Automatic Feature Selection via Weighted Kernels and Regularization

Genevera I. Allen*

Abstract

Selecting important features in non-linear kernel spaces is a difficult challenge in both classification and regression problems. We propose to achieve feature selection by optimizing a simple criterion: a feature-regularized loss function. Features within the kernel are weighted, and a lasso penalty is placed on these weights to encourage sparsity. We minimize this feature-regularized loss function by estimating the weights in conjunction with the coefficients of the original classification or regression problem, thereby automatically procuring a subset of important features. Our algorithm, KerNel Iterative Feature Extraction (KNIFE), is applicable to a wide variety of kernels and high-dimensional kernel problems. In addition, a modification of KNIFE gives a computationally attractive method for graphically depicting non-linear relationships between features by estimating their feature weights over a range of regularization parameters. We demonstrate the utility of KNIFE in selecting features through simulations and examples for both kernel regression and support vector machines. Feature path realizations also give graphical representations of important features and the non-linear relationships among variables.

Keywords: Feature selection, Kernel methods, Support vector machine, Kernel ridge regression, Non-Linear Regression, Regularization Paths, Lasso.

[0]Department of Pediatrics-Neurology, Baylor College of Medicine, Jan and Dan Duncan Neurological Research Institute, Texas Children's Hospital, & Department of Statistics, Rice University, MS 138, 6100 Main St., Houston, TX 77005 (email: gallen@rice.edu)

1 Introduction

The merits of L_1-regularized linear regression and classification are well known. Regression methods such as the lasso and the elastic net and classification methods such as the L_1-SVM (support vector machine) and the L_1-logistic regression enjoy immense popularity due to their ability to select important variables in high-dimensions. The L_1-norm penalty, as the direct convex relaxation of the L_0-norm or best subsets penalty, also has many desirable theoretical properties (Candes and Plan, 2009). In addition, the linear L_1-norm regularized problem has an appealing form that is simply an extension of the original problem formulation, namely a loss function plus an L_1-norm on the coefficients. This direct approach via the L_1-norm penalty gives a disciplined method and criterion for feature selection in linear problems.

In non-linear problems, however, and especially kernel regression and classification problems, the L_1-norm penalty has had limited use due to numerical and computational challenges. Instead, indirect or heuristic methods are often employed. Filtering methods and subset methods, especially the popular Recursive Feature Elimination (Guyon et al., 2002) which removes features in a backwards stepwise manner are the most commonly used methods. Other methods such as RODEO (Lafferty and Wasserman, 2008) and the method of Bertin and Lecué (2008) perform feature selection for non-linear functions via local regressions. The later uses a lasso penalty to select features in a local polynomial kernel. The COSSO method (Lin and Zhang, 2007) also uses an L_1 penalty on the features, estimated in conjunction with smoothing splines. In addition, there are several regularization methods that perform feature selection specific to the linear SVM (Neumann et al., 2005; Zhu et al., 2003; Xu et al., 2009; Bach et al., 2004; Lanckriet et al., 2004), and some also that can be adapted for kernel SVMs (Wang, 2008; Navot and Tishby, 2004; Weston et al., 2000; Guyon, 2003). Many of the former, however, do not directly incorporate feature selection into the original problem formulation.

Weighting features within the kernel to achieve feature selection has been proposed in several methods (Cao et al., 2007; Li et al., 2006; Grandvalet and Canu, 2002; Navot and

Tishby, 2004; Argyriou et al., 2006; Weston et al., 2000). Most of these, however, do not directly optimize the original regression or classification problem, but instead seek to find a good set of weights on the features for later use within the kernel of the model. Grandvalet and Canu (2002), however, formulate a direct optimization problem for the support vector machine. They place an L_p-norm penalty on the feature weights, and develop an algorithm for the L_2-norm penalty with non-negative feature weights. An L_2 penalty, however, often does not encourage sufficient sparsity in the feature weights. They comment that an L_1 penalty would achieve greater feature selection, but never pursue this approach algorithmically, as the problem is non-convex and hence is not conducive to simple optimization. This, however, is the approach we will employ.

We summarize our regularization approach to kernel feature selection by giving our simple optimization criterion. Given a response, $\mathbf{Y}$, a feature-weighted kernel, $\mathbf{K_w}$ (defined in Section 2), coefficients, $\boldsymbol{\alpha}$, and feature weights, $\mathbf{w}$ (defined in Section 2), we optimize the following criterion:

$$\underset{\boldsymbol{\alpha},\mathbf{w}}{\text{minimize}} \quad L(\mathbf{Y}, \mathbf{K_w}\,\boldsymbol{\alpha}) \;+\; \lambda_1\,\boldsymbol{\alpha}^T\,\mathbf{K_w}\,\boldsymbol{\alpha} \;+\; \lambda_2||\,\mathbf{w}\,||_1 \tag{1}$$

$$\text{loss function} \quad + \text{coefficient regularization} \quad + \text{feature-weight regularization.}$$

Thus, feature selection is integrated into the original problem formulation through the feature-weighted loss function with an L_1 penalty on the feature weights. Selecting relevant features is achieved automatically by optimizing this criterion.

Our main contribution is a novel algorithm that finds a local minimum (1) for general kernel problems, meaning that it is applicable to any kernel problem that can be formulated via a loss function. These include, but are not limited to kernel ridge regression, kernel SVMs, kernel logistic regression, kernel discriminant analysis and kernel principal components analysis. As an extension, we give a method for visualizing non-linear relationships between features by estimating the feature weights over a range of regularization parameters,

similar to coefficient regularization paths. Hence, we achieve disciplined, automatic feature selection in high-dimensional kernel regression and classification problems.

The paper is organized as follows. In Section 2 we discuss weighted kernels and our optimization problem along with its mathematical and computational challenges. We present our main algorithm, KerNel Iterative Feature Extraction (KNIFE), in Section 3 and discuss minimization through kernel linearization. (Convergence results and further technical properties of KNIFE are given in Appendix A.) An extension of KNIFE is presented to visualizing non-linear feature relationships in Section 3.4. Section 4 gives simulation results and real data examples on gene selection for microarray data and feature selection in vowel recognition and ozone prediction data. We conclude with a discussion, Section 5.

