Nightmare at Test Time: Robust Learning by Feature Deletion

Amir Globerson GAMIR@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA

Sam Roweis ROWEIS@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Canada

Abstract

When constructing a classifier from labeled data, it is important not to assign too much weight to any single input feature, in order to increase the robustness of the classifier. This is particularly important in domains with nonstationary feature distributions or with input sensor failures. A common approach to achieving such robustness is to introduce regularization which *spreads* the weight more evenly between the features. However, this strategy is very generic, and cannot induce robustness specifically tailored to the classification task at hand. In this work, we introduce a new algorithm for avoiding single feature over-weighting by analyzing robustness using a game theoretic formalization. We develop classifiers which are optimally resilient to *deletion* of features in a minimax sense, and show how to construct such classifiers using quadratic programming. We illustrate the applicability of our methods on spam filtering and handwritten digit recognition tasks, where feature deletion is indeed a realistic noise model.

1. Building Robust Classifiers

When constructing classifiers over high dimensional spaces such as texts or images, one is inherently faced with the problem of under-sampling of the true data distribution. Even so-called "discriminative" methods which focus on minimizing classification error (or a bound on it) are exposed to this difficulty since the training objective will be calculated over the observed input vectors only, and thus may not be a good approximation of the average objective on the test data. This is especially important in settings such as document classification where features may take on certain observed values (e.g. a zero count for a particular vocabulary item) due to small sample effects. Furthermore, it may not be the case that the test data and training data come from the same distributions. The distribution of words in spam email, for example, changes very rapidly and keywords which are highly predictive of class in the training set (e.g. "hurricane") may not be indicative or even present in the test data. As another example, consider a digital camera whose output is fed to a face recognition system. Due to hardware or transmission failures, a few pixels may "die" over the course of time. In the image processing literature, this is referred to as *pepper* noise (Bovik et al., 2000) (*salt* noise refers to the case when pixels values are clipped to some fixed value). Any classifier which attached too much weight to any single pixel would suffer a substantial performance loss in this case. As a final example, consider a network of local processing elements in an artificial sensor network or a biological network such as the cortex. The hardware/wetware of such systems is known to be extremely unreliable (thousands of neurons die each day) and yet the overall architecture maintains its function, indicating a remarkable robustness to such non-stationarities in its input.

Given this uncertainty about the true underlying nature of the data, it is crucial not too attach too much weight to any single data dimension during testing, however informative it may seem at training time. All the above examples describe a scenario where features that were present when constructing the classifier (i.e., in the training data), are potentially *deleted* at some future point in time. Such deletion may manifest itself differently depending on the particular domain: a deleted feature may be known to be unavailable or unmeasured; it may take on random values; or its value may set to some constant. In our formal treatment, we focus on the case where deletion corresponds to setting the feature's value to zero. Indeed, in the examples given above this is an appropriate description.

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

Of course, when constructing the classifier, we cannot anticipate in advance which features may be deleted in the future. One possible strategy is to analyze the performance under random deletion of features. However, this may not be a correct model of the deletion statistics. The approach we take here is to construct a classifier which is optimal in the worst case deletion scenario, thus avoiding any modeling assumptions about the deletion mechanism. This can be formulated as a two player game, where the action of one player (the classifier builder) is to choose robust classifier parameters, whereas the other player (the feature removal mechanism) tries to delete the feature which would be most harmful given the current classifier. Robust minimax approaches to learning classifiers have recently attracted interest in the machine learning community (Lanckriet et al., 2004; El Ghaoui et al., 2003; Kim et al., 2006). Our approach is related to (El Ghaoui et al., 2003) where the location of sample points is only known up to an ellipsoidal region, and a classifier that is optimal in the worst case is sought. However, in our case, the structure of uncertainty is inherently different and is related to the existence vs. non-existence of a feature.

In the next section we formalize this minimax game for classifiers such as the support vector machine (Schölkopf & Smola, 2002) in which the training objective is measured using a regularized hinge loss. We show that the game results in a tractable quadratic program for training robust classifiers, which we denote by the name FDROP . We then go on to illustrate the method's performance on several datasets.

2. Minimax Problem Formulation

Given a labeled sample $(\mathbf{x}_i, y_i)$ $(i = 1, \ldots, n)$, with input feature vectors $\mathbf{x}_i \in \Re^d$ and class labels[1] $y_i \in \{\pm 1\}$, we would like to construct classifiers which are robust to *deletion* of features. We focus on the case where a feature is assigned the value of zero if it is deleted, and denote by K the maximum number of features which may be deleted for any given point $\mathbf{x}$. (This limitation is necessary since if for example all features may be deleted, then in the worst case there is no access to the true data.)

