Projection Penalties: Dimension Reduction without Loss

Yi Zhang yizhang1@cs.cmu.edu
Machine Learning Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA.

Jeff Schneider schneide@cs.cmu.edu
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA.

Abstract

Dimension reduction is popular for learning predictive models in high-dimensional spaces. It can highlight the relevant part of the feature space and avoid the curse of dimensionality. However, it can also be harmful because any reduction loses information. In this paper, we propose the *projection penalty* framework to make use of dimension reduction without losing valuable information.

Reducing the feature space before learning predictive models can be viewed as restricting the model search to some parameter subspace. The idea of projection penalties is that instead of restricting the search to a parameter subspace, we can search in the full space but penalize the projection distance to this subspace. Dimension reduction is used to guide the search, rather than to restrict it.

We propose projection penalties for linear dimension reduction, and then generalize to kernel-based reduction and other nonlinear methods. We test projection penalties with various dimension reduction techniques in different prediction tasks, including principal component regression and partial least squares in regression tasks, kernel dimension reduction in face recognition, and latent topic modeling in text classification. Experimental results show that projection penalties are a more effective and reliable way to make use of dimension reduction techniques.

1. Introduction

Learning a model in high-dimensional spaces with limited training examples is a difficult task. As a result, researchers have invented various dimension reduction techniques to reduce the feature space. A dimension reduction method may focus on feature selection or feature extraction, can be linear or nonlinear, may utilize the target variable (supervised) or not (unsupervised), and may be specifically designed for regression or classification. Despite the wide variety of approaches, dimension reduction is useful for learning predictive models because it can discover and highlight the relevant part of the feature space in fewer dimensions. However, performing dimension reduction and then learning in the reduced space can also degrade the prediction performance, because any reduction loses information. Indeed, it is difficult to tell whether the information lost by a dimension reduction procedure is relevant to a prediction task or not.

In this paper, we propose *projection penalties* to enable predictive modeling to gain from dimension reduction techniques *without* losing relevant information. A reduction of the feature space can be viewed as a restriction of model search to a *parameter subspace*. The basic idea of projection penalties is that, instead of restricting model search to a parameter subspace, we can still search in the full parameter space but penalize the projection distance to this subspace. As a result, dimension reduction is used to *guide* the model search in the parameter space, rather than to *restrict* it.

This paper is organized as follows. Section 2 proposes the projection penalty framework. In Section 2.1 we discuss the connection between a linear dimension reduction and the corresponding parameter subspace. In Section 2.2 we introduce projection penalties for linear dimension reduction. In Section 2.3 we generalize projection penalties to kernel-based dimension reduction. Then in Section 2.4 and 2.5, we discuss two extensions to our framework: regularization in the parame-

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

ter subspace and adaptation to an arbitrary nonlinear reduction technique. Section 3 presents our experimental results. In Section 4 we discuss related work. Section 5 concludes the paper.

2. Projection Penalties

In this section we propose the projection penalty framework, which enables us to use a given dimension reduction to improve predictive modeling without the risk of losing relevant information.

2.1. Dimension reduction and parameter subspace

Consider learning a linear prediction model $(\mathbf{w}, b)$ in a p-dimensional space by minimizing an empirical loss L given a set of n training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$:

$$\operatorname*{argmin}_{\mathbf{w}\in R^p, b} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) \tag{1}$$

where the parameter vector $\mathbf{w} \in R^p$ and the intercept b represent the prediction model, and $\mathbf{x}_i$ and y_i are the p-dimensional feature vector and the response variable of the ith example, respectively. The empirical loss L depends on the choice of prediction models, e.g., squared error loss, logistic log-likelihood loss or hinge loss. If the dimension p of the feature space is high, a penalty term $J(\mathbf{w})$ may be added to eq. (1) to regularize the model complexity, such as an ℓ2-norm penalty in ridge regression (Tikhonov & Arsenin, 1977) and support vector machines (Vapnik, 1995) and an ℓ1-norm penalty in lasso (Tibshirani, 1996).

