Robust Feature Induction for Support Vector Machines

Rong Jin RONGJIN@CSE.MSU.EDU
Department of Computer Science and Engineering, Michigan State University, East Lansing, MI48824

Huan Liu HLIU@ASU.EDU
Department of Computer Science and Engineering, Arizona State University, Tempe, AZ85287-8809

Abstract

The goal of feature induction is to automatically create nonlinear combinations of existing features as additional input features to improve classification accuracy. Typically, nonlinear features are introduced into a support vector machine (SVM) through a nonlinear kernel function. One disadvantage of such an approach is that the feature space induced by a kernel function is usually of high dimension and therefore will substantially increase the chance of over-fitting the training data. Another disadvantage is that nonlinear features are induced implicitly and therefore are difficult for people to understand which induced features are critical to the classification performance. In this paper, we propose a boosting-style algorithm that can explicitly induces important nonlinear features for SVMs. We present empirical studies with discussion to show that this approach is effective in improving classification accuracy for SVMs. The comparison with an SVM model using nonlinear kernels also indicates that this approach is effective and robust, particularly when the number of training data is small.

1. Introduction

For the last couple of years, support vector machines (SVMs) have been successfully applied to many applications, such as text classification (Joachims, 2001), image classification (Teytaud & Sarrut, 2001), and face recognition (Phillips, 1998). Compared to other learning machines, SVMs have two advantages: 1) controlling model complexity through a regularization term, and 2) inducing nonlinear features through a nonlinear kernel function. The first advantage makes SVMs more robust to noises and less likely to over-fit the training data. The second advantage increases the expressive power of SVMs and makes SVMs suitable for complicated classification tasks. Many empirical studies have shown that appropriate choice of nonlinear kernels can significantly improve the classification accuracy (Teytaud & Sarrut, 2001). However, there are also some shortcomings of using kernel-induced features for an SVM model:

1) *Non-sparse weights*. The feature space induced by a nonlinear kernel function is usually of large dimensionality and could be infinite in some cases. When the data can be separated with a few nonlinear features, a kernel-induced feature scheme may have to assign non-zero weights to many training examples in order to concentrate on those important induced features. This can result in sparseness in the induced feature space that can lead to a non-sparse weight distribution of the training examples.

2) *Inappropriate regularization*. In practice, people found that data normalization (i.e., subtracted by means and divided by standard variance) usually improves the performance of an SVM. This is because an SVM uses the ℓ_2 norm of vector $\mathbf{w}$ (i.e. $\|\mathbf{w}\|_2$) as its penalty term, in which weights of different features are added together using the same scale (i.e., 1). Thus, the penalty term in an SVM requires the values of different input features comparable. However, it is rather difficult to normalize kernel-induced nonlinear features because they usually do not have explicit expressions. As a result, high order nonlinear features may suffer more from the penalty term $\|\mathbf{w}\|_2$ since their values are rather small and need large weights to have noticeable impact on decision boundaries.

3) *Implicit features*. It is usually difficult to derive explicit expressions for kernel-induced features. This makes it hard to infer critical nonlinear features from learned SVM models using nonlinear kernel functions. This increases the difficulty for users to comprehend the learned results.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

In this paper, we propose a feature induction algorithm that *explicitly* introduces nonlinear features into an SVM to boost its classification performance. Particularly, this new algorithm addresses the above three shortcomings with the kernel-induced features. The explicit introduction of nonlinear features allows the proposed algorithm to concentrate on the subspace of important features. Data normalization can be performed on the induced features in the same fashion as on the existing features. It is easier to interpret an SVM model using the explicitly induced features than an SVM model with the features implicitly induced by kernel functions.

The proposed algorithm works as follows: for each iteration, the algorithm examines the characteristics of the misclassified training examples and generates a new feature aimed to correct mistakes. The algorithm is similar to AdaBoost (Schapire & Singer, 1999) in that both boost the classification performance by utilizing the misclassified training examples. The most important distinction between them is that the proposed algorithm creates new features while the AdaBoost algorithm generates new instances of 'weak' classifiers. If we view each instance of the 'weak' classifier as a feature generation function, the AdaBoost algorithm may also be treated as a feature induction method. However, the proposed algorithm differs from AdaBoost in the following three important aspects:

1) The proposed algorithm only needs to identify the important nonlinear features while the AdaBoost algorithm has to create new 'weak' classifiers as well as compute the optimal weights that combine multiple 'weak' classifiers.