We assume that the objective function we would like to maximize when training the classifier is measured using the hinge loss[2] $\sum_i [1 - y_i \mathbf{w} \cdot \mathbf{x}_i]_+$ (Recall that the hinge loss is also a convex upper bound on the zero one loss: $l_{zo}(\mathbf{w}, y, \mathbf{x}) \leq \sum_i [1 - y_i \mathbf{w} \cdot \mathbf{x}_i]_+$, so that by minimizing hinge loss we are actually minimizing a bound on the training error). We do not introduce a bias term explicitly, but including a bias is straightforward by adding a constant feature $x_{d+1} = 1$.

We would like our classifier to be robust to test time deletion of features from the input vectors, i.e. robust against the setting of arbitrary feature values to zero. Thus, we seek a classifier which minimizes the worst case hinge loss when up to K features may be deleted from each data vector. In this setting, the worst case hinge loss for example i is given by

$$\begin{array}{rll} h^{wc}(\mathbf{w}, y_i \mathbf{x}_i) = & \max & [1 - y_i \mathbf{w} \cdot (\mathbf{x}_i \circ (1 - \boldsymbol{\alpha}_i))]_+ \\ & s.t. & \boldsymbol{\alpha}_i \in \{0, 1\} \\ & & \sum_j \alpha_{ij} = K \end{array} \quad (1)$$

where maximization is over all legal assignments to $\boldsymbol{\alpha}_i$, and α_{ij} denotes the j^{th} element of $\boldsymbol{\alpha}_i$ which equals 1 if the j^{th} feature of $\mathbf{x}_i$ is deleted (we use $\circ$ to denote the element-wise multiplication operation).

The worst case hinge loss over the entire training set is $\sum_i h^{wc}(\mathbf{w}, y_i \mathbf{x}_i)$. In analogy to the standard support vector machine formulation, we will minimize some tradeoff between this worst-case error and a regularization term involving the weight vector norm [3]

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i h^{wc}(\mathbf{w}, y_i \mathbf{x}_i) \quad (2)$$

To solve the above problem we first analyze the worst case hinge loss $h^{wc}(\mathbf{w}, y_i \mathbf{x}_i)$. This loss can be seen to be minimized when $\boldsymbol{\alpha}_i$ is chosen to delete the K features x_{ij} with highest values $y_i w_j x_{ij}$, since these will have the strongest decreasing effect on the loss. Thus we can rewrite $h^{wc}(\mathbf{w}, y_i \mathbf{x}_i)$ as

$$h^{wc}(\mathbf{w}, y_i \mathbf{x}_i) \quad = \quad \left[1 - y_i \mathbf{w}^T \mathbf{x}_i + s_i\right]_+ \ ,$$

where we have defined

$$s_i = \max_{\boldsymbol{\alpha}_i \in \{0,1\}, \sum \alpha_{ij} = K} y_i \mathbf{w} \cdot (\mathbf{x}_i \circ \boldsymbol{\alpha}_i) \quad (3)$$

as the maximum contribution of K features to the margin of sample $\mathbf{x}_i$.

To simplify the expression for s_i, we note that the integer constraint on the variables $\boldsymbol{\alpha}_i$ may be relaxed to $0 \leq \boldsymbol{\alpha}_i \leq 1$ without changing the optimum. This is true since the vertices of the resulting $2d + 1$ linear inequalities are integral. Since the maximization is over

[1]We focus on the binary case here. All results can be easily generalized to the multi class case.

[2]It would be interesting to consider other loss functions, such as the log loss

[3]Note that when bias is added using a feature $x_{d+1} = 1$, its corresponding weight is not included in the regularization. All results are easily extendable to this case.

a linear function, the optimum will be at the vertices, and is therefore integral. We rewrite s_i using this relaxation, and also changing the order of multiplication

$$\begin{array}{rll} s_i = & \max & y_i\,(\mathbf{w} \circ \mathbf{x}_i) \cdot \boldsymbol{\alpha}_i \\ & s.t. & 0 \leq \boldsymbol{\alpha}_i \leq 1 \\ & & \sum_j \alpha_{ij} = K \end{array} \tag{4}$$

The above expression for h^{wc} is bilinear in $\boldsymbol{\alpha}_i, \mathbf{w}$. Since this may potentially contribute a non-convex factor into the optimization, we use a duality transformation to avoid bilinearity. The dual problem of the s_i maximization optimizes over the variables $\mathbf{v}_i \in \Re^d, z_i \in \Re$, yielding

$$\begin{array}{rll} s_i = & \min & K z_i + \sum_j v_{ij} \\ & s.t. & z_i + \mathbf{v}_i \geq (y_i \mathbf{x}_i \circ \mathbf{w}) \\ & & \mathbf{v}_i \geq 0 \end{array} \tag{5}$$

To plug this into the minimization problem, we introduce a variable t_i which upper bounds the minimum. The resulting problem is

$$\begin{array}{ll} \underline{\text{FDROP:}} & \\ \min & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \left[1 - y_i \mathbf{w}^T \mathbf{x}_i + t_i\right]_+ \\ \text{s.t.} & t_i \geq K z_i + \sum_j v_{ij} \\ & \mathbf{v}_i \geq 0 \\ & z_i + \mathbf{v}_i \geq (y_i \mathbf{x}_i \circ \mathbf{w}) \end{array} \tag{6}$$

The above problem is quadratic and can be solved efficiently. Note that the hinge function $[x]_+$ can be written using linear constraints (as is commonly done in SVM formulations).