A linear reduction of a p-dimensional feature space can be represented as a $d \times p$ matrix P where $d < p$ is the dimension of the reduced feature space. For an example $\mathbf{x}$ in the original feature space, $P\mathbf{x}$ is the representation in the reduced space. In this sense, performing a linear dimension reduction and then learning predictive models in the reduced space can be written as:

$$\operatorname*{argmin}_{\mathbf{v}\in R^d, b} \sum_{i=1}^{n} L(y_i, \mathbf{v}^T(P\mathbf{x}_i) + b) \tag{2}$$

where $\mathbf{v} \in R^d$ is the parameter vector learned in the reduced feature space. Eq. (2) can be rewritten as:

$$\operatorname*{argmin}_{\mathbf{v}\in R^d, b} \sum_{i=1}^{n} L(y_i, (P^T\mathbf{v})^T\mathbf{x}_i + b) \tag{3}$$

Comparing eq. (3) with eq. (1), we see the connection between a linear reduction of the feature space and the restriction on the model parameter space. Specifically, performing a linear reduction P in the feature space simply corresponds to confining the p-dimensional parameter vector $\mathbf{w}$ to the following subspace:

$$\mathcal{M}_P = \{\mathbf{w} \in R^p \mid \mathbf{w} = P^T\mathbf{v},\ \exists \mathbf{v} \in R^d\} \tag{4}$$

In this sense, eq. (2) is equivalent to:

$$\operatorname*{argmin}_{\mathbf{w}\in \mathcal{M}_P, b} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) \tag{5}$$

2.2. Projection penalties for linear reduction

In eq. (5) we see that performing a linear dimension reduction P in the feature space is equivalent to restricting the model search to a parameter subspace $\mathcal{M}_P$ as defined in eq. (4). From the perspective of model search, we eliminate all candidates that are not in $\mathcal{M}_P$. This is risky since there is no guarantee that the optimal model in R^p for a prediction task will belong to the given subspace $\mathcal{M}_P$.

We propose to search models in the full parameter space and penalize the projection distance to $\mathcal{M}_P$. This leads to the formulation of a projection penalty for a linear dimension reduction P:

$$\operatorname*{argmin}_{\mathbf{w}\in R^p, b} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) + \min_{\mathbf{w}^*\in \mathcal{M}_P} \lambda J(\mathbf{w} - \mathbf{w}^*)$$

or using the definition of $\mathcal{M}_P$ in (4), we have:

$$\operatorname*{argmin}_{\mathbf{w}\in R^p, b} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) + \min_{\mathbf{v}\in R^d} \lambda J(\mathbf{w} - P^T\mathbf{v}) \tag{6}$$

where λ is a regularization parameter, $J()$ is a penalty function such as $\|\|_2^2$ or $\|\|_1$, $\mathbf{v} \in R^d$ is a parameter vector on the reduced feature space and $\mathbf{w}^* = P^T\mathbf{v}$ defines a member in the parameter subspace $\mathcal{M}_P$. Eq. (6) allows us to learn the parameter vector $\mathbf{w}$ in the full parameter space R^p and penalize the part of $\mathbf{w}$ that can *not* be interpreted by models in $\mathcal{M}_P$. This is different from directly performing a dimension reduction P, as shown in eq. (5), where model search is completely restricted to the parameter subspace $\mathcal{M}_P$.

It is straightforward to solve eq. (6) by a change of notation $\tilde{\mathbf{w}} = \mathbf{w} - P^T\mathbf{v}$:

$$\operatorname*{argmin}_{\tilde{\mathbf{w}}\in R^p, \mathbf{v}\in R^d, b} \sum_{i=1}^{n} L(y_i, \tilde{\mathbf{w}}^T\mathbf{x}_i + \mathbf{v}^T(P\mathbf{x}_i) + b) + \lambda J(\tilde{\mathbf{w}}) \tag{7}$$

This can viewed as redefining the representation for each training example $\mathbf{x}_i$ as a $(p + d)$-dimensional vector $[\mathbf{x}_i; (P\mathbf{x}_i)]$, and solving the regularized linear model

$([\tilde{\mathbf{w}}; \mathbf{v}], b)$ on the new representation. The specific algorithm for solving eq. (7) depends on the choice of empirical loss L and penalty function J, e.g., conjugate gradient methods, subgradient methods, or methods for constrained optimization (Boyd & Vandenberghe, 2004). The parameter vector $\mathbf{w}$ can be computed as:

$$\mathbf{w} = \tilde{\mathbf{w}} + P^T \mathbf{v}$$

2.3. Projection penalties for kernel-based reduction

Section 2.2 shows that a linear dimension reduction can be used to penalize a linear model via a projection penalty, as in eq. (6). A limitation of eq. (6) is that the resulting model is still a linear model. Recently, SVMs and other kernel-based learning algorithms have drawn considerable interest in machine learning (Muller et al., 2001). In this section, we generalize projection penalties to dimension reduction and predictive modeling in kernel feature spaces.

Consider learning a prediction model $\mathbf{w}$ given a set of n training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and a dimension reduction operator P, where both the prediction model and the reduction operator are designed to operate on a kernel feature space $\mathcal{F}$. Each training example is represented as $\Phi(\mathbf{x}_i) \in \mathcal{F} \subseteq R^p$ using the kernel feature mapping Φ. The feature mapping Φ is characterized by its inner product via a Mercer kernel $k(\cdot, \cdot)$:

$$(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j)$$

Depending on the choice of kernel functions, the kernel feature space is usually a very high (or infinite) dimensional space and thus p is large or infinite.