2) The proposed algorithm can take advantage of the regularization mechanism within an SVM model to avoid the over-fitting problem. In contrast, AdaBoost is a greedy algorithm. Empirical studies have shown that the AdaBoost algorithm tends to over-fit training examples when the data is noisy (Opitz & Macline, 1999; Ratsch et al., 2000; Grove & Schuurmans,1998; Dietterich, 2000; Jin et al., 2003).

3) Unlike AdaBoost where the induced 'weak' classifiers are linearly combined, the new features induced by the proposed algorithm can be combined nonlinearly through kernel functions.

The rest of the paper is arranged as follows: Section 2 discusses the related work; Section 3 describes the details of the proposed feature induction algorithm for SVMs; Section 4 presents an empirical study for the proposed algorithm; and Section 5 concludes this work.

2. Related Work

We will first discuss the previous work on feature induction, next review an SVM model and its related kernel method because this work targets on the feature induction problem for SVMs, and then describe the core idea of the AdaBoost algorithm since the proposed algorithm utilizes the idea of boosting algorithms.

Previous work on feature induction. Della Pietra et al. (1997) proposed an efficient algorithm to search for features that can effectively increase the likelihood of training data (i.e. $\sum_{i \in Training} \log p(y_i \mid x_i)$). This idea was applied to the whole sentence language model (Rosenfeld et al., 1999), in which a sentence is summarized into a set of features and described by a conditional exponential model. It is later extended to the conditional random field (MacCallum, 2003) to allow for more tuning of the algorithm in order to further improve the learning efficiency. Similarly, the proposed feature induction algorithm is to search for features that can effectively decrease the objective function in an SVM. The difficulty of inducing effective nonlinear features for SVMs arises due to the fact that SVMs attempt to solve a constrained optimization problem in which features only appear in the constraints. As a result, changes in features cannot directly influence the objective function of an SVM model. This is in contrast to the feature induction for a conditional exponential model in which the objective function is expressed explicitly in terms of input features. Rennie & Jaakkola's work (2003) also involved feature induction for classifying spam emails. They propose a compression algorithm for text categorization and introduce effective features to achieve the maximum compression rate. It is different from aforementioned works on feature induction in that it does not have an analytic objective function that is related to features. Therefore, the search of new features is rather heuristic and ad-hoc.

Support Vector Machines. Here we only review the SVM model for binary classification (Burges, 1998). Let D denote the training data, $D = \{< \vec{x}_i, y_i >\}_{i=1}^{N}$. Each pair in the training data consists of an input vector $\vec{x}_i$ and output label y_i (either +1 or -1). An SVM model with the linear kernel is written as the following optimization problem:

$$\begin{aligned} &\min_{\vec{w},b,\xi_1,\ldots,\xi_N} \|\vec{w}\|_2 + c\sum_{i=1}^{N}\xi_i \\ &\text{subject to} \\ &\forall i \in [1\ldots N],\ y_i(\vec{x}_i \cdot \vec{w} - b) \geq 1 - \xi_i,\ \xi_i \geq 0 \end{aligned} \tag{1}$$

where c is a constant that controls the amount of admissible errors. The dual form of the above optimization problem is:

$$\max_{\alpha_1,\ldots,\alpha_N}\left(\sum_{i=1}^{N}\alpha_i\right)-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\vec{x}_i\cdot\vec{x}_j\right)$$
$$\text{subject to } \sum_{i=1}^{N}\alpha_i y_i=0,\ \forall i\in[1\ldots N],\ 0\le\alpha_i\le c \tag{2}$$

By replacing the dot product $\vec{x}_i\cdot\vec{x}_j$ in the above equation with a kernel function $K(\vec{x}_i,\vec{x}_j)$, we can implicitly introduce nonlinear features into the SVM model.

As indicated from Equation (1), the input features do not appear in the objective function. Instead, they show up in the constraints $y_i(\vec{x}_i\cdot\vec{w}-b)\ge 1-\xi_i$ and can only influence the objective function through the constraints. This indirectness makes the induction of effective features rather difficult. To directly relate input features to the objective function, we can rewrite the optimization problem in Equation (1) in the following form:

$$\min_{\vec{w},b}\|\vec{w}\|_2+c\sum_{i=1}^{N}\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1) \tag{3}$$

We can observe in the above Equation that constraints are removed by replacing each slack variable ξ_i with a term $\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1)$ in the objective function. Optimizing Equation (3) is still difficult because it involves discontinuous function `max'. To solve this problem, we follow the idea in (Zhang et al., 2003) to approximate the function $\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1)$ with a log-linear function $\frac{1}{\gamma}\log\left\{1+\exp(\gamma\left[-y_i(\vec{x}_i\cdot\vec{w}-b)+1\right])\right\}$. It is proved by Zhang et al. (2003) that this log-linear function forms an upper bound for function $\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1)$ and will uniformly converge to the max function when $\gamma\to+\infty$. The proposed feature induction algorithm is based on this particular approximation form of an SVM model.