2.1. The Dual Robust Problem

The classic support vector machine problem is a convex quadratic problem, and has a dual convex which reveals some interesting properties and allows the use of kernel classifiers. Since our robust problem is also quadratic and convex, it is interesting to consider its dual problem. It can be shown (see Appendix A) that the dual of our robust classifier construction problem is the following optimization [4]

$$\begin{array}{ll} \min & \frac{1}{2}\|\sum_i y_i \alpha_i \mathbf{x}_i \circ (\mathbf{1} - \boldsymbol{\lambda}_\mathbf{i})\|^\mathbf{2} - \sum_\mathbf{i} \alpha_\mathbf{i} \\ \text{s.t.} & 0 \leq \boldsymbol{\alpha} \leq C \\ & 0 \leq \boldsymbol{\lambda}_i \leq 1 \\ & \sum_j \lambda_{ij} = K \end{array} \tag{7}$$

where the are variables are: $\boldsymbol{\alpha}$, a vector of dimension n (the number of samples), and $\boldsymbol{\lambda}_i$ a set of n vectors, each of dimension d (the dimension of the input). Furthermore, the optimal set of weights $\mathbf{w}$ can be expressed as:

$$\mathbf{w} = \sum_i y_i \alpha_i \mathbf{x}_i \circ (\mathbf{1} - \boldsymbol{\lambda}_\mathbf{i}) \tag{8}$$

The above problem can be written in an alternative form, where it is more clearly convex

$$\begin{array}{ll} \min & \frac{1}{2}\|\sum_i y_i \mathbf{x}_i \circ (\alpha_i \mathbf{1} - \boldsymbol{\lambda}_\mathbf{i})\|^\mathbf{2} - \sum_\mathbf{i} \alpha_\mathbf{i} \\ \text{s.t.} & 0 \leq \boldsymbol{\alpha} \leq C \\ & 0 \leq \boldsymbol{\lambda}_i \leq \alpha_i \\ & \sum_j \lambda_{ij} = K\alpha_i \end{array} \tag{9}$$

Here the expression in the norm is an affine function of the variables, and thus the problem is convex.

Recall that the SVM dual is

$$\begin{array}{ll} \min & \frac{1}{2}\|\sum_i y_i \mathbf{x}_i \alpha_i\|^2 - \sum_i \alpha_i \\ \text{s.t.} & 0 \leq \boldsymbol{\alpha} \leq C \end{array} \tag{10}$$

where $\mathbf{w} = \sum_i y_i \mathbf{x}_i \alpha_i$.

Thus, in our case the weight vector is not a combination of input vectors, but rather a combination of vectors weighted by elements of weight *up to* α_i where the maximal number of elements that may be set to zero is K. Interestingly, the $\boldsymbol{\lambda}_i$ values can be fractional, so that none of the features has to be completely deleted.

Note that, as opposed to the classic SVM, our robust objective does not involve dot products between $\mathbf{x}_i$, but also $\boldsymbol{\lambda}_i$, and the dimension of the variable $\boldsymbol{\lambda}_i$ is still d (the dimension of $\mathbf{x}_i$). Thus it is not immediately clear if and how kernel methods may be put to use in this case. This is not surprising, since the algorithm is strongly linked to the structure of the sample space $\Re^d$, where features are dropped. Dropping such features alters the kernel function, and it remains an interesting challenge to obtain algorithms for this case.

2.2. An Alternate Setting: Uniform Feature Deletion

In the previous section, we assumed that different features may be deleted for different data points. We can also consider an alternative formulation where once a feature is chosen to be deleted it is deleted uniformly from all data points simultaneously. Clearly, this scenario is subsumed by the one described in the previous section, and is thus less pessimistic.

The worst case hinge is defined as in the non-uniform case in Equation 1. However, now there is a single $\boldsymbol{\alpha}$ vector for all examples, whereas in the previous scenario, each sample had its own vector. The optimiza-

[4] We denote by $\mathbf{1}$ the vector with all elements one.

tion thus becomes

$$\mathbf{w}^* = \min_{\mathbf{w}} \max_{\substack{\boldsymbol{\alpha} \in \{0,1\} \\ \sum_j \alpha_j = d-K}} \|\mathbf{w}\|^2 + C \sum_i \left[1 - y_i \mathbf{w} \cdot (\mathbf{x}_i \circ \boldsymbol{\alpha})\right]_+$$

We first note that the above optimization problem is still convex. To see this, denote by $f(\mathbf{w})$ the maximum value over all legal $\boldsymbol{\alpha}$ assignments. Then $f(\mathbf{w})$ is a pointwise maximum over a set of convex functions and is thus convex (Boyd & Vandenberghe, 2004) . The problem of minimizing over $\mathbf{w}$ is therefore convex.