2 KNIFE Problem

We propose to select important features by forming a regularized loss function that involves a set of weights on the features within a kernel. Before establishing the KNIFE problem, we briefly motivate the need for feature selection in non-linear kernel regression and classification problems.

Many have noted that non-linear kernel methods perform particularly poorly when irrelevant features are present (Bertin and Lecué, 2008; Lafferty and Wasserman, 2008; Wang, 2008; Lin and Zhang, 2007; Navot and Tishby, 2004; Grandvalet and Canu, 2002; Weston et al., 2000). Let us consider this mathematically, using the example of the SVM with polynomial kernels. Given data $\mathbf{x}_i \in \Re^p$ for $i = 1 \dots n$ observations and p features with the response $\mathbf{Y} \in \{-1, 1\}^n$. The kernel matrix, $\mathbf{K} \in \Re^{n \times n}$ is defined by $\mathbf{K}(i, i') = k(\mathbf{x}_i, \mathbf{x}_{i'}) = \left(\sum_{j=1}^{p} x_{ij} x_{i'j} + 1\right)^d$, for example with polynomial kernels. Recall that the support vector machine, presented in its primal form, can be written as an unconstrained minimization problem with the hinge loss (Wahba et al., 1999): $\text{minimize}_{\boldsymbol{\alpha}, \alpha_0} \quad \sum_{i=1}^{n} [1 - y_i \alpha_0 - y_i (\mathbf{K}\,\boldsymbol{\alpha})_i]_+ + \lambda\, \boldsymbol{\alpha}^T \mathbf{K}\, \boldsymbol{\alpha}$. Here, the coefficients are $\boldsymbol{\alpha} \in \Re^n$ and $\alpha_0 \in \Re$. Notice that each feature is given

the same weight in the kernel matrix, thus explaining the poor performance of SVMs for data with many irrelevant features. We, then, propose to place feature weights within the kernels to differentiate between the true and noise features.

2.1 Feature Weighted Kernels

We observed data, $\mathbf{x}_i \in \Re^p$ for $i = 1 \ldots n$ observations that can be written as a data matrix $\mathbf{X} \in \Re^{n \times p}$. We assume that $\mathbf{x}_i$ is standardized so that it has mean zero and variance one. For regression, we observe a response, $\mathbf{Y} \in \Re^n$, and for classification, $\mathbf{Y} \in \{-1, 1\}^n$. Our feature weighted kernels place a weight, $\mathbf{w} \in \Re^{p+}$, on each feature of the data within the kernel. Some examples for three common kernels are: inner product kernel, $k_{\mathbf{w}}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{p}(w_j x_j)(w_j x'_j)$, Gaussian or radial kernel, $k_{\mathbf{w}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \sum_{j=1}^{p}(w_j x_j - w_j x'_j)^2\right)$, polynomial kernel, $k_{\mathbf{w}}(\mathbf{x}, \mathbf{x}') = \left(\sum_{j=1}^{p}(w_j x_j)(w_j x'_j) + c\right)^d$. With these feature weighted kernels, we can define the weighted kernel matrix as $\mathbf{K}_{\mathbf{w}} \in \Re^{n \times n}$ such that $\mathbf{K}_{\mathbf{w}}(i, i') = k_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_{i'})$. Notice that the weights multiply each data feature in the kernel instead of placing a single feature weight within the kernel. (For example, in linear kernels, $(w_j x_j)(w_j x'_j)$ instead of $w_j x_j x'_j$). This seemingly minor detail is important for the kernel linearization part of our algorithm discussed in Section 3.2. The weights also play the role of a scaling parameter as well as a feature weight, similar to the adaptive scaling approach of (Grandvalet and Canu, 2002).

2.2 KNIFE Optimization Problem

We incorporate these feature weighted kernels into the regression or classification model. The response, $\mathbf{Y}$ is modeled by $\hat{\mathbf{Y}} = f(\mathbf{X})$, where $f(\mathbf{x}_i) = \sum_{i'=1}^{n} \alpha_i \mathbf{K}_w(i, i')$ for regression, or $f(\mathbf{x}_i) = \text{sign}(\alpha_0 + \sum_{i'=1}^{n} \alpha_i \mathbf{K}_w(i, i'))$ for classification with $\boldsymbol{\alpha} \in \Re^n$ the coefficients that must be estimated. For a positive definite kernel, $\mathbf{K}_{\mathbf{w}}$, and $f(\mathbf{X})$ a member of the reproducing kernel Hilbert space, $\mathcal{H}_{K_{\mathbf{w}}}$, this problem can be written as a minimization problem of the form: $\text{minimize}_{f \in \mathcal{H}_{K_{\mathbf{w}}}} \left[L(\mathbf{Y}, f(\mathbf{X})) + \lambda ||f||^2_{\mathcal{H}_{K_{\mathbf{w}}}} \right]$, where $L(\mathbf{Y}, f(\mathbf{X}))$ is the loss function. Some common examples of loss functions include the hinge loss (support vector machine),

squared error loss (regression) or binomial deviance loss (logistic regression).

To obtain a selection of important variables in a problem of this form, we need the weights to be both non-negative and sparse. To this end, we add an L_1 penalty on the weights and optimize over the set of non-negative weights that are less than one. (Note that the magnitude of the weights are limited to ensure non-degenerate scaling between $\boldsymbol{\alpha}$ and $\mathbf{w}$.) This gives the KNIFE optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha},\mathbf{w}}{\text{minimize}} \quad & f(\boldsymbol{\alpha},\mathbf{w}) = L(\mathbf{Y},\mathbf{K}_{\mathbf{w}}\,\boldsymbol{\alpha}) + \lambda_1\,\boldsymbol{\alpha}^T\,\mathbf{K}_{\mathbf{w}}\,\boldsymbol{\alpha} + \lambda_2 \mathbf{1}^T\,\mathbf{w} \\ \text{subject to} \quad & 0 \leq w_j < 1, \text{ for all } j = 1 \ldots p. \end{aligned} \tag{2}$$

Here, λ_1 and λ_2 are regularization parameters such that $\lambda_1 > 0$ and $\lambda_2 \geq 0$.