AdaBoost. As we mentioned earlier, the proposed feature induction algorithm is similar to AdaBoost in that both are iterative algorithms and create a new entity at each iteration to correct the mistakes made by the current model. The main idea of AdaBoost is to introduce a new instance of a 'weak' classifier at each step to greedily reduce the classification error. Specifically, let $H_0(x)$ be the classification function of current iteration and our goal is to generate a new 'weak' classifier $h(x)$ such that the combination of $H_0(x)$ with $h(x)$, i.e., $H(x)=H_0(x)+\alpha h(x)$, will have a smaller classification error. In AdaBoost, a 'weak' classifier is trained over weighted examples with a weight $\exp(-H_0(x_i)y_i)$ for each example $<x_i,y_i>$. The final classifier $H_f(x)$ is the linear combination of all 'weak' classifiers generated from every iteration, i.e., $H_f(x)=\sum_{t=1}\alpha_t h_t(x)$.

If each 'weak' classifier $h_t(x)$ is viewed as a feature function, the final classifier $H_f(x)$ is simply a linear model for the induced features $\{h_t(x)\}$. In this sense, we can treat AdaBoost as a special feature induction algorithm. However, unlike other feature induction algorithms that are only responsible for generating new features, AdaBoost needs to not only generate new features (i.e., 'weak' classifiers) but also learn the linear model (i.e., combination weights α_t). Thus, the feature induction algorithms have advantages over AdaBoost in that they can rely on the classification model to appropriately combine the induced features with the existing features. Particularly, a feature induction algorithm for SVMs can benefit greatly from the nice properties of SVMs: 1) Unlike AdaBoost that may over-fit noisy data, the feature induction algorithm for SVM can avoid the over-fitting problem because of the regularization mechanism inside the SVM model; 2) Unlike AdaBoost where the induced 'weak' classifiers are linearly combined, the features induced for the SVM model are able to interact with each other nonlinearly through a kernel function.

3. Feature Induction Algorithm for SVMs

As discussed in Section 2, in order to find effective features for an SVM, we use the special form of SVMs in Equation (3) and approximate function $\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1)$ with a log-linear function $\frac{1}{\gamma}\log\left\{1+\exp(\gamma\left[-y_i(\vec{x}_i\cdot\vec{w}-b)+1\right])\right\}$. Putting them together, the objective function in the original SVM model is approximated as follows:

$$\begin{aligned} l_{SVM}(\vec{w},b)&=c\|\vec{w}\|_2+\sum_{i=1}^{N}\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1)\\ &\approx l_{app}(\vec{w},b)=\sum_{i=1}^{N}\frac{1}{\gamma}\log\left\{1+\exp(\gamma\left[-y_i(\vec{x}_i\cdot\vec{w}-b)+1\right])\right\}\end{aligned} \tag{5}$$

Note that the regularization term $c\|\vec{w}\|_2$ is removed from the approximated objective function $l_{app}(\vec{w},b)$. This is because

$$\frac{1}{\gamma}\log\left\{1+\exp(\gamma\left[-y_i(\vec{x}_i\cdot\vec{w}-b)+1\right])\right\}\ge\max(0,-y_i(\vec{x}_i\cdot\vec{w}-b)+1)$$

where the equality is taken only when $\gamma\to+\infty$. Therefore, we can always choose constant $\gamma>0$ to

make the approximated objective function $l_{app}(\vec{w},b)$ equal to the true one $l_{SVM}(\vec{w},b)$. In effect, constant γ plays the similar role as the regularization term. An appropriate choice of γ will lead to less generalization error. This point will be further discussed later.