However, although it is convex, the current optimization problem appears more difficult than the one in the previous section, due to the presence of the $\boldsymbol{\alpha}$ in all the sum elements. As before, the integral constraints on $\boldsymbol{\alpha}$ can be relaxed, since the maximum of the inner optimization is attained at the vertices (because the target is convex). However, since the target is non-linear (a hinge function) this maximization is not itself a convex problem, and does not seem to be efficiently solvable.

The problem is easily tractable when $\binom{d}{K}$ is sufficiently small so that all the feasible values of $\boldsymbol{\alpha}$ can be enumerated over. Our experiments show that in many cases K needs to be $O(10)$, so that the uniform method is often not applicable.

2.3. Computational Considerations

The FDROP optimization problem in Equation 6 is a quadratic program and can thus be solved in polynomial time. The number of variables in both the primal and dual problems is $O(nd)$, where most of these are the $\mathbf{v}_i$ (or dual $\boldsymbol{\lambda}_i$) variables. This should be contrasted with the $n + d$ variables in the standard SVM problem. Our algorithm is thus more computationally demanding than SVMs. However, it should be noted that the constraints are sparse, such that no more than $O(d)$ variables participate in each constraint. This makes the problem feasible for $n, d = O(1000)$. In the experiments described below we used the ILOG-CPLEX optimization software for solving the quadratic program.

3. Relation to Feature Selection

The adversary in the FDROP minimax problem identifies those input features whose contribution to the margin is maximal. In this way, the adversary can be thought of as being related to feature selection algorithms which try to find the set of features which, when taken alone, would yield optimal generalization. A clear illustration of this effect can be seen in Figure 2 (section 4.2).

However, the current minimax setup differs from the standard feature selection approach in two important aspects. The first is that here we focus on feature *elimination*, i.e., finding the set of features whose elimination would maximally decrease performance. Intuitively, these features should also convey high information when taken on their own, but this is not guaranteed to be the case.

The other aspect which distinguishes the current approach from feature selection is that here features are selected (or eliminated to be precise), for every sample individually. The uniform feature deletion approach described in Section 2.2 is more in line with the standard feature selection one.

We can provide a slightly more formal treatment of feature selection optimization algorithms which highlights their relation to the current approach. The standard feature selection goal is to find a set of K features which minimize generalization error. A reasonable approximation is the empirical error, or the hinge loss in our case. Thus the feature selection problem can be posed as (we omit the regularization term here)

$$\begin{array}{ll} \min & \sum_i \left[1 - y_i \mathbf{w} \cdot (\mathbf{x}_i \circ \boldsymbol{\alpha})\right]_+ \\ s.t. & \boldsymbol{\alpha} \in \{0,1\} \\ & \sum_j \alpha_j = K \end{array} \tag{11}$$

such that minimization is over both $\boldsymbol{\alpha}$ and $\mathbf{w}$. Denote by $f(\mathbf{w})$ the minimum over $\boldsymbol{\alpha}$ assignments for a given value of $\mathbf{w}$. Then $f(\mathbf{w})$ is a pointwise minimum of convex functions and is thus generally non-convex. Thus the optimization problem is not convex, and is generally hard to solve . Furthermore for a large number of features, calculating $f(\mathbf{w})$ requires enumeration over possible $\boldsymbol{\alpha}$ assignments. The problem may be approximated via different relaxations as in (Gilad-Bachrach et al., 2004; Weston et al., 2000).

The above problem may be slightly altered to resemble our current formulation by allowing the best K features to be chosen on a *per sample* basis. (a single set of features might then be selected, for example, by taking the features chosen most often across samples). The resulting optimization problem is

$$\begin{array}{ll} \min & \sum_i \left[1 - y_i \mathbf{w} \cdot (\mathbf{x}_i \circ \boldsymbol{\alpha}_i)\right]_+ \\ s.t. & \boldsymbol{\alpha}_i \in \{0,1\} \\ & \sum_j \alpha_{ij} = K \end{array} \tag{12}$$

This problem is easier than that in Equation 11 in that the minimization over $\boldsymbol{\alpha}_i$ is always tractable: the minimizing $\boldsymbol{\alpha}_i$ is the one which has the minimum contribution to the margin. However, the function $f(\mathbf{w})$ is again non-convex, and thus it seems that the problem remains hard.