The KNIFE optimization problem, (2), is non-convex and it is therefore difficult to find a minimum, even for problems of small dimensions. One could approach this as a difference of convex programming problem (convex-concave programming), a direction taken in Argyriou et al. (2006). This approach splits the optimization problem with respect to $\mathbf{w}$ into a convex part associated with positive α_i's and a concave part associated with negative α_i's. It then optimizes the criterion by majorization minimization (MM), with full optimization achieved by alternating between MM steps using a cutting plane method to estimate $\mathbf{w}$ and traditional kernel regression or classification to estimate $\boldsymbol{\alpha}$. For solving (2), is computationally prohibitive in high-dimensional settings.

We seek to optimize our feature-regularized loss function with two main objectives: 1) to give a computationally tractable algorithm for non-linear feature selection in high-dimensions that leads to 2) an efficient method of visualizing non-linear relationships among the features and the response. To achieve these, we propose to circumvent the computational challenges of minimizing the non-convex KNIFE problem by linearizing non-linear kernels with respect to the feature weights. This approach is discussed in detail in the following sections.

3 KNIFE Algorithm

Given the KNIFE optimization problem based on the feature weighted kernels, we propose an algorithm to minimize the feature-regularized loss function for general kernel classification and regression methods. Our approach is to find a minimum by alternating between minimizing with respect to the coefficients, $\boldsymbol{\alpha}$, and the feature weights, $\mathbf{w}$.

3.1 Linear Kernels

We begin by outlining the KNIFE algorithm for linear kernels which forms the foundation of our approach to solving (2) non-linear kernels. With linear kernels, the kernel matrix becomes $\mathbf{K}_{\mathbf{w}} = \mathbf{X}\,\mathbf{W}\,\mathbf{X}^T$ where $\mathbf{W} = \text{diag}(\mathbf{w}^2)$. This gives the following objective function: $f(\boldsymbol{\alpha}, \mathbf{w}) = L(\mathbf{Y}, \mathbf{X}\,\mathbf{W}\,\mathbf{X}^T\,\boldsymbol{\alpha}) + \lambda_1\,\boldsymbol{\alpha}^T\,\mathbf{X}\,\mathbf{W}\,\mathbf{X}^T\,\boldsymbol{\alpha} + \lambda_2 \mathbf{1}^T\,\mathbf{w}$. Letting $\boldsymbol{\beta} = \mathbf{X}^T\,\boldsymbol{\alpha}$, we arrive at $f(\boldsymbol{\beta}, \mathbf{w}) = L(\mathbf{Y}, \mathbf{X}\,\mathbf{W}\,\boldsymbol{\beta}) + \lambda_1\,\boldsymbol{\beta}^T\,\mathbf{W}\,\boldsymbol{\beta} + \lambda_2 \mathbf{1}^T\,\mathbf{w}$. Here, notice that $f(\boldsymbol{\beta}, \mathbf{w})$ is a bi-convex function of $\boldsymbol{\beta}$ and $\mathbf{w}$, meaning that if we fix $\boldsymbol{\beta}$, $f(\cdot, \mathbf{w})$ is convex in $\mathbf{w}$ and if we fix $\mathbf{w}$, $f(\boldsymbol{\beta}, \cdot)$ is convex in $\boldsymbol{\beta}$. Recall that $\mathbf{w} < 1$, $\lambda_1 > 0$, and $\lambda_2 \geq 0$ so that the scaling between the parameters $\boldsymbol{\beta}$ and $\mathbf{w}$ does not permit degenerate solutions.

This biconvex property leads to a simple block-wise algorithm for minimization: minimize with respect to $\boldsymbol{\beta}$ and then with respect to $\mathbf{w}$. This block descent algorithm is monotonic, meaning that each iteration decreases the objective function $f(\boldsymbol{\beta}, \mathbf{w})$ and the algorithm converges. (See Appendix A).

Hence, for linear kernels, we are able to obtain a simple algorithm for minimizing the KNIFE problem because of the bi-convex property between the coefficients and the feature weights.

3.2 KNIFE Algorithm

For non-linear kernels, we propose to linearize kernels with respect to the feature weights to obtain a function convex in the weights and hence conducive to simple minimization of

the KNIFE objective, (2). From the previous section, we saw that if the kernel is linear in the feature weights, then we can apply a block-wise algorithm, estimating the coefficients and then estimating feature weights. This, then, is the motivation for our non-linear kernel algorithm which estimates coefficients, linearizes the kernel to obtain a surrogate objective, and estimates the feature weights from this surrogate function.

We linearize kernels as follows: Given the current estimate of the weights, $\mathbf{w}^{(t)}$, define the linearized kernel, $\tilde{k}_{\mathbf{w}}$ as: $\tilde{k}_{\mathbf{w}^{(t)}}(i,i') \triangleq k_{\mathbf{w}^{(t-1)}}(i,i') + \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i,i')^T(\mathbf{w}^{(t)} - \mathbf{w}^{(t-1)})$. Notice that $\tilde{k}_{\mathbf{w}}(i,i')$ is the linearization of the $(i,i')^{th}$ element of the kernel matrix $\mathbf{K}_{\mathbf{w}}$. Here, $\mathbf{w}^{(t-1)}$ is the weight vector from the previous iteration of the algorithm.