Analogously to eq. (6), the projection penalty is:

$$\operatorname*{argmin}_{\mathbf{w} \in R^p, b} \sum_{i=1}^n L(y_i, \mathbf{w}^T \Phi(\mathbf{x}_i) + b) + \min_{\mathbf{v} \in R^d} \lambda J(\mathbf{w} - P^T \mathbf{v}) \tag{8}$$

where $\mathbf{w}$, $\mathbf{v}$ and P are of size $p \times 1$, $d \times 1$ and $d \times p$, respectively. The reduction operator P is provided by a kernel-based dimension reduction such as kernel PCA (Schölkopf et al., 1998) or generalized discriminant analysis (Baudat & Anouar, 2000). Thus, each $p \times 1$ column basis of P^T is represented by a weighted combination of examples in kernel feature space $\mathcal{F}$.[1]

To solve $\mathbf{w}$ in eq. (8), we consider classification problems and use the hinge loss of SVMs as the empirical loss $L()$ and the $\ell 2$-norm penalty as $J()$. By introducing slack variables $\{\xi_i\}_{i=1}^n$ for hinge loss (Burges, 1998), we have the following quadratic program:

$$\begin{aligned} &\operatorname*{argmin}_{\mathbf{w} \in R^p, \mathbf{v} \in R^d, b, \{\xi_i\}_{i=1}^n} \frac{1}{2} \parallel \mathbf{w} - P^T \mathbf{v} \parallel_2^2 + C \sum_{i=1}^n \xi_i \\ &\text{s.t.} \quad y_i(\mathbf{w}^T \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i \\ &\qquad\;\; \xi_i \geq 0, \quad \forall i \end{aligned} \tag{9}$$

where we use $C \propto \frac{1}{\lambda}$ to replace λ in eq. (8). Also by changing to $\tilde{\mathbf{w}} = \mathbf{w} - P^T \mathbf{v}$, we have:

$$\begin{aligned} &\operatorname*{argmin}_{\tilde{\mathbf{w}} \in R^p, \mathbf{v} \in R^d, b, \{\xi_i\}_{i=1}^n} \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + C \sum_{i=1}^n \xi_i \\ &\text{s.t.} \quad y_i(\tilde{\mathbf{w}}^T \Phi(\mathbf{x}_i) + \mathbf{v}^T P \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \forall i \\ &\qquad\;\; \xi_i \geq 0, \quad \forall i \end{aligned} \tag{10}$$

Note that the reduction of examples from $\mathcal{F}$ into d-dimensional space, i.e., $P\Phi(\mathbf{x}_i)$, is actually performed via kernel tricks (Muller et al., 2001).

Since the primal problem (10) is convex, the KKT conditions are sufficient and necessary for optimal primal and dual solutions (Boyd & Vandenberghe, 2004). To derive KKT conditions and the dual, we add Lagrangian multipliers $\{\alpha_i\}_{i=1}^n$ and $\{\mu_i\}_{i=1}^n$ for the constraints in the primal (10), which gives the Lagrangian:

$$\begin{aligned} \mathcal{L} \;=\; & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i y_i (\tilde{\mathbf{w}}^T \Phi(\mathbf{x}_i) + \mathbf{v}^T P \Phi(\mathbf{x}_i) + b) \\ & - \sum_{i=1}^n \alpha_i (\xi_i - 1) - \sum_{i=1}^n \mu_i \xi_i \\ & \text{s.t. } \alpha_i \geq 0, \, \mu_i \geq 0, \, \forall i \end{aligned}$$

The KKT optimality conditions include:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{w}}} = \tilde{\mathbf{w}} - \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i) &= 0 \qquad (11) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{v}} = -\sum_{i=1}^n \alpha_i y_i P \Phi(\mathbf{x}_i) &= 0 \qquad (12) \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^n \alpha_i y_i &= 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \mu_i &= 0 \\ y_i(\tilde{\mathbf{w}}^T \Phi(\mathbf{x}_i) + \mathbf{v}^T P \Phi(\mathbf{x}_i) + b) - 1 + \xi_i &\geq 0 \\ \xi_i &\geq 0 \\ \alpha_i &\geq 0 \\ \mu_i &\geq 0 \\ \alpha_i [y_i(\tilde{\mathbf{w}}^T \Phi(\mathbf{x}_i) + \mathbf{v}^T P \Phi(\mathbf{x}_i) + b) - 1 + \xi_i] &= 0 \\ \mu_i \xi_i &= 0 \end{aligned}$$

[1] These can be the same set of examples $\{\Phi(\mathbf{x}_i)\}_{i=1}^n$ for learning $\mathbf{w}$, or a different set of examples in $\mathcal{F}$.