It is interesting that these two feature selection variants, while similar in spirit to our minimax problems, seem to belong to a different computational class. This suggests that our algorithm may also prove useful for feature selection by finding the set of features it tends to *delete*. Initial evaluations have shown that its performance is similar to that of the InfoGain method (Yang & Pedersen, 1997).

4. Experiments

We now illustrate the application of our algorithm to synthetic and real data. We shall especially be interested in evaluating performance when features are deleted from the test set. Thus, for example we test handwritten digit recognition when pixels are removed from the image, or spam filtering when words are removed from the text. We also focus on relatively small training sets, such that the inherent sparseness of the problem is high, and most classification algorithms are likely to overfit. We compare our method with a linear support vector machine algorithm. In all experiments, both FDROP and SVM used a bias term. The FDROP algorithm was not *allowed* to delete the bias feature $x_{d+1} = 1$.

4.1. A Synthetic Example

To illustrate the advantages of the current method, we apply it to a setting where the test data indeed differs from the training data by deleting features. We consider a feature vector in $\mathbf{x} \in \Re^{20}$ where training examples are drawn uniformly in that space. The label is assigned according to a logistic regression rule

$$p(y = 1|\mathbf{x}) \propto e^{\mathbf{w}\cdot\mathbf{x}+b} \ . \qquad (13)$$

In our experiments, $w_1 = 5$ and all the other $w_i = -2$. The bias b was set to the mean of $\mathbf{w}$. Thus the feature x_1 is likely to be assigned a high weight by any sensible learning algorithm. In the test data, we delete the feature x_1, i.e. set it to zero, with a given probability $p(delete)$. We compare the performance of our FDROP minimax algorithm (with $K = 1$) to that of a standard SVM. For both methods, we choose the weight of the regularization parameter C via cross validation.

Figure 1 shows the resulting error rates. It can be seen that as the probability of deletion increases, the performance of SVM decreases, while that of the minimax algorithm stays roughly constant. This constant behavior is due to the fact that the FDROP classifier is optimized for the worst case when this feature is deleted. To understand this behavior further, we checked which feature was deleted by FDROP for every one of the samples. Indeed, on *all* the cases where $x_1 = 1$ and $y = 1$, it was x_1 that was deleted in the optimization.

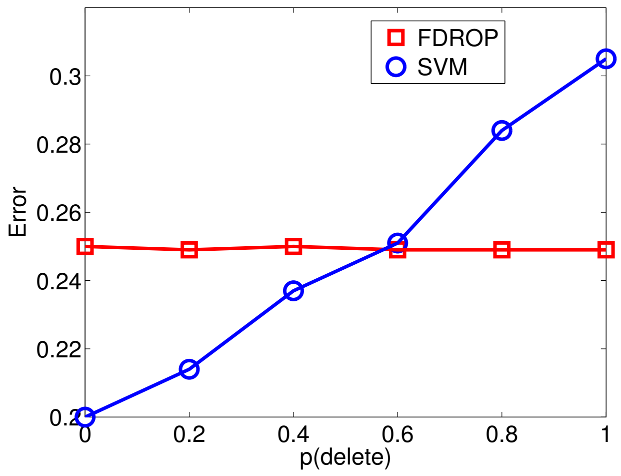

Figure 1. Evaluation of FDROP and SVM on a toy logistic regression example, where a highly informative feature is randomly dropped from the test sample. The value of K was set to 1. The figure shows classification error as a function of the deletion probability $p(delete)$.

4.2. Handwritten Digit Classification

Image classification into categories should in principle be robust to pixel deletion, or in other words deletion of parts of the image. Our game theoretic framework captures this intuition by modeling the worst case pixel deletion scenario.

We investigated the application of FDROP to classifying handwritten digits, and focused on robustness to pixel deletion in these images. We applied FDROP to the MNIST dataset (LeCun et al., 1995) of handwritten digits, and focused on binary problems with small training sets of 50 samples per digit. Furthermore, we only considered binary problems created by label pairs which had more than 5% error when learned using an SVM (The chosen pairs were $(4,9)$,$(3,5)$,$(7,9)$,$(5,8)$,$(3,8)$,$(2,8)$,$(2,3)$,$(8,9)$,$(5,6)$, $(2,7)$,$(4,7)$ and $(2,6)$). The size in pixels of each digit was (28×28). A holdout sample of size 200 was used to optimize the algorithm parameters, and a set of 300 samples was used for testing. The holdout set underwent the same pixel deletion as the test set, in order to achieve a fair comparison between SVM and FDROP. Experiments were repeated with 20 random subsets of the above sizes.

To evaluate the robustness of the algorithm to feature deletion, we trained it on the raw data (i.e., without deleted features), and then tested it on data from which K features were deleted. The values of K were $(0, 25, 50, 75, 100, 125, 150)$.