As previously mentioned in Section 2.1, we place a weight on each data feature within the kernel. This is integral to our linearization step. The advantages can be seen with an example of the gradient of a polynomial kernel: $\bigtriangledown k_{\mathbf{w}^{(t-1)}}(i,i')_k = 2d w_k^{(t-1)} x_{ik} x_{i'k} \left(\sum_{j=1}^{p} \left(w_j^{(t-1)} \right)^2 x_{ij} x_{i'j} + 1 \right)^{d-1}$. Notice that the gradient is scaled by the weights of the previous iteration, $\mathbf{w}^{(t-1)}$. Thus, if several weights were previously set to zero, the gradient in those directions is zero meaning that the weights will remain zero in all subsequent iterations of the algorithm (i.e. the weights are *sticky* at zero). This property, first, maintains sparsity in the feature weights throughout the algorithm, and secondly, limits the number of directions in which the weight vector can move in succeeding iterations. The former property allows us to approximate continuous feature paths (see Section 3.4) and the later property can be critical to algorithm convergence (see Appendix A).

With this linearization step, we form a surrogate objective function, $\tilde{f}(\boldsymbol{\alpha}, \mathbf{w})$ that is convex in the feature weights. This surrogate function is defined as follows: Let $\mathbf{B} \in \Re^{n \times n}$: $\mathbf{B}_{ii'} \triangleq k_{\mathbf{w}^{(t-1)}}(i,i') - \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i,i')^T \mathbf{w}^{(t-1)}$, and $\mathbf{A} \in \Re^{n \times p} : \mathbf{A}_{ii'} \triangleq \sum_{i'=1}^{n} \alpha_{i'} \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i,i')^T$, then the surrogate objective is $\tilde{f}(\boldsymbol{\alpha}, \mathbf{w}) \triangleq L(\mathbf{Y}, \mathbf{B}\,\boldsymbol{\alpha} + \mathbf{A}\,\mathbf{w}) + \lambda_1\, \boldsymbol{\alpha}^T\, \mathbf{A}\, \mathbf{w} + \lambda_2 \mathbf{1}^T \mathbf{w}$. Hence, the surrogate function is convex in $\mathbf{w}$, allowing for simple minimization. We outline the KNIFE algorithm for non-linear kernels in Algorithm 1:

The KNIFE algorithm iterates between finding the coefficients of the regression or clas-

Algorithm 1 KNIFE Algorithm

1. Initialize $\boldsymbol{\alpha}^{(0)}$ and $\mathbf{w}^{(0)}$ where $0 < w_j^{(0)} < 1$ for $j = 1 \ldots p$.
2. Set $\boldsymbol{\alpha}^{(t)} = \text{argmin}_{\boldsymbol{\alpha}} \left\{ L(\mathbf{Y}, \mathbf{K}_{\mathbf{w}^{(t-1)}}\, \boldsymbol{\alpha}) + \lambda_1\, \boldsymbol{\alpha}^T\, \mathbf{K}_{\mathbf{w}^{(t-1)}}\, \boldsymbol{\alpha} \right\}$.
3. Set $\mathbf{w}^{(t)}$ to the solution of:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \;\; & L(\mathbf{Y}, \mathbf{B}\, \boldsymbol{\alpha}^{(t)} + \mathbf{A}\, \mathbf{w}) + \lambda_1 (\boldsymbol{\alpha}^{(t)})^T\, \mathbf{A}\, \mathbf{w} + \lambda_2 \mathbf{1}^T\, \mathbf{w} \\ \text{subject to} \;\; & 0 \leq w_j < 1, j = 1, \ldots p, \end{aligned}$$

   where $\mathbf{B} \in \Re^{n \times n}$: $\mathbf{B}_{ii'} = k_{\mathbf{w}^{(t-1)}}(i, i') - \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i, i')^T\, \mathbf{w}^{(t-1)}$, $\mathbf{A} \in \Re^{n \times p}$: $\mathbf{A}_{ii'} = \sum_{i'=1}^{n} \alpha_{i'} \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i, i')^T$, and $\nabla k_{\mathbf{w}^{(t-1)}}(i, i')$ is the gradient of the (i, i') element of $\mathbf{K}_{\mathbf{w}^{(t-1)}}$ with respect to $\mathbf{w}^{(t-1)}$.
4. Repeat steps 2-3 until convergence.

sification problem and finding the sparse set of feature weights. Kernel linearization has allowed us to circumvent the difficulties associated with direct minimization of the feature regularized kernel loss function and formulate a computationally efficient algorithm for kernel feature selection.

We pause to make note of an interesting attribute of the KNIFE algorithm. By linearizing kernels with respect to the weights, the iterative optimization to estimate coefficients and weights are both problems of the same form but in different spaces. For example, with squared error loss, minimization with respect to the coefficients is a least squares problem in n dimensional space whereas minimization with respect to the weights in the linearized kernel is also a least squares problem in p dimensional feature space. We discuss further properties of the KNIFE problem and algorithm through comparisons to other methods in the next section.

3.3 Connections with Other Methods

While the KNIFE optimization problem may appear unfamiliar, there are several variations that are the same or similar in form to existing methodology. Let us consider linear kernels

for regression problems with squared error loss which can be written as:

$$\text{minimize} \quad ||\,\mathbf{Y} - \mathbf{X}\,\mathbf{W}\,\boldsymbol{\beta}\,||_2^2 + \lambda_1\,\boldsymbol{\beta}^T\,\mathbf{W}\,\boldsymbol{\beta} + \lambda_2||\,\mathbf{w}\,||_1 \;\text{ subject to }\; \mathbf{1} > \mathbf{w} \geq 0. \tag{3}$$

$$\text{or} \quad ||\,\mathbf{Y} - \mathbf{X}\,\tilde{\boldsymbol{\beta}}||_2^2 + \lambda_1\tilde{\boldsymbol{\beta}}^T\,\mathbf{W}^{-1}\,\tilde{\boldsymbol{\beta}} + \lambda_2||\,\mathbf{w}\,||_1 \;\text{ subject to }\; \mathbf{1} > \mathbf{w} \geq 0. \tag{4}$$

These are closely related to several common regression methods. (While in our problem, $\lambda_1 > 0$ and $\lambda_2 \geq 0$, for comparison purposes, we relax these constraints). First, if we let $\lambda_2 = 0$ and $\mathbf{w} = \mathbf{1}$ in (3), we get ridge regression. If we let $\lambda_1 = 0$, then KNIFE is closely related to the form of the non-negative garrote (Breiman, 1995). The form of both (3) and (4) is very similar to the elastic net which places an L_1 and L_2 penalty on the coefficients (Zou and Hastie, 2005). In KNIFE, however, the L_1 penalty is not on the coefficients, but on the weights that multiply the coefficients. Letting $\lambda_1 = 0$, we get a problem very similar in structure and intent to the lasso (Tibshirani, 1996). Also, if we let $\lambda_2 = 0$, then we have a problem that puts weights on the L_2 penalty of the coefficients. This is similar to the adaptive lasso which places weights on the L_1 penalty on the coefficients (Zou, 2006).