Let $\vec{x}$ be the current feature vector and $g(\vec{x})$ be the new feature that is introduced to the current SVM model. Let $l_{app}(\vec{w},b)$ be the approximated objective function that only uses the old features $\vec{x}$, and $l'_{app}(\{\vec{w},\alpha\},b)$ be the approximated objective function that uses both old features $\vec{x}$ and the new feature $g(\vec{x})$. Therefore $l'_{app}(\{\vec{w},\alpha\},b)$ can be expressed as follows:

$$l'_{app}(\{\vec{w},\alpha\},b) = \sum_i \frac{1}{\gamma}\log\{1+\exp(\gamma-\gamma y_i(\vec{x}_i\cdot\vec{w}+\alpha g(\vec{x}_i)-b))\} \quad (6)$$

where α stands for the weight of the new feature and b' stands for the new threshold value. In the above expression for the new objective function, we assume both the weights for the old features, i.e., $\vec{w}$, and the threshold value b are almost unchanged with the inclusion of the new feature $g(\vec{x})$. This is a reasonable assumption when the classification error of an SVM using the old features $\vec{x}$ is low and the induced feature will not result in a large change in the SVM model.

To find an effective new feature $g(\vec{x})$, we consider the difference between the two objective functions:

$$\begin{aligned} & l'_{app}(\{\vec{w},\alpha\},b)-l_{app}(\vec{w},b) \\ &= \sum_i \frac{1}{\gamma}\log\left\{\frac{1+\exp(\gamma-\gamma y_i(\vec{x}_i\cdot\vec{w}+\alpha g(\vec{x}_i)-b))}{1+\exp(\gamma-\gamma y_i(\vec{x}_i\cdot\vec{w}-b))}\right\} \\ &= \sum_i \frac{1}{\gamma}\log\left(1+\frac{\exp(-\gamma y_i\alpha g(\vec{x}_i)))-1}{1+\exp(\gamma H(\vec{x}_i))}\right) \end{aligned} \quad (7)$$

where $H(\vec{x}_i)$ is defined as $H(\vec{x}_i)=y_i(\vec{x}_i\cdot\vec{w}-b)-1$. Using the inequality $\log(1+x)\le x$, we have an upper bound for the difference of two objective functions as:

$$l'_{app}(\{\vec{w},\alpha\},b')-l_{app}(\vec{w},b) \le \sum_i \frac{1}{\gamma}\frac{\exp(-\gamma\alpha y_i g(\vec{x}_i)))}{1+\exp(\gamma H(\vec{x}_i))} - \sum_i \frac{1}{\gamma}\frac{1}{\left[1+\exp(\gamma H(\vec{x}_i))\right]} \quad (8)$$

To remove the interaction between the induced feature $g(\vec{x})$ and the weight α, we further assume the outputs of $g(\vec{x})$ fall into the interval [-1,1]. Based on the inequality $e^{\lambda x}\le\left(\frac{1+x}{2}\right)e^{\lambda}+\left(\frac{1-x}{2}\right)e^{-\lambda}$ for $x\in[-1,1]$, the upper bound in Equation (8) can be further simplified as:

$$\begin{aligned} & l'_{app}(\{\vec{w},\alpha\},b')-l_{app}(\vec{w},b) \le \\ & \left[\frac{e^{\alpha\gamma}+e^{-\alpha\gamma}}{2}\right]\sum_i \frac{1}{1+\exp(\gamma H(\vec{x}_i))} \\ & -\sum_i \frac{1}{\gamma}\frac{1}{\left[1+\exp(\gamma H(\vec{x}_i))\right]} \\ & -\left[\frac{e^{\alpha\gamma}-e^{-\alpha\gamma}}{2}\right]\sum_i \frac{y_i g(\vec{x}_i)}{1+\exp(\gamma H(\vec{x}_i))} \end{aligned} \quad (8')$$

The first two terms in the above equation are not related to the induce feature and therefore can be ignored. The third term comprises of a summation over all the training examples with each example contributing $\frac{y_i g(\vec{x}_i)}{1+\exp(\gamma H(\vec{x}_i))}$. Therefore, a good feature function $g(\vec{x})$ that minimizes the quantity in Equation (8) can be obtained by learning a classification model over the training examples that are weighted by the factor $\frac{1}{1+\exp(\gamma H(\vec{x}_i))}$. This factor gives high weights to misclassified examples in which $H(\vec{x}_i)$ are negative, and low weights to correctly classified examples where $H(\vec{x}_i)$ are positive. This is similar to the weight distribution $e^{-H(\vec{x})}$ used in AdaBoost but with two important distinctions:

1) The weights used by the proposed feature induction algorithm is bounded, i.e., $\frac{1}{1+\exp(\gamma H(\vec{x}_i))}<1$, while the weights used by AdaBoost is unbounded. This bounded nature allows the proposed algorithm to avoid over-emphasis on the misclassified examples.