Figure 2 gives a visual representation of the feature deletion process. The FDROP minimax optimization deletes K features from every sample point. We can find which features were deleted from each sample by finding the K features with maximum margin contribution at the optimal $\mathbf{w}$. Figure 2 illustrates these features for three sample points. Each row displays the original raw input image and the same input image with the K most destructive features deleted (here $K = 50$). It can be seen that FDROP chooses the features which maximize the resemblance between the given digit and the digit in the other class. These results suggest that FDROP may indeed be useful as a feature selection mechanism.

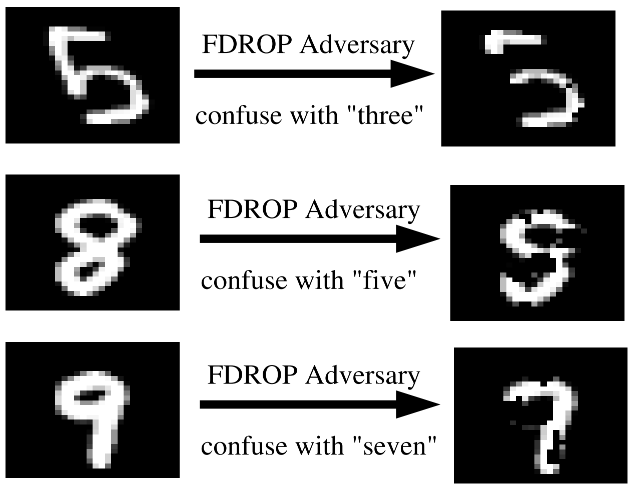

Figure 2. Illustration of adversarial feature (pixel) deletion for handwritten digits. Three binary classification problems were created from the MNIST digit database by discriminating the classes “five” vs. “three” (top), “eight” vs. “five” (middle) and “seven” vs. “nine”. The training data consisted of 50 samples per class. The number of deleted features was $K = 50$. The images show three corresponding examples of features deleted by the FDROP adversary. The left column shows the original digit, and the right column shows the digit with the 50 pixels dropped by the FDROP algorithm. It can be seen that the worst case against which our algorithm attempts to be robust corresponds to the deletion of extremely discriminative features for each example: the top right digit has been made to look as much as possible like a “three”, the middle right digit very much like a “five” and the bottom right digit has been distorted to look very much like a “seven”.

Classification error rate should intuitively decrease as more features are deleted. The goal of FDROP is to minimize the damage incurred by such deletion. Figure 3 shows the dependence of classification error on the number of deleted features for both FDROP and SVM. The parameter K is taken as an unknown and is chosen to minimize error on the holdout sample for each digit pair and deletion level separately. It can be seen that FDROP suffers less degradation in error when compared to SVM. Furthermore, the optimal K grows monotonously with the number of deleted features, as is intuitively expected. The dependence on K for a specific digit pair (4 and 7) and deletion level (50 deletions) is shown in Figure 4. It can be seen that performance is improved up to a value of $K = 25$ which supposedly matches the deletion level in the data set (recall that FDROP considers a worst case scenario, whereas here features are dropped randomly, so that K and the number of deleted features should not be expected to be close numerically).

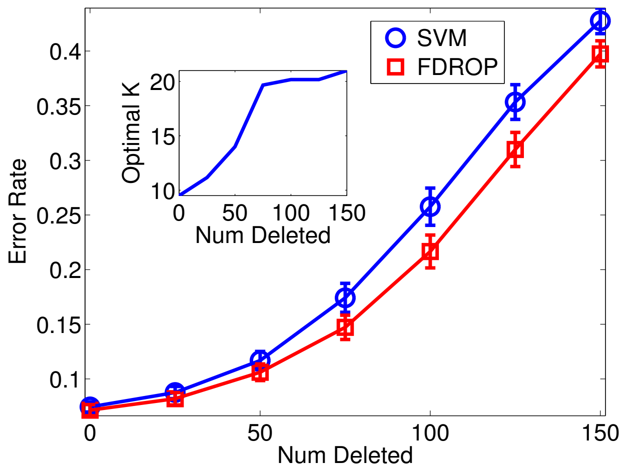

Figure 3. Classification error rate for the MNIST dataset, as a function of the number of features deleted from the test set. Standard errors over 20 repetitions are shown on the curve. The optimal K parameter for the FDROP algorithm was chosen per classification problem and per number of deleted features. The inset shows the optimal K for each deletion scheme.