These similarities between other regression methods and KNIFE hold with other forms of loss functions also. For support vector machines, we have a problem similar to the L_0, L_1 and L_2 support vector machines in the same way that the inner-product squared error loss KNIFE relates to ridge and lasso regression. The same is true of L_0, L_1 and L_2 regularized logistic regression.

In addition to these methods, the COSSO (Component Selection and Smoothing Operator), which performs variable selection in smoothing splines, gives (3) and (4) exactly (Lin and Zhang, 2007) when the splines are linear. This method minimizes the sums of squares between the response and a function with an L_2 penalty on the projection of the function scaled by the inverse of a non-negative weight. This proposed form is also the form of (4). Additionally, the COSSO employs an algorithm which first fits a smoothing spline and then fits a non-negative garrote, noting that these steps can be repeated. This approximate

descent algorithm is similar to the KNIFE approach.

For support vector machines with the hinge loss, KNIFE with linear kernels is also closely related to multiple kernel learning (Lanckriet et al., 2004; Bach et al., 2004). Multiple kernel learning seeks to "learn" the kernel by replacing the typical kernel matrix in SVMs with the weighted sums of a set of linear kernels. Thus, one can form a linear kernel for each feature and combine them in a weighted sum (Xu et al., 2009), creating something similar to our feature weighted kernels. (Note that our feature weighted kernels contain the weights squared, instead of linear weights used in multiple kernel learning). Lanckriet, Cristianini, Bartlett, and Ghaoui (2004) show how this problem can be transformed into a convex optimization problem, namely a semi-definite program, while Bach, Lanckriet, and Jordan (2004) and Xu, Jin, Ye, Lyu, and King (2009) show that this is equivalent to a quadratically constrained quadratic program. Hence, if we re-parametrize our feature weighted linear kernels, KNIFE can also be written in the dual form as a convex problem via the multiple kernel learning framework. We note that as a reviewer pointed out, (4) is also jointly convex in $\tilde{\boldsymbol{\beta}}$ and $\mathbf{W}$.

3.4 Graphically Illustrating Non-Linear Feature Relationships

Since the KNIFE method uses regularization to extract important features, we can extend the KNIFE algorithm to graphically depict non-linear feature relationships. We compute the KNIFE solution using warm starts over a grid of regularization parameters. Connecting these solutions allows us to visualize the non-linear relationships among features as they relate to the response. We loosely call these feature paths, nothing that as a reviewer pointed out, these are not the result of a path algorithm such as Hastie et al. (2004), but instead a series of connected solutions.

With KNIFE, we have two penalty parameters, λ_1 and λ_2 and both of these parameters penalize the feature weights. The first penalty, λ_1 affects the feature weights through the kernel matrix, $K_{\mathbf{w}}$ and also penalizes the coefficients, $\boldsymbol{\alpha}$, while λ_2 places a direct L_1 penalty on

the weights. The later encourages sparsity in the feature weights. Hence, when formulating an algorithm to estimate feature paths, we focus on λ_2, fixing the value of λ_1. In general, setting $\lambda_1 = 1$, or if the loss function is given as $\frac{1}{n}L(\mathbf{Y}, f(\mathbf{X}))$, then $\lambda_1 = \frac{1}{n}$, performs well in all our simulations and examples, and is thus our default value for the remainder of the paper. Also, fixing λ_1 at a small value may especially be of interest in support vector machines as this permits a large margin and then allows the weights to both select features and further restrict the margin size.

Setting $\lambda_2 = 0$ gives no direct penalty on the feature weights and thus all features are permitted to be non-zero. Our algorithm computes solutions starting from $\lambda_2 = 0$ where all feature weights are non-zero to $\lambda_2 = M$, where M is the value at which all weights become zero. The grid of values between can be taken as 100 log-spaced values, an approach used in Friedman et al. (2010). The feature path algorithm is outlined in Algorithm 2.

Algorithm 2 KNIFE Feature Path Algorithm

1. Fix λ_1, set $\lambda_2 = 0$ and initialize $\boldsymbol{\alpha}^{(0)}$ and $0 < \mathbf{w}^{(0)} < 1$.
2. Fit KNIFE with $\boldsymbol{\alpha}^{(t-1)}$ and $\mathbf{w}^{(t-1)}$ as warm starts.
3. Increase λ_2.
4. Repeat Steps 2-3 until $\mathbf{w}_j^{(k)} = \mathbf{0}$ for all $j = 1, \ldots p$.

Two attributes of the KNIFE algorithm permit us to estimate feature paths in this manner. First, recall that we linearize kernels with respect to the weights, and in doing so, create an algorithm that is *sticky* at zero. Thus, as we increase λ_2 once a particular feature's weight is set to zero, it cannot ever become non-zero. This attribute permits us to efficiently use warm starts for the coefficients and weights, speeding computational time considerably. Additionally, it ensures that subsequent solutions are close to the previous solutions allowing us to connection solutions as to visualize the feature paths. Also, with warm starts and a small increase in λ_2, one can use a single update of the coefficients and weights at each iteration to approximate the feature paths. In addition, the *sticky* property

allows us to limit the features under current consideration in the algorithm to the active set, or the current set of features with non-zero weights. Hence, with all of these advantages, the computational time does not dramatically increase from that of the original KNIFE algorithm.