Using the above conditions to eliminate primal variables in the Lagrangian, we have the dual problem:

$$\begin{aligned}\underset{\{\alpha_i\}_{i=1}^n}{\text{argmax}} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) && (13)\\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad \forall i \\ & \sum_{i=1}^n \alpha_i y_i = 0 && (14)\\ & \sum_{i=1}^n \alpha_i y_i P\Phi(\mathbf{x}_i) = 0 && (15)\end{aligned}$$

where the kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ computes $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. This dual can be solved as a quadratic programming problem, and $\tilde{\mathbf{w}}$, according to condition (11), is:

$$\tilde{\mathbf{w}} = \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i) \qquad (16)$$

Note that the dual form (13) is similar to the dual form of SVMs (Burges, 1998), with one key difference: the SVM dual form has only one equality constraint (14), while the above dual form has additional d equality constraints (15). These d additional constraints basically say that dual solutions $\{\alpha_i\}_{i=1}^n$ and the resulting $\tilde{\mathbf{w}}$ do *not* operate on the reduced feature space (w.r.t. the training examples). This is very intuitive: $\tilde{\mathbf{w}}$ in the primal form (10) is defined by a change of variable: $\tilde{\mathbf{w}} = \mathbf{w} - P^T\mathbf{v}$. Thus, $\tilde{\mathbf{w}}$ refers to the part of $\mathbf{w}$ that can not be replaced by operating on the reduced feature space. This also coincides with the fact that the constraints (15) result from the KKT condition (12), which says the derivative of the Lagrangian w.r.t. $\mathbf{v} \in R^d$ is zero. Indeed, if $\tilde{\mathbf{w}}$ is still operating on the reduced space, then $\mathbf{v} \in R^d$ should be changed to take the responsibility of $\tilde{\mathbf{w}}$ since $\mathbf{v}$ is not penalized in the primal form (10).

To solve for $\mathbf{v}$ and b, we plug the solution to $\tilde{\mathbf{w}}$ (solved as (16)) into the primal form (10). The quadratic term becomes a constant, and therefore, $\mathbf{v}$ and b (and slack variables $\{\xi_i\}_{i=1}^n$) can be solved efficiently in (10) as a linear programming problem. Note that both $\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i)$ and $P\Phi(\mathbf{x}_i)$ in the constraints are computed via kernel tricks and treated as known. Finally, given $\tilde{\mathbf{w}}$, $\mathbf{v}$ and b, the prediction on a new example $\mathbf{x}$ is:

$$\tilde{\mathbf{w}}^T\Phi(\mathbf{x}) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b$$

2.4. Regularization in the parameter subspace

Section 2.2 and 2.3 propose projection penalties for linear dimension reduction and kernel-based dimension reduction. In both cases, there is no penalty on the parameter vector $\mathbf{v} \in R^d$ that operates on the reduced feature space. In some applications where the number of training examples is very small or the dimensionality of the reduced space is relatively high, a small penalty on $\mathbf{v} \in R^d$ will be helpful for both the generalization of the model and the numerical stability for optimization. Including a small penalty on $\mathbf{v} \in R^d$ into eq. (6) or eq. (7) is straightforward and will not affect the optimization algorithm. But for projection penalties in kernel feature space, such as in eq. (10), including another penalty will change the dual form in a non-trivial way, and thus is the focus of this section.

We start from eq. (10) and add a new penalty on $\mathbf{v}$:

$$\begin{aligned}\underset{\tilde{\mathbf{w}}\in R^p, \mathbf{v}\in R^d, b, \{\xi_i\}_{i=1}^n}{\text{argmin}} \quad & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + \frac{\gamma}{2} \parallel \mathbf{v} \parallel_2^2 + C\sum_{i=1}^n \xi_i && (17)\\ \text{s.t.} \quad & y_i(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b) \ge 1 - \xi_i, \forall i \\ & \xi_i \ge 0, \quad \forall i\end{aligned}$$

where $0 \le \gamma \le 1$ is a new regularization parameter. Adding Lagrangian multipliers $\{\alpha_i\}_{i=1}^n$ and $\{\mu_i\}_{i=1}^n$ for the two sets of constraints, the Lagrangian is:

$$\begin{aligned}\mathcal{L} \quad = \quad & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + \frac{\gamma}{2} \parallel \mathbf{v} \parallel_2^2 + C\sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i y_i(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b) \\ & - \sum_{i=1}^n \alpha_i(\xi_i - 1) - \sum_{i=1}^n \mu_i \xi_i \\ & \text{s.t. } \alpha_i \ge 0, \, \mu_i \ge 0, \, \forall i\end{aligned}$$