4.3. Spam Filtering

One of the difficulties in filtering spam email from legitimate email is that the problem is dynamic in nature, in the sense that spam authors react to spam filters by changing content. In this sense, it is indeed a game where the two players are the spam filter and spam authors. Our formalism captures this competition, and it is therefore interesting to apply it to this case. To test FDROP, we applied it to the UCI SpamBase dataset, which consists of 4601 emails, 39.4% of which are spam. Each email is represented via a bag

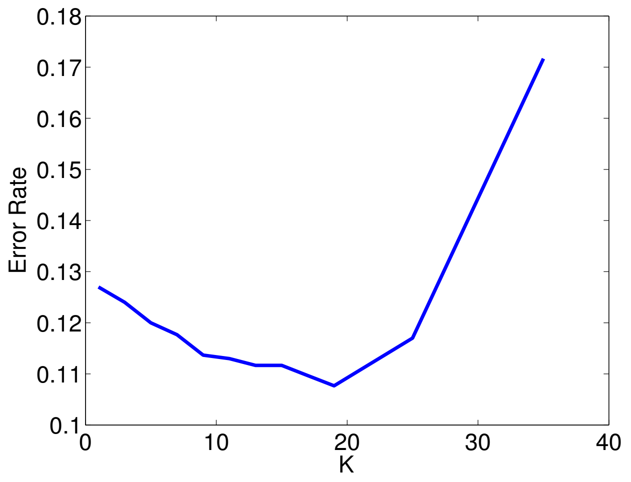

Figure 4. Classification error as a function of the parameter K for the digit pair $(4, 7)$ with 50 deleted features.

of words model with 57 terms. We also used a spam dataset collected by one of the authors from his personal mailbox. The latter contained 5000 emails with 40% spam, and 185 terms. For training we used 50 samples, a holdout set of 200, and a test set of 300.

Figure 5 shows the performance of the algorithm for different feature deletion levels, as in Figure 2. It can be seen that in this case the algorithm outperforms SVM even on the raw dataset, presumably because the original data is either non-stationary or the level of sparsity is such that SVM tends to overfit. Figure 6 shows the same analysis for our own spam dataset. Here again results are better than SVM for deletion levels up to 20, after which both algorithms converge to the apriori probability of the less common class.

5. Discussion & Conclusions

We have introduced a novel method for learning classifiers which are minimax optimal under a worst case scenario of feature deletion at test time. This is an important step towards extending statistical learning paradigms beyond the restrictive assumption that the training and testing data must come from the same (conditional) distribution. An alternative view of our algorithm is as a feature selection method which seeks the features which are most crucial for performance. A key assumption of our approach is that small sets of features should not be relied upon at test time to faithfully represent the class structure. Thus, in some sense, the features available to the algorithm at training time are viewed as being subject to random, or even deliberate removal at test time. Interestingly, a recent paper (Krupka & Tishby, 2006) presents a related view of features, where one considers a learning scheme where features are selected randomly from a large set, and generalization is studied with respect to unseen features. It will be interesting to study the relation between our approach and theirs.

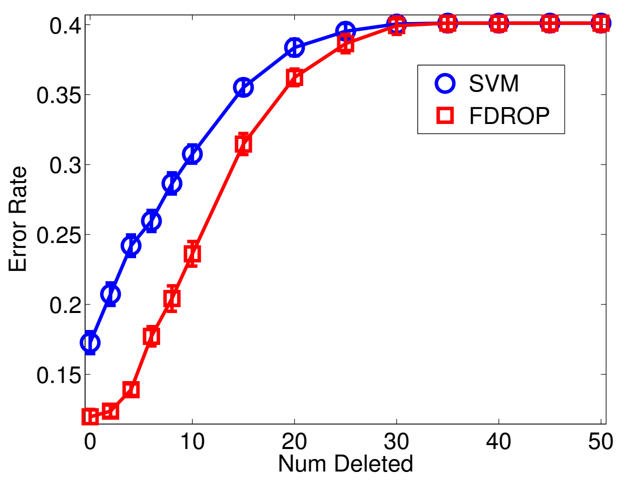

Figure 5. Classification error rate for the Spam-Base dataset, as a function of the number of features deleted from the test set. Setting as in Figure 2 with number of deleted features up to 50.

Another game theoretic approach to feature selection has previously appeared in (Cohen et al., 2005). Their approach is related to Shapley values in cooperative games. The Shapley value is a measure of the performance drop incurred by dropping a feature from a given set of features, where this performance is averaged over all subsets in which this feature participates. It is thus close in spirit to our feature elimination approach. However, our approach searches for multiple features simultaneously and is furthermore tractable, as opposed to exact calculation of Shapley values.

The extension of our method to kernel based classifiers is an interesting challenge, as mentioned in Section 2.1. It is also very interesting to consider the applicability of similar ideas to unsupervised methods such as principal component analysis or linear discriminant analysis.

A. Deriving the Dual Problem

Rewrite the problem in Equation 6 using auxiliary variables ξ_i to represent the hinge function.