We briefly compare our feature path algorithm to the well known coefficient paths of the lasso and LAR (Least Angle Regression) algorithms (Osborne et al., 2000; Efron et al., 2004). In these regularization paths, the algorithm begins with no variables in the model and incrementally includes variables who are most correlated with the response. In our algorithm, however, we begin with all features in the model and incrementally eliminate the features that are uncorrelated in the kernel space with the response. Thus, the KNIFE path estimation algorithm can be thought of as a coherent regularization approach to the more heuristic backwards elimination for kernels. Also, the lasso regularization paths permit coefficient paths to cross zero and enter and re-enter the model. However, KNIFE does not allow this because of the *sticky* property of the feature weights, meaning that once a feature weight is set to zero it cannot move away from zero.

4 Results

We investigate the performance of the KNIFE algorithm and the KNIFE feature path algorithm on both real and simulated data.

4.1 Simulations

Two simulation examples are presented demonstrating the performance of KNIFE with kernel regression and kernel SVMs for non-linear regression and non-linear classification respectively. For both simulations, fifty training sets of size 100×10 and test sets of size 1000×10 were generated. Parameters for KNIFE and all comparison methods were selected on a separate validation set prior to training.

4.1.1 Non-linear Regression

We simulate a sinusoidal, non-linear regression problem with five true features and five noise features as follows: Let $\mathbf{x} \in \Re^p$ be standard normal with true coefficients, $\boldsymbol{\beta}^{true} = [6, -4, 3, 2, -2, 0, 0, 0, 0, 0]^T$. Then, the response is given by $y = \sin(\mathbf{x})\,\boldsymbol{\beta}^{true} + \epsilon$, where $\epsilon \sim N(0, 1)$. KNIFE with squared error loss and radial and second order polynomial kernels is compared to linear ridge regression, kernel ridge regression, the filtering method Sure Independence Screening (SIS) (Fan and Lv, 2008), Recursive Feature Elimination (RFE) (Guyon et al., 2002) (both with kernel ridge regression), and COSSO (Lin and Zhang, 2007). For a fair comparison, one parameter was validated for each method. For KNIFE, λ_2 is selected and λ_1 is fixed at $\lambda_1 = 1$, a default used in the remainder of the examples in this paper. For kernel ridge methods, the scale parameter, γ, for the radial kernel is set to $\gamma = 1/p$, a commonly used default.

Method	Training Error	Test Error	% Features	% Non-Features
Ridge	4.8337 (.1428)	6.2589 (.0771)	-	-
COSSO	2.2851 (.7584)	7.0126 (.6998)	94.0 (1.74)	31.7 (4.32)
Kernel Ridge (KR) - radial	0.0874 (.0035)	6.1239 (.0750)	-	-
SIS/ KR - radial	0.9592 (.0862)	3.8119 (.2752)	91.6 (1.63)	8.4 (1.63)
RFE/ KR - radial	0.7848 (.0526)	5.4386 (.2585)	83.6 (1.70)	36.4 (1.70)
KNIFE/ KR - radial	2.1187 (.0410)	**3.5498** (.0587)	**98.8** (3.39)	**2.4** (4.67)
KNIFE/ KR - polynomial	5.2376 (.1280)	6.8591 (.1315)	94.8 (7.42)	0.0 (0.00)

Table 1: *Simulation results for sinusoidal, non-linear regression. The data has ten features, five of which are noise features. Mean squared error on training and test sets are given with the standard errors in parenthesis. The average percentage of correct features and of noise features selected by the methods is also given. The best performing method is in bold.*

In Table 1, we report the mean squared error for the training and test sets over the fifty simulations. We see that KNIFE with radial kernels outperforms competing methods both in terms of mean squared error and in the correct selection of true features. KNIFE with radial kernels gives better error rates than second order polynomial kernels. Figure 1 presents example feature paths for both radial and polynomial kernels. We see that while the polynomial kernel gives much smoother feature paths, the radial kernel estimates the true

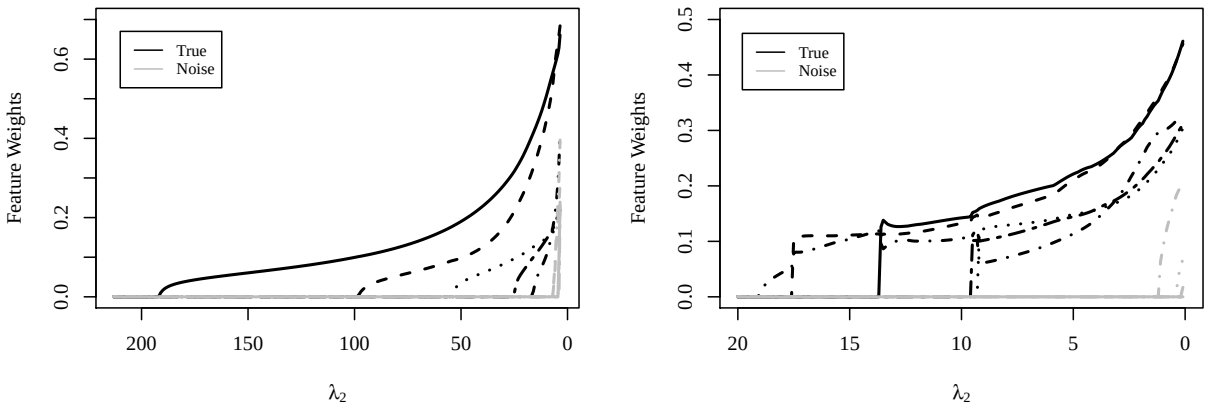

Figure 1: *Example feature paths for KNIFE with polynomial kernels of order 2 (left) and radial kernels (right) with a squared error loss function. Data of dimension* 100×10 *was simulated from the sinusoidal simulation with five true features and five noise features. KNIFE for both kernel types gives non-zero weights to the five true features for much of the feature paths.*

features for a much larger range of the regularization parameter. Moving from large values of the penalty parameter to smaller values, notice that the weights for the radial kernel shift when additional features are added to the model. This occurs as the feature weights also behave as automatic scaling factors, a point also noted in Grandvalet and Canu (2002).

