approach.

4.2 Text Data

For the second experiment, we evaluated the proposed method on text categorization data. This domain is well suited for our setting, because the internet provides a wealth of labeled text documents that can be used for inductive transfer. The experiments are based on the datasets from TechTC, the Technion repository of text categorization datasets [9]. Each dataset contains between 150 and 200 web documents, which were obtained by crawling the english language section of a web directory. The binary classification task posed by a dataset consists in telling which category of the web directory a document was taken from. An instance is represented by a vector whose components state the number of occurrences of a word. We sorted the features in descending order from frequently to infrequently occurring words. For the experiments, we randomly chose fifty datasets as transfer data, and selected ten base datasets for evaluation.

As a first step, we applied Algorithm 1 to construct a predictive kernel for each transfer dataset. As before, we used half of the data for classifier estimation and set $C = 1$. The kernels were again represented as linear combinations of nine main kernels $\bar{k}_1, \ldots, \bar{k}_9$. The first three main kernels were applied on only the first quarter of the features (that is, the 25% of most frequently occurring words), the second three main kernels used the 50% most frequently occurring words, and the last three kernels were computed on all features. As in the previous experiment, we had a linear, a quadratic and a Gauss main kernel for each feature (sub-)set. In the next step, we computed the meta kernel matrix for the fifty transfer datasets. The meta kernel was computed based on the overlap of word features between two datasets. More precisely, we chose $\mathring{k}(\bar{X}_1, \bar{X}_2) := |\bar{W}_1 \cap \bar{W}_2| / \max(|\bar{W}_1|, |\bar{W}_2|)$, where $\bar{W}_1$ and $\bar{W}_2$ denote the set of words in $\bar{X}_1$ and $\bar{X}_2$, respectively.

Plugging this meta kernel into the optimization criterion (6), we obtain a meta classifier that predicts a kernel for each base dataset. Table 2 gives an evaluation of the predictive accuracy of the classifiers that were induced with those base kernels (setting $C = 1$ for the base SVM). As before, we give also the predictive performance of a kernel induced from the training data and the single main kernels. All estimates are averaged over ten runs of tenfold-cross validation. An estimate is marked with a bullet ("•"), if it is significantly worse than the estimate for the inductive transfer method according to a paired Wilcoxon rank test on the 5% significance level, and with a circle ("◦"), if it is better. The inductive transfer method outperformed the kernel learning on all datasets (significantly so on eight out of the ten) and it was better than all main kernels in nine cases (in seven of them significantly), a clear indication that the predicted base kernels provide indeed a good bias for text categorization tasks on web documents.

5 Conclusion

In this paper, we described a kernel-based approach to inductive transfer. In this setting, the main goal is to make use of existing datasets to increase the predictive performance of classifiers induced on a new dataset. We framed the problem of finding a good bias for the new data as a meta learning problem: Given the transfer data, what should a kernel for the new data look like? We proposed a kernel-based approach to this problem. First, we presented an iterative procedure to find transfer kernels, which encode the bias necessary to perform well on the transfer datasets. Second, we introduced a convex optimization criterion for the meta learning problem to predict a suitable base kernel for the new data. Third, we described the histogram kernel, a general purpose kernel for the meta learning task. Of course, the work can be extended in various directions. For example, it would be interesting to investigate other meta kernels that use more (possibly domain specific) knowledge about the datasets.

References

1. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems 19. Advances in Neural Information Processing Systems, pp. 41–48. MIT Press, Cambridge (2006)
2. Baxter, J.: A model of inductive bias learning. Journal of Artificial Intelligence Research 12, 149–198 (2000)
3. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)
4. Crammer, K., Keshet, J., Singer, Y.: Kernel design using boosting. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 15, pp. 537–544. MIT Press, Cambridge (2002)
5. Erhan, D., Bengio, Y., L'Heureux, P.-J., Yue, S.Y.: Generalizing to a zero-data task: a computational chemistry case study. Technical Report 1286, Département d'informatique et recherche opérationnelle, Université de Montréal (2006)
6. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. J. Mach. Learn. Res. 6, 615–637 (2005)
7. Fang, H., Tong, W., Shi, L.M., Blair, R., Perkins, R., Branham, W., Hass, B.S., Xie, Q., Dial, S.L., Moland, C.L., Sheehan, D.M.: Structure-activity relationships for a large diverse set of natural, synthetic, and environmental estrogens. Chemical Research in Toxicology 14(3), 280–294 (2001)
8. Fontaine, F., Pastor, M., Zamora, I., Sanz, F.: Anchor-GRIND: Filling the gap between standard 3D QSAR and the grid-independent descriptors. Journal of Medicinal Chemistry 48(7), 2687–2694 (2005)
9. Gabrilovich, E., Markovitch, S.: Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In: Proceedings of The 27th Annual International ACM SIGIR Conference, Sheffield, UK, pp. 250–257. ACM Press, New York (2004)
10. Girolami, M., Rogers, S.: Hierarchic Bayesian models for kernel learning. In: ICML 2005: Proceedings of the 22nd international conference on Machine learning, pp. 241–248. ACM Press, New York (2005)
11. Helma, C., Cramer, T., Kramer, S., De Raedt, L.: Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. Journal of Chemical Information and Computer Sciences 44(4), 1402–1411 (2004)
12. Hertz, T., Hillel, A.B., Weinshall, D.: Learning a kernel function for classification with small training samples. In: ICML 2006: Proceedings of the 23rd international conference on Machine learning, pp. 401–408. ACM, New York (2006)
13. Kaski, S., Peltonen, J.: Learning from relevant tasks only. In: Kok, J.N., Koronacki, J., de Mántaras, R.L., Matwin, S., Mladenic, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 608–615. Springer, Heidelberg (2007)
14. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., El Ghaoui, L., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. J. Mach. Learn. Res. 5, 27–72 (2004)
15. Li, H., Yap, C.W., Ung, C.Y., Xue, Y., Cao, Z.W., Chen, Y.Z.: Effect of selection of molecular descriptors on the prediction of blood-brain barrier penetrating and nonpenetrating agents by statistical learning methods. Journal of Chemical Information and Modeling 45(5), 1376–1384 (2005)
16. Micchelli, C.A., Pontil, M.: Learning the kernel function via regularization. J. Mach. Learn. Res. 6, 1099–1125 (2005)

17. Ong, C.S., Smola, A.J., Williamson, R.C.: Learning the kernel with hyperkernels. J. Mach. Learn. Res. 6, 1043–1071 (2005)
18. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.G.: Meta-learning by landmarking various learning algorithms. In: ICML 2000: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 743–750. Morgan Kaufmann, San Francisco (2000)
19. Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. J. Mach. Learn. Res. 7, 1531–1565 (2006)
20. Wegner, J.K., Fröhlich, H., Zell, A.: Feature selection for descriptor based classification models. 1. theory and ga-sec algorithm. Journal of Chemical Information and Modeling 44(3), 921–930 (2004)
21. Weston, J., Herbrich, R.: Adaptive margin support vector machines. In: Advances in Large-Margin Classifiers, pp. 281–295. MIT Press, Cambridge (2000)
22. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with dirichlet process priors. J. Mach. Learn. Res. 8, 35–63 (2007)
23. Yoshida, F., Topliss, J.: QSAR model for drug human oral bioavailability. J. Med. Chem. 43, 2575–2585 (2000)
24. Yu, K., Tresp, V., Schwaighofer, A.: Learning gaussian processes from multiple tasks. In: ICML 2005: Proceedings of the 22nd international conference on Machine learning, pp. 1012–1019. ACM Press, New York (2005)