2) In the proposed weights for the training examples, constant γ controls the tradeoff between the training error and the model complexity. With a large value of γ, more weights are assigned to the misclassified examples and therefore training errors can be reduced more efficiently. On the other hand, a small γ will result in less weight assigned to the misclassified examples and less chance to over-fit training data. Therefore, constant γ has the similar impact on the generalization error as the regularization term in an SVM model.

The remaining question is how to determine the appropriate value for constant γ, which has been justified as an important factor in the above discussion. Since the introduction of γ is to approximate the original objective function of an SVM model in Equation (3), the optimal γ can be found by minimizing

the difference between the two objective functions $l_{app}(\vec{w},b)$ and l_{SVM}, i.e.,

$$\gamma^* = \arg\min_{\gamma \in \Re} \left(l_{app}(\vec{w},b) - l_{SVM} \right)^2 \quad (9)$$

Feature Induction for SVM

F: a set of features $\mathbf{F} = \{f^1, f^2, ..., f^D\}$

W: weights for training examples $\mathbf{W} = \{W^1, W^2, ..., W^N\}$

D: training examples $\mathbf{D} = \{< \vec{x}_i, y_i >\}_{i=1}^N$

Initialization:

- **W** =$\{1, 1, \ldots, 1\}$
- **F** = {all the original features}

Repeat until the desired performance

- Train a SVM model using feature set **F**
- Compute **W** using $W^i = 1/\left[1 + \exp(\gamma H(\vec{x}_i))\right]$
- Sample examples according the distribution $W^i / \sum_j W^j$
- Train a classification model $g(\vec{x}) : \mathbf{X} \rightarrow [-1...1]$ using the sampled examples
- $\mathbf{F} \leftarrow \{\mathbf{F}, g(\vec{x})\}$

Figure 1: Description of the feature induction algorithm for SVMs

It can be seen that that function $\left(l_{app}(\vec{w},b) - l_{SVM} \right)^2$ is a convex function with regard to γ, which can be easily justified by its second order derivative. Therefore, there is a global minimum solution for γ. Standard optimization algorithms such as conjugate gradient and the Newton method can be applied to solve the optimization problem in Equation (9).

Figure 1 summarizes the proposed feature induction algorithm for SVMs, which alternatively applies the procedure of computing weights and the procedure of training a new classification model.

4. Experiments

In this experiment, we will examine the effectiveness of the proposed feature induction algorithm for SVM. Particularly, we will address the following three questions:

1) *How effective and robust is the proposed feature induction algorithm for improving classification accuracy of SVMs*? 50 different nonlinear features are generated for each experiment and the change of testing errors using linear SVM model with respect to the number of induced features are examined.

2) *How effective is the proposed feature induction algorithm compared to SVMs using nonlinear kernels*? SVMs with polynomial kernels and a RBF kernel are experimented and their results are compared with those using the feature induction algorithm.

3) *How effective is the proposed feature induction algorithm compared to AdaBoost*? The AdaBoost algorithm using a linear SVM as its basis classifier is run over the same set of datasets and its results are compared to the feature induction algorithm.

Data Set	# Examples	# Features
ionosphere	351	341
breast_cancer	268	9
spam	4601	58
cmc	1473	10
people	3840	9
outdoor	3500	126

Table 1: Statistics of Experimental Datasets

Six different datasets are used in experiments: four of them are from the UCI machine learning repository and the other two are from real world applications. The statistics of all eights datasets are listed in Table 1. 'People' and 'Outdoors' are the two datasets from image classification tasks: dataset 'People' is used for the task of identifying scenes that have more than two people, and dataset 'Outdoor' is used for the task of identifying outdoor scenes. Both are data samples used for TREC video retrieval evaluation (TRECVID, 2003).

In the experiments, we choose the decision tree algorithm (C4.5, Quinlan, 1993) to be the classification model for the proposed feature induction algorithm. This is because the decision tree algorithm can introduce nonlinear interactions between multiple features through conjunctions of Boolean literals. A Newton method is used to find γ that optimizes Equation (9). To further prevent over-fitting the training data, we set the upper bound for γ to be 50 in the experiments. The WEKA SVM implementation (Witten & Frank, 1999) is used for experiments. The kernel functions used in comparison are a RBF kernel and polynomial kernels with degrees ranging from 1 to 5. The maximum number of features induced by the proposed algorithm is 50. Unless specified, all experiments are conducted using the 10-fold cross validation, with 90% of the data used for training and the remaining 10% for testing.