4.1.2 Non-linear Classification

We use kernel support vector machines to assess KNIFE's performance on a non-linear classification simulation based on the skin of the orange example taken from Hastie et al. (2009). In this example, the first class has four standard normal features, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$, and the second class has the same conditioned on $9 \leq \sum_{j=1}^{4} \mathbf{x}_j^2 \leq 16$. Thus, the model has one class which is spherical with the second class surrounding the sphere like the skin of the orange. The two classes are not completely separable, however, giving a Bayes error of 0.0611. KNIFE with SVMs is compared to a standard SVM, SIS and RFE with SVMs and the adaptive scaling approach of Grandvalet and Canu (2002). All methods use second order polynomial kernels. We also note that KNIFE was used with the squared error hinge loss approximation to the hinge loss of the SVM. (See Appendix A for details.)

Method	Training Error	Test Error	% Features	% Non-Features
SVM	0 (0)	0.1919 (.0042)	-	-
SIS / SVM	0 (0)	0.2147 (.0082)	92.5 (1.63)	88.3 (1.09)
RFE / SVM	0.0872 (.0090)	0.2188 (.0097)	78.0 (2.44)	31.3 (1.65)
Adaptive Scaling / SVM	0.0374 (.0028)	0.1093 (.0040)	100.0 (0)	88.7 (1.81)
KNIFE / SVM	0.0366 (.0025)	**0.0986** (.0043)	**100.0** (0)	**16.7** (1.98)

Table 2: *Average misclassification errors for the skin of the orange simulation with four true features and six noise features. All methods use support vector machines with second order polynomials. Additionally, the average percentage of correct features and of noise features selected by the methods is given. Standard errors are in parenthesis with the best performing method in bold.*

In Table 2, we present results on the skin of the orange simulation in terms of test and training misclassification error and the percentage of true and noise features selected. Here, KNIFE outperforms competing methods in terms of test error and selection of true features. Also, notice that the adaptive scaling method performs similarly to KNIFE in terms of error rates, but is less selective of features as a large percentage of noise features are selected. Recall that this method employs a similar objective to KNIFE, but with an L_2 penalty instead of an L_1 penalty. Results on this simulation clearly illustrate the advantages of using an $L_!$ penalty in this context for feature selection.

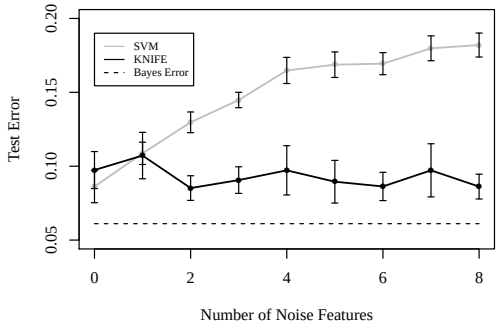

Figure 2: *Mean test misclassification error with standard errors when noise features are added to the skin of the orange simulation. SVMs and KNIFE for SVMs with second order polynomial kernels are trained on data with 100 observations and tested on data with 1000 observations. The Bayes error of this simulation is 0.0611. By selecting the true features, the error rates for KNIFE remain constant as the number of noise features in the model increase.*

To explore the behavior of KNIFE when the number of noise features in the mode in-

creases, we begin with the skin of the orange model with four true features and add noise features. Results compared to the standard SVM are shown in Figure 2. We see that the misclassification errors for KNIFE remain stable as the number of noise features in the model increase, whereas those of the standard SVM increase dramatically.

Also, we pause to comment on an interesting characteristic of KNIFE for support vector machines. The SVM is sparse in the observation space, meaning that only a subset of the observations are chosen as "support vectors". With KNIFE/SVM, we get sparsity in the feature space also, leaving us with an important sub-matrix of observations and features that can limit computational storage for prediction purposes.

4.2 Examples

We apply KNIFE to three feature selection applications: gene selection in microarrays, variable selection in vowel recognition data, and variable selection in predicting ozone levels.

4.2.1 High-Dimensional Data: Microarrays

With thirty thousand human genes, doctors often need a small subset of genes to test that are predictive of a disease. To demonstrate the performance of KNIFE for gene selection, we apply our method to microarray data on colon cancer, publicly available at http://genomics-pubs.princeton.edu/oncology/ (Alon et al., 1999). The data set consists of 62 samples, 22 of which are normal and 40 of which are from colon cancer tissues. The genes are already pre-filtered, consisting of the 2,000 genes with the highest variance across samples.

For this analysis, we use a linear SVM for classification, comparing KNIFE to the L_1-norm SVM (Zhu et al., 2003) and gene filtering using SIS and RFE. We evaluate eight subsets of previously fixed numbers of genes on the three methods. For the gene selection with RFE, we begin by eliminating 50 genes, ten genes and then one gene at each step as outlined in Guyon et al. (2002). To determine predictive ability, we split the samples randomly into training and test sets of equal sizes. This is repeated ten times and misclassification rates

# Genes	SIS/SVM	RFE/SVM	L_1-Norm/SVM	KNIFE/SVM
500	**0.0452** (.016)	0.1290 (.016)	0.1097 (.022)	0.0484 (.019)
250	0.0581 (.022)	0.1452 (.020)	0.1032 (.023)	**0.0516** (.018)
100	**0.0548** (.019)	0.1677 (.021)	0.1097 (.022)	0.0710 (.0054)
50	**0.0677** (.0047)	0.1774 (.0065)	0.1097 (.0074)	0.0806 (.018)
25	0.0968 (.017)	0.1742 (.013)	0.1097 (.023)	**0.0871** (.017)
15	0.1161 (.019)	0.1484 (.016)	0.1161 (.022)	**0.1065** (.019)
10	**0.1194** (.027)	0.1581 (.019)	0.1355 (.021)	**0.1194** (.019)

Table 3: *Average misclassification rates with standard errors on ten randomly created test sets trained on a set of equal size for the colon cancer microarray data. All methods use a linear support vector machine. Best performing methods are in bold.*

are averaged with results given in Table 3.