Most KKT conditions remain the same as in Section 2.3, except condition (12), which is now:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \gamma\mathbf{v} - \sum_{i=1}^n \alpha_i y_i P\Phi(\mathbf{x}_i) = 0 \qquad (18)$$

Given the new Lagrangian and the new set of KKT conditions, the dual becomes:

$$\begin{aligned}\underset{\{\alpha_i\}_{i=1}^n}{\text{argmax}} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) && (19)\\ & - \frac{1}{2\gamma}\sum_{i,j}\alpha_i\alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T(P\Phi(\mathbf{x}_j)) \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad \forall i \\ & \sum_{i=1}^n \alpha_i y_i = 0\end{aligned}$$

Comparing this dual form to the dual (13), the constraints (15) are removed, and instead, the quadratic term in the objective now includes a new part: $-\frac{1}{2\gamma}\sum_{i,j}\alpha_i\alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T(P\Phi(\mathbf{x}_j))$. We can see the connection between duals (19) and (13) by setting $\gamma \to 0$. In this case, the new dual has an infinite weight on $-\sum_{i,j}\alpha_i\alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T(P\Phi(\mathbf{x}_j))$. The matrix of inner products is positive semidefinite. Therefore, to maximize objective (19), the solution $\{\alpha_i\}_{i=1}^n$ must satisfy the constraints (15) in order to keep the infinitely weighted non-positive term $-\sum_{i,j}\alpha_i\alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T(P\Phi(\mathbf{x}_j))$ as zero. In this sense, the dual (19) comes back to (13) when $\gamma \to 0$.

Given the solution $\{\alpha_i\}_{i=1}^n$ to the dual (19), $\tilde{\mathbf{w}}$ and $\mathbf{v}$ are obtained via KKT conditions (11) and (18):

$$\begin{aligned}\tilde{\mathbf{w}} &= \sum_{i=1}^{n}\alpha_i y_i \Phi(\mathbf{x}_i)\\ \mathbf{v} &= \frac{1}{\gamma}\sum_{i=1}^{n}\alpha_i y_i P\Phi(\mathbf{x}_i)\end{aligned}$$

Finally, b can be obtained similarly as in SVMs (Burges, 1998) via examples that lie on the margin (i.e., examples with $0 < \alpha_i < C$).

2.5. Adaption to other nonlinear reduction

In this section, we extend projection penalties to work with an arbitrary reduction operation. Recall that projection penalties proposed in Section 2.2 and 2.3 requires the reduction operator P to be either linear in the input space or at least "linear" in the kernel feature space. This requirement is the basis for defining the parameter subspace $\mathcal{M}_P$ in eq. (4), which is then used to formulate eq. (6) and eq. (8). However, when we proceed to eq. (7) and eq. (10), the parts that involve the reduction operator P are just $P\mathbf{x}_i$ in (7) and $P\Phi(\mathbf{x}_i)$ in (10). Therefore, a simple trick is to replace these two terms by an arbitrary reduction function $\Psi(\mathbf{x}_i)$. Note that eq. (7) and (10) with this trick are no longer connected to eq. (6) and (8), but they can still be solved as discussed in Section 2.2 and 2.3. As long as we can apply the same function $\Psi(\mathbf{x})$ to any new example $\mathbf{x}$, this trick will work for prediction.

This trick may be very useful for some application domains. Consider a fully probabilistic topic model such as latent Dirichlet allocation (Blei et al., 2003). The reduction operation is neither linear in the input space nor linear in any kernel space. In fact, extracting a topic distribution for a given document via latent Dirichlet allocation involves intractable inference of posterior distribution. In this sense, the trick allows us to gain from a topic model when learning either linear classifiers or kernel-based classifiers on text.

For learning a prediction model in a kernel space, this trick also enables us to use a different kernel or the same kernel with different kernel parameters for the dimension reduction operation. This is not allowed in Section 2.3, since the kernel-based reduction P in eq. (8) was assumed to operate on exactly the same kernel feature space as the prediction model $\mathbf{w}$.

Table 1. R^2 for predicting Boston housing prices: means and standard errors over 500 random runs.

	R^2: Mean	R^2: Standard Error
Ridge	49.53%	0.98%
PCR$(d = 11)$	52.93%	0.96%
PLS$(d = 11)$	52.58%	0.97%
Proj-PCR$(d = 4)$	53.55%	0.68%
Proj-PLS$(d = 1)$	53.63%	0.72%

3. Empirical Studies

In this section, we present our empirical studies on applying projection penalties to different dimension reduction techniques for housing price prediction, text classification, and face recognition.