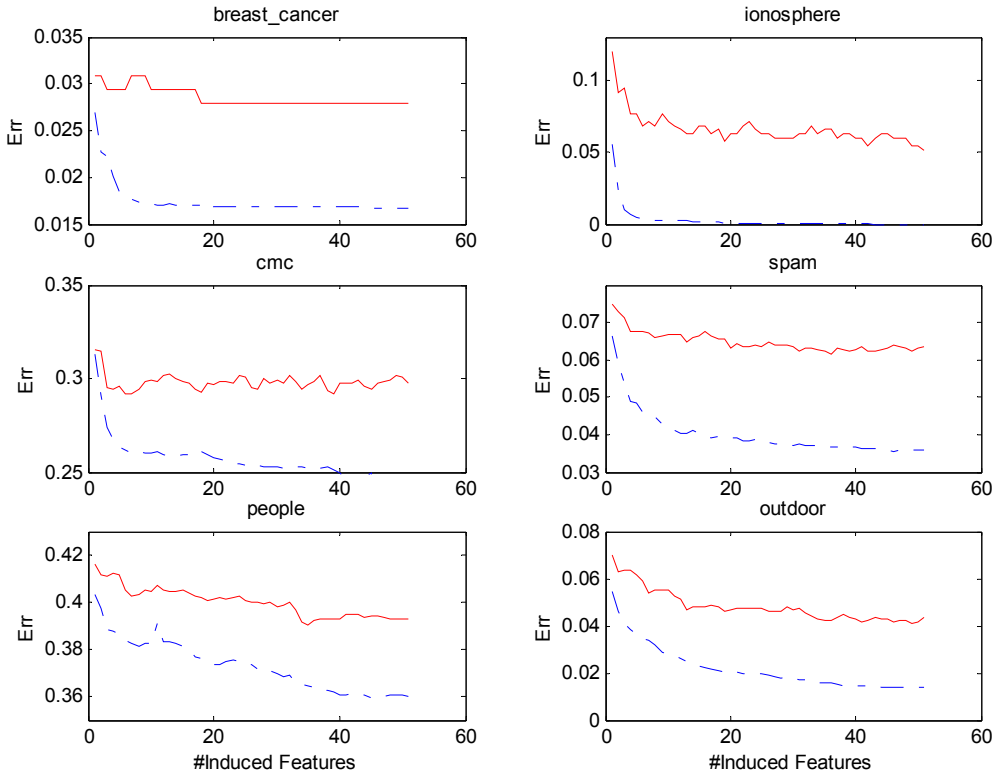

Figure 2: Training and testing errors of a linear SVM model using the proposed feature induction algorithm for six different datasets. Each solid line represents the change of testing errors with respect to the number of induced features and each dash-dot line represents the corresponding change in training errors.

4.1 Experiment I: Effectiveness and Robustness of The Feature Induction Algorithm

Figure 2 displays both the training error and the testing errors of a linear SVM model that uses the proposed feature induction algorithm for six different datasets. Each solid line represents the change of training errors with respect to the number of induced features. Each dash-dot line represents the change of the corresponding testing errors. First, for all datasets, the induced features are able to significantly reduce the training error of a linear SVM model. The most noticeable case is dataset 'ionosphere', in which the training error has been reduced from 5.6% to zero. This fact indicates that the approximation used in the previous analysis is valid and reasonably accurate. Second, despite of the minor fluctuation in testing errors, for all datasets, the overall trend is that the more the induced features are, the smaller the classification error is. Notice that although datasets such as 'people', 'cmc', and 'breast_cancer' have no more than 10 features, by introducing 50 new nonlinear features, we are still able to reduce the testing error of a linear SVM model without significantly overfitting training data. The most noticeable case is dataset 'breast_cancer', in which a significant drop in the testing error is observed when the number of induced features is between 7 and 9. However, with more induced features, we see that the testing error for dataset 'breast_cancer' keeps decreasing. This trend is consistent with the analysis presented in the introduction section. Explicitly induced new nonlinear features can be normalized to the same scale as the original features. As a result, the

Poly. Degree	breast_cancer	ionosphere	cmc	spam	people	outdoor
1	3.97%	9.71%	33.40%	6.67%	41.98%	7.13%
2	2.06%	10.29%	30.48%	18.07%	45.35%	1.12%
3	2.79%	16.57%	30.75%	28.61%	45.54%	1.12%
4	2.65%	18.57%	30.54%	36.43%	45.51%	1.15%
5	3.82%	18.29%	33.13%	37.17%	45.51%	1.17%