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{s.t.} & \xi_i \geq 0 \quad , \quad \xi_i \geq 1 - y_i \mathbf{w}^T \mathbf{x}_i + t_i \\ & t_i \geq K z_i + \sum_j v_{ij} \\ & \mathbf{v}_i \geq 0 \quad , \quad z_i + \mathbf{v}_i \geq (y_i \mathbf{x}_i \circ \mathbf{w}) \end{array} \tag{14}$$

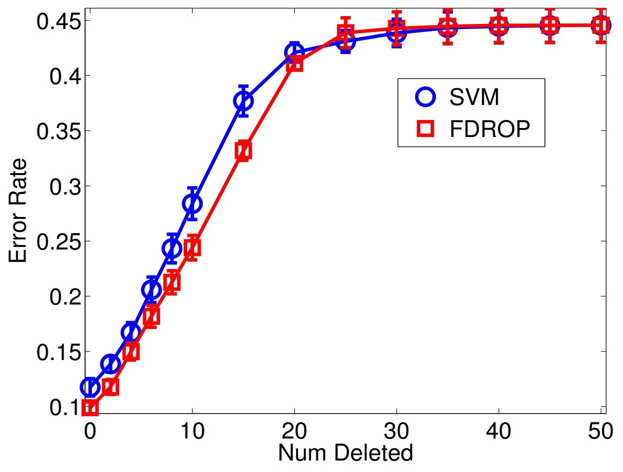

Figure 6. Classification error rate for our spam dataset, as a function of the number of features deleted from the test set. Setting as in Figure 5.

Define the following non-negative dual variables $\alpha_i^+, \alpha_i^-, \gamma_i, \boldsymbol{\delta}_i, \boldsymbol{\lambda}_i$ $(i = 1, \ldots, n)$ corresponding to the above five constraints, where $\boldsymbol{\delta}_i, \boldsymbol{\lambda}_i$ are of dimension d. The Lagrangian, $\mathcal{L}$, for the FDROP problem is a function of the above dual variables and of the primal variables $\xi_i, z_i, t_i, \mathbf{v}_i, \mathbf{w}$.

To obtain the minimum of $\mathcal{L}$ for a fixed set of dual variables, derive w.r.t the primal variables and equate to zero

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \mathbf{w} - \sum_i \left(\alpha_i^+ y_i \mathbf{x}_i - y_i \boldsymbol{\lambda}_i \circ \mathbf{x}_i \right) = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_i} &= C - \alpha_i^- - \alpha_i^+ = 0 \\ \frac{\partial \mathcal{L}}{\partial z_i} &= K\gamma_i - \sum_j \lambda_{ij} = 0 \\ \frac{\partial \mathcal{L}}{\partial \mathbf{v}_i} &= \gamma_i - \boldsymbol{\delta}_i - \boldsymbol{\lambda}_i = 0 \\ \frac{\partial \mathcal{L}}{\partial t_i} &= \alpha_i^+ - \gamma_i = 0 \end{aligned}$$

The second equation implies $\alpha_i^+ \leq C$. The fourth and fifth imply $\boldsymbol{\lambda}_i \leq \alpha_i^+$ (elementwise), and the third implies $\sum_j \lambda_{ij} = K\alpha_i^+$. The structure of the solution $\mathbf{w}$ is given by the first equation. Substituting all variables into the Lagrangian and some algebra results in

$$\min \mathcal{L} = -\frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \alpha_i^+ \qquad (15)$$

where $\mathbf{w}$ here is actually a function of the dual variables as implied by the equation $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$. The above minimum is the dual function (Boyd & Vandenberghe, 2004), whose maximization is equivalent to the original minimization. By rescaling $\boldsymbol{\lambda}_i$ according to $\hat{\boldsymbol{\lambda}}_i = \frac{\boldsymbol{\lambda}_i}{\alpha_i}$, we obtain Equations 7 and 8.

References

Bovik, A. C., Gibson, J. D., & Bovik, A. (Eds.). (2000). *Handbook of image and video processing*. Orlando, FL, USA: Academic Press, Inc.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York, NY, USA: Cambridge University Press.

Cohen, S., Ruppin, E., & Dror, G. (2005). Feature selection based on the shapley value. *IJCAI* (pp. 665–670). Professional Book Center.

El Ghaoui, L., Lanckriet, G., & Natsoulis, G. (2003). *Robust classification with interval data* (Technical Report UCB/CSD-03-1279). EECS Department, University of California, Berkeley.

Gilad-Bachrach, R., Navot, A., & Tishby, N. (2004). Margin based feature selection - theory and algorithms. *ICML 21* (pp. 43–50). ACM Press.

Kim, S., Magnani, A., & Boyd, S. (2006). Robust fisher discriminant analysis. *NIPS 18* (pp. 659–666). MIT Press.

Krupka, E., & Tishby, N. (2006). Generalization in clustering with unobserved features. *NIPS 18* (pp. 683–690). MIT Press.

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Sackinger, E., Simard, P., & Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. *ICANN* (pp. 53–60).

Schölkopf, B., & Smola, A. J. (Eds.). (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. *NIPS 13* (pp. 668–674). MIT Press.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. *ICML 14* (pp. 412–420). Morgan Kaufmann.