3.1. Linear Reduction in Regression

Experiment Settings. We use the Boston Housing data set from the UCI Machine Learning Repository[2] as an example regression task. The data set contains 506 examples, each corresponding to a U.S. census tract in the Boston area. The target variable is the median value of owner-occupied homes and the 13 input variables include crime rate, student-teacher ratio, tax rate, and so forth. We conducted 500 random runs in our experiments. In each run, we randomly sample 50 training examples and the remaining 456 records are used as testing examples. We use R^2, one minus the proportion of unexplained variance, as the performance measure for this regression task.

Competing Methods. We study the following methods. 1) Ridge regression (*Ridge*) trained on the original 13 variables. 2) Principal component regression (*PCR*), which first performs dimension reduction using PCA and then fits a least squares model on the principal components. 3) Partial least squares (*PLS*), which can be viewed as sequentially extracting a number of components using both input variables and the response variable and fitting a least squares model on the extracted components. 4) Projection penalty (*Proj-PCR*) in eq. (6) with PCA as the dimension reduction P and least squares as the loss function L. 5) Projection penalty (*Proj-PLS*) in eq. (6) using a

[2] http://www.ics.uci.edu/∼mlearn/MLRepository.html

trained partial least squares model to provide dimension reduction and least squares as the loss function. For ridge regression and the projection penalty models, the regularization parameter λ is chosen by 5-fold cross-validation from the range $[10^{-8}, 10^{-6}, \ldots, 10^{10}]$. Also, for the projection penalty models, $\frac{\lambda}{1000}||\beta||_2^2$ is added to penalize the model in the reduced space, as discussed in Section 2.4. The dimension of the reduced space d is chosen for each model to optimize the performance.

Results. The experimental results are shown in Table 1. For each model, we report the average R^2 over 500 random runs as well as the standard errors. The optimal dimension of the reduced space is also displayed with each model (if applicable). From the results we can see that dimension reduction techniques are helpful for predicting the house prices. Both principal component regression (PCR) and partial least squares (PLS) perform better than ridge regression. Projection penalty models (Proj-PCR and Proj-PLS) further improve the predictions.

3.2. Topic Modeling in Text Classification

Experiment Settings. We use the 20-Newsgroups data set[3], which contains 11314 training and 7532 testing documents collected from 20 newsgroups. We denote training and testing sets as D_{tr} and D_{ts}, respectively. Documents are represented as bags of words. We select the most frequent 200 words in each newsgroup except the 20 most frequent common words across all newsgroups. This leads to $p = 1443$ words (i.e., features) in the vocabulary. Several latent Dirichlet allocation (Blei et al., 2003) models are estimated from D_{tr} and used as dimension reduction models, with the number of topics d specified as $10, 20, 30, 50$, and 100. We construct 190 binary classification tasks using all pairs of newsgroups. For each task, we randomly sample 2%, 5% and 10% of the relevant documents in D_{tr} as training examples for classifiers. We use the average classification error over 190 tasks as the performance measure. This procedure is repeated for 20 random runs. The testing documents for each task are fixed as all relevant documents in D_{ts}.

Competing Methods. We consider three methods. 1) $\ell2$-regularized logistic regression (L2-LGR) in the original feature space: we directly train a logistic regression classifier for each task, and use $\ell2$ regularization to prevent overfitting in the high-dimensional feature (word) space. 2) Logistic regression in the topic space (LGR-Topic): we use latent Dirichlet allocation to reduce the feature space (into topic space), and then train a logistic regression for each task in the topic space. 3) Projection penalty (Proj-Topic) as in eq. (7) with latent Dirichlet allocation as the dimension reduction operator. Note that the trick discussed in Section 2.5 is needed since extracting the topic distribution is not a linear reduction of the word space. The parameter λ in the $\ell2$ regularization is chosen via 5-fold stratified cross-validation from the range $[10^{-8}, 10^{-6}, \ldots, 10^{10}]$. For methods using dimension reduction, $d = 30$ topics gives the best performance.