Table 2: Testing errors for SVM model using polynomial kernels with degree ranging from 1 to 5.

regularization mechanism in the SVM model is effective in preventing the overfitting of training data that could potentially be caused by the large number of induced features. In contrast, applying nonlinear kernels to induce nonlinear features can actually lead to a significant overfitting problem. Table 2 presents the testing errors of the SVM model using polynomial kernels of different degrees. Notice that, for many datasets, a high degree polynomial kernel can significantly degrade the SVM performance. For example, compared to a linear SVM model (poly. degree equals 1), using a polynomial kernel of degree 5 results in a substantially worse classification error for almost all datasets except 'outdoor' as shown in Table 2. The most noticeable case is dataset 'spam', whose testing error is only 6.67% for a linear SVM model. However, the testing error shoots up to 37.17% when a polynomial kernel of degree 5 is used. As discussed in the introduction section, this phenomenon can be attributed to the fact that nonlinear features are introduced implicitly through kernel functions and therefore are difficult to be normalized to the same scale as the original features. As a result, the regularization mechanism in SVM may not be able to work appropriately to prevent overfitting problems.

The trend seen in Figure 2 shows that the increased number of induced features can often help improve testing accuracy. However, the increased number of induced features is associated with increased computing costs. Its diminishing gain in accuracy as the number of induced features increases suggests a heuristic stopping criterion: stop when the gain between the two subsequent runs is insignificant.

4.2 Experiment II: Comparison with Nonlinear Kernels

We apply both polynomial kernels and the RBF kernel to introduce nonlinear features for SVMs. Both the degree of polynomial kernels and variance for RBF kernel are determined through an internal cross validation procedure using 20% of training data that are randomly selected from the original training pool. The same cross validation procedure is applied to determine the variance for the RBF kernel. The results of testing errors for polynomial kernels and RBF kernels are presented in Table 3, together with the results for the proposed feature induction algorithm. For the purpose of comparison, we also include the testing errors for an SVM model using the linear kernel, which is referred as baseline model in Table 3. A boldfaced number indicates that its testing error is substantially better than the baseline model, and an italic font is applied when it is substantially worse than the baseline model. First, between the two kernel functions, the RBF kernel appears to perform significantly better than the polynomial kernels for almost all datasets. It manages to improve the performance substantially for three out of six datasets and achieves the same performance as the baseline model for the rest datasets. In contrast, the polynomial kernels improve classification accuracy for only two out of six datasets while degrading the performance for the other three datasets. Second, the proposed feature induction algorithm consistently performs better than the baseline model for all datasets. Particularly, it achieves substantial reduction in testing errors for five out of six datasets. This is noticeably better than both the RBF and polynomial kernels. Based on the above observation, we conclude that the proposed algorithm is more robust and effective than the nonlinear kernels in terms of inducing nonlinear features for the SVM model.

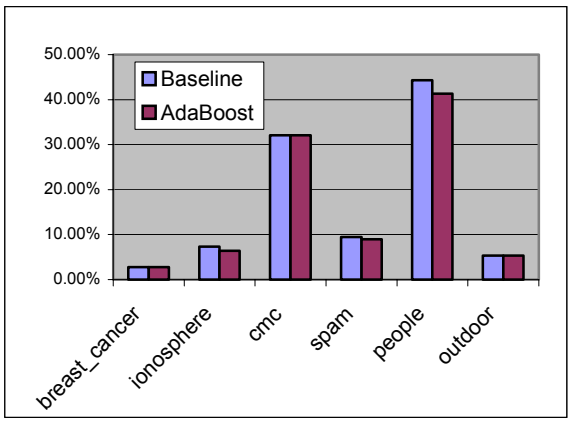

Figure 3: Training errors of AbaBoost and a linear SVM model (referred as baseline).