Table 2. Text classification error averaged over tasks (2% train data): mean and standard error over 20 random runs

	Mean	Standard Error
L2-LGR	17.78%	0.082%
LGR-Topic$(d = 30)$	12.51%	0.059%
Proj-Topic$(d = 30)$	10.36%	0.067%

Table 3. Text classification error averaged over tasks (5% train data): mean and standard error over 20 random runs

	Mean	Standard Error
L2-LGR	11.08%	0.039%
LGR-Topic$(d = 30)$	10.45%	0.049%
Proj-Topic$(d = 30)$	7.87%	0.042%

Table 4. Text classification error averaged over tasks (10% train data): mean and standard error over 20 random runs

	Mean	Standard Error
L2-LGR	8.30%	0.029%
LGR-Topic$(d = 30)$	9.39%	0.027%
Proj-Topic$(d = 30)$	6.77%	0.029%

Results. Empirical results on average classification errors over tasks are shown in Tables 2, 3 and 4, with 2%, 5% and 10% training examples used. Means and standard errors over 20 random runs are reported. Using a projection penalty with a topic model (*Proj-Topic*) achieves the best performance in all sizes of training sets. The performance is better than both directly fitting regularized models in the word space (*L2-LGR*) and fitting models in the reduced topic space (*LGR-Topic*). It is interesting to compare the performance of *L2-LGR* and *LGR-Topic* when the training size varies. Using 2% of the training examples, learning classifiers on topics performs significantly better than fitting models in the original word space, as the topic space highlights important information in a lower dimensionality. However, when the training size is increased to 10%, directly fitting a model on the word distribution is superior to learning models in the reduced topic space. This confirms that the dimension

[3] http://people.csail.mit.edu/jrennie/20newsgroups

Table 5. Face classification error averaged over tasks (3 training images per subject): mean and standard error over 50 random runs

	Mean	Standard Error
SVM-Kernel	16.17%	0.26%
SVM-KPCA($d = 45$)	16.22%	0.25%
Proj-KPCA($d = 45$)	15.33%	0.26%
SVM-GDA($d = 14$)	11.21%	0.25%
Proj-GDA($d = 14$)	10.38%	0.24%
SVM-OLap($d = 14$)	8.58%	0.26%
Proj-OLap($d = 14$)	7.93%	0.22%

Table 6. Face classification error averaged over tasks (5 training images per subject): mean and standard error over 50 random runs

	Mean	Standard Error
SVM-Kernel	12.68%	0.30%
SVM-KPCA($d = 75$)	12.72%	0.30%
Proj-KPCA($d = 75$)	12.47%	0.29%
SVM-GDA($d = 14$)	9.32%	0.23%
Proj-GDA($d = 14$)	7.13%	0.20%
SVM-OLap($d = 14$)	4.74%	0.21%
Proj-OLap($d = 14$)	4.57%	0.20%

Table 7. Face classification error averaged over tasks (7 training images per subject): mean and standard error over 50 random runs

	Mean	Standard Error
SVM-Kernel	11.43%	0.28%
SVM-KPCA($d = 105$)	11.37%	0.29%
Proj-KPCA($d = 105$)	11.19%	0.28%
SVM-GDA($d = 14$)	7.59%	0.23%
Proj-GDA($d = 14$)	6.37%	0.23%
SVM-OLap($d = 14$)	3.43%	0.17%
Proj-OLap($d = 14$)	3.40%	0.17%

reduction also loses valuable information. Finally, the projection penalty with a topic model dominates the other two methods for all training sizes, which indicates the projection penalty improves on topic models and also avoids the loss of information.

3.3. Kernel Methods in Face Recognition

Experiment Settings. We use the Yale face data set[4], which contains face images of 15 subjects. There are 11 images per subject, corresponding to different configurations in terms of expression, emotion, illumination, and wearing glasses (or not), etc. Each image is scaled to 32 × 32 pixels. We vary the size of the training set as $3, 5$ and 7 images per subject. We have 50 random runs for each size of training sets. In each run, we randomly select training examples and the rest is used as the testing set. Training examples of all subjects are used to learn dimension reduction models (discussed later). We consider all 105 binary classification tasks, each classifying between two subjects. The average classification error over tasks is the performance measure, and aggregated results over 50 runs are reported.

In this experiment, we study dimension reduction and predictive modeling in a kernel feature space. We use the degree-2 polynomial kernel to define our kernel feature space. We consider three dimension reduction methods: kernel PCA (Schölkopf et al., 1998), generalized discriminant analysis (GDA) (Baudat & Anouar, 2000) and orthogonal Laplacian faces (OLap) (Cai et al., 2006).

Competing Methods. We compare the following methods. 1) SVM in the kernel feature space (*SVM-Kernel*): train an SVM classifier for each task with a polynomial kernel. This can be viewed as fitting a linear model directly in the high-dimensional kernel feature space. 2) SVM in the reduced feature space (*SVM-KPCA, SVM-GDA, SVM-OLap*): perform KPCA, GDA or OLap, and then fit a linear SVM in the reduced space. KPCA and GDA are used with polynomial kernels. Thus, SVM-KPCA and SVM-GDA can be viewed as a reduction of the high-dimensional kernel feature space followed by a linear model fit in the reduced space. 3) Projection penalty as in eq. (17) with KPCA, GDA and OLap (*Proj-KPCA, Proj-GDA, Proj-OLap*). Note that OLap is not a kernel-based reduction technique, so the trick in Section 2.5 is used. The regularization parameter C for SVMs is chosen by stratified cross-validation from the range $[10^{-8}, 10^{-6}, \ldots, 10^{10}]$. For the projection penalty in eq. (17), we set $\gamma = 10^{-5}$.

Results. Experimental results are shown in Tables 5, 6 and 7. The three tables present results for 3, 5 and 7 training images per subject, respectively. The optimal dimension d of the reduced space is reported with each method. In each table, *SVM-Kernel* corresponds to learning a linear SVM in the kernel feature space and serves as the baseline. We observe the following. 1) *SVM-KPCA* vs *Proj-KPCA*: KPCA is not a very effective dimension reduction tool for classification as it does not use the information from labels. *SVM-KPCA* performs similarly or even worse than the baseline model. *Proj-KPCA* performs better than *SVM-KPCA* and the baseline *SVM-Kernel*, showing that the projection penalty is safe and re-

[4] http://cvc.yale.edu/projects/yalefaces/yalefaces.html

liable when used with an ineffective reduction. 2) *SVM-GDA* vs *Proj-GDA*: GDA is significantly more effective than KPCA for dimension reduction. *SVM-GDA*'s performance is superior to the baseline, and *Proj-GDA* further improves the performance. 3) *SVM-OLap* vs *Proj-OLap*: orthogonal Laplacian faces is a powerful reduction technique for supervised learning. All OLap-based methods predict better than KPCA- and GDA-based methods. Performance of *Proj-OLap* is slightly better than *SVM-OLap*: since OLap provides very high-quality dimension reduction for classification, *SVM-OLap*'s performance is approaching the proposed *Proj-OLap*.

4. Related Work

This paper proposes a new approach to use dimension reduction techniques. The main contribution is a regularization framework that utilizes the information from dimension reduction to penalize model parameters. Regularized learning has been the focus of statistics and machine learning for decades. Classical examples include ridge regression (Tikhonov & Arsenin, 1977) and lasso (Tibshirani, 1996) in statistics, and support vector machines (Vapnik, 1995; Burges, 1998) in machine learning. Recently, designing good regularization penalties has been one of the main approaches for multi-task learning (Argyriou et al., 2007) and semi-supervised learning (Belkin et al., 2006). In (Ando & Zhang, 2005), researchers proposed learning "predictive structures" from multiple related tasks. This paper motivates our work in that the predictive structure discussed in the paper is actually a parameter subspace (shared by related tasks).

5. Conclusion

In this paper, we propose the projection penalty framework to make use of dimension reduction techniques and avoid the risk of losing information. Reducing the feature space can be viewed as restricting the model search to some parameter subspace. The idea of a projection penalty is that instead of restricting the search to a parameter subspace, we can still search in the full space but penalize the projection distance to this subspace. We propose projection penalties for linear dimension reduction, then generalize to kernel-based reduction and other nonlinear reduction methods. We empirically study projection penalties with various dimension reduction techniques in regression, text classification, and face recognition. Experimental results show that the projection penalty framework is an effective and reliable way to gain from dimension reduction techniques without losing important information.

Acknowledgments

This work was funded in part by the National Science Foundation under grant NSF-IIS0911032 and the Department of Energy under grant DESC0002607.

References

Ando, Rie Kubota and Zhang, Tong. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

Argyriou, Andreas, Evgeniou, Theodoros, and Pontil, Massimiliano. Multi-task feature learning. In *NIPS 19*. MIT Press, 2007.

Baudat, G. and Anouar, F. Generalized discriminant analysis using a kernel approach. *Neural Comput.*, 12(10):2385–2404, 2000.

Belkin, Mikhail, Niyogi, Partha, and Sindhwani, Vikas. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent Dirichlet Allocation. *JMLR*, 3:993–1022, 2003.

Boyd, Stephen and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

Burges, Christopher J. C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

Cai, Deng, He, Xiaofei, Han, Jiawei, and Zhang, HongJiang. Orthogonal laplacianfaces for face recognition. *TIP*, 15(11):3608–3614, 2006.

Muller, K-R., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B. An Introduction to Kernel-Based Learning Algorithms. *IEEE Trans. Neural Networks*, 12(2):181–201, 2001.

Schölkopf, Bernhard, Smola, Alexander, and Müller, Klaus-Robert. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5): 1299–1319, 1998.

Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Soceity, Series B*, 58(1):267–288, 1996.

Tikhonov, A. N. and Arsenin, V. Y. *Solutions of Ill-posed Problems*. Winston and Sons, 1977.

Vapnik, Vladimir N. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.