4.3 Experiment III: Comparison with AdaBoost

Methods	breast_cancer	ionosphere	cmc	spam	people	outdoor
baseline	3.97%	9.71%	33.40%	6.67%	41.98%	7.13%
polynomial	**2.69%**	*13.43%*	34.90%	*9.50%*	*45.40%*	**1.03%**
RBF	**2.79%**	8.97%	**30.30%**	6.81%	41.4%	**0.50%**
Feature Induction	**2.75%**	**5.14%**	**29.80%**	6.35%	**39.32%**	**4.33%**
AdaBoost	3.20%	*12.81%*	33.00%	*9.37%*	41.40%	6.51%

Table 3: Testing errors for different methods. The baseline model refers to a linear SVM model. A bold font is applied when a testing error is substantially better than the baseline model, and an italic font is applied when it is substantially worse than the baseline model

We compare the proposed feature induction algorithm for SVMs to the AdaBoost algorithm that uses SVM model as its base classifier. For both algorithms, a linear kernel is used for the SVM model. The maximum number of iterations for AdaBoost is set to be 20. The testing errors for AdaBoost are presented in Table 3. Compared to the proposed feature induction algorithm, AdaBoost appears to be less effective in improving the performance of an SVM model. For most datasets, it only achieves the same performance as the original SVM model. A further examination of training errors for AdaBoost in Figure 3 indicates that the AdaBoost algorithm cannot even reduce the training errors significantly for almost all datasets except 'people'. This can be explained by the fact that both a linear SVM model and the AdaBoost algorithm using a linear combination model. As a result, the introduction of AdaBoost does not change the linear nature of a linear SVM model and therefore has little impact on the classification accuracy.

5. Conclusions

Induced features can reduce classification errors. In this paper, we propose a new feature induction algorithm for SVMs. Unlike the kernel method where the number of induced features is usually large (even infinite), the proposed algorithm only creates nonlinear features that can effectively reduce training errors. Specifically, new features are induced iteratively. At each step, it weights training examples differently according to the outputs of the current SVM model. The weighted training examples are then used to train a classification model that becomes a new feature for the SVM model. Empirical studies over both UCI datasets and datasets of real applications indicate that the proposed feature induction algorithm is not only effective in improving the performance of a linear SVM model but also robust even when the number of induced features is substantially larger than the number of original features.

References

Joachims, T. (2001). A Statistical Learning Model of Text Classification for SVMs. In Proceeding of 24th ACM International Conference on Research and Development in Information Retrieval.

Teytaud, O. and Sarrut, O. (2001). Kernel-based Image Classification, Lecture Notes in Computer Science, http://www.citeseer.nj.nec.com/496638.html.

Phillips, P. J. (1998). Support Vector Machines Applied to Face Recognition. In Advances in Neural Information Processing Systems 11, page 803.

Opitz, D., & Macline, R. (1999). Popular Ensemble methods: An Empirical Study. Journal of AI Research pp.169–198.

Jin, R., Liu, Y., Si, L., James, C., & A.G. Hauptmann (2003). A New Boosting Algorithm Using Input-Dependent Regularizer. In Proceedings of 20th International Conference on Machine Learning.

Dietterich, T. G. (2000). An Experimental Comparison of Three Methods for constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Machine Learning, 40, 139–157.

Grove, A. J., & Schuurmans, D. (1998). Boosting in the Limit: Maximizing the Margin of Learned Ensembles. In Proceedings of the Fifteenth National Conference on Artificial Intelligence pp. 692–699.

Ratsch, G., Onoda, T., & Muller, K. (2000). Soft Margins for Adaboost. Machine Learning, 42, 287–320.

Rennie, J D. M. & Jaakkola, T. (2002). Automatic Feature Induction for Text Classification, MIT Artificial Intelligence Laboratory Abstract Book.

Della Pietra, S., Della Pietra, V.J., & Lafferty, J.D. (1997). Inducing Features of Random Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, 380-393.

Rosenfeld, R., Wasserman, L., Cai, C., & Zhu, X.J. (1999). Interactive Feature Induction and Logistic Regression for Whole Sentence Exponential Language Models. In Proc. IEEE workshop on Automatic Speech Recognition and Understanding, Keystone, Colorado.

Burges, C.J.C. (1998). A Tutorial on Support Vector Machine for Pattern Recognition, Knowledge Discovery and Data Mining, 2(2).

Witten, I.H. & Frank, E. (1999). Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann.

Schapire, R.E. & Singer, Y. (1999). Improved Boosting Algorithms using Confidence-rated Predictions, *Machine Learning* 37 (3): 291-336.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA.

TRECVID (2003). http://www-nlpir.nist.gov/projects/tv2003/tv2003.html.

Zhang, J., Jin, R., Yang, Y. & Hauptmann, A.G. (2003). Modified Logistic Regression: An Approximation to SVM and its Applications in Large-Scale Text Categorization, In Proc. of International Conference on Machine Learning (ICML 2003).

McCallum, A. (2003). Efficiently Inducing Features of Conditional Random Fields. In Proc. Of 19th Uncertainty in Articifical Intelligence (UAI 2003).