The results indicate that KNIFE outperforms RFE filtering and the L_1-norm SVM for all subsets of genes, and performs similarly to the SIS filtering method with a small advantage when selecting small subsets of genes. While the subsets of genes determined by SIS may perform similarly in terms of classification, often researchers are interested in a subset of genes that are members of different pathways and are hence less correlated. For SIS filtering, the median pair-wise correlation of genes selected is 0.665 and 0.714 for 25 and 10 genes respectively, compared to 0.590 and 0.590 for KNIFE. The later is close to the median correlation seen among genes int eh data set as a whole at 0.591. KNIFE, then, selects genes that classify the sample well and are less correlated.

4.2.2 Non-Linear Classification: Vowel Recognition

As an example of KNIFE for non-linear classification, we apply our method to the vowel recognition data set available at http://www-stat.stanford.edu/ tibs/ElemStatLearn/ (Hastie et al., 2009). This data consists of eleven classes of vowels broken down into ten features in which fifteen individuals were recorded speaking six times. We apply a radial kernel SVM and KNIFE / SVM to classify between vowel sounds 'i' and 'I' based on ten features (related to the log periodogram) shown on the left in Figure 3. We use five fold cross-validation to determine the margin size for the SVM and the λ_2 value for KNIFE. Both methods were

trained on a data set with 48 instances and tested with 42 instances of each vowel. KNIFE gives a test misclassification error of 8.3% while the SVM gives an error of 19.1%. We see from the example feature paths in Figure 3 that KNIFE selects six features that are indicative of the two vowel types and gives a visual representation of the highly non-linear relationship between the features and vowel sounds.

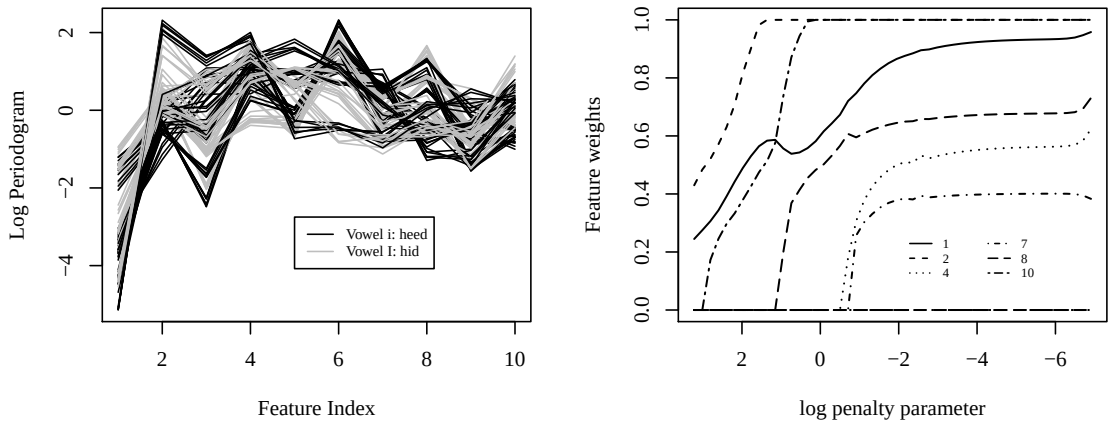

Figure 3: *Vowel recognition data for vowels 'i' and 'I' (left) with KNIFE estimated feature paths for classifying between the two vowels using a support vector machine with radial kernels (right). Five fold cross-validation on the training set chose to include the six features shown. KNIFE gives a test misclassification error of 8.3%, while a radial kernel SVM has 19.1% test error.*

4.2.3 Non-Linear Regression: Ozone Data

The ozone data set (Breiman and Friedman, 1985), available in the COSSO R package (Zhang and Lin, 2010) , has been commonly used to compare methods for non-linear regression. The data consists of 330 daily readings of ozone levels and eight predictors from Los Angeles in 1976. We compare KNIFE using L_2 loss with radial kernels to the COSSO non-linear regression method (Lin and Zhang, 2007) by randomly splitting the data into ten test and training sets of equal sizes. Five fold cross-validation was used to selected penalty parameters for both methods. KNIFE performs well in terms of mean squared error yielding a training and test error of 14.27 (.40) and 16.96 (.39) respectively (standard errors are in parenthesis), while COSSO yields errors of 15.28 (.36) and 18.57 (.36) respectively. In Figure 4, KNIFE

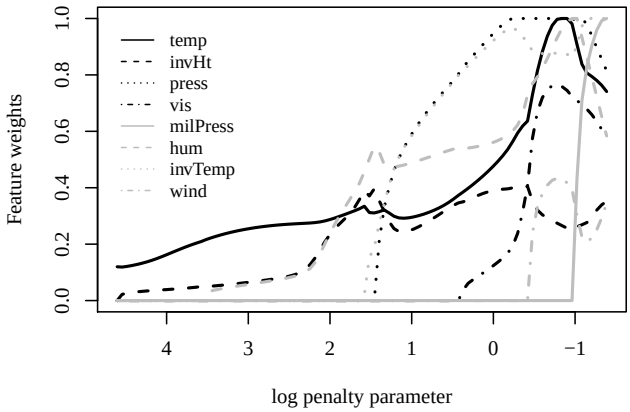

Figure 4: *Non-linear feature relationships estimated by KNIFE with radial kernels to predict ozone levels based upon eight predictors. Five-fold cross-validation selects six variables, all except wind speed and millibar pressure. On ten random splits of this data into test and training sets, KNIFE yields a training and test mean squared error of 14.17 (.40) and 16.96 (.39) respectively, compared to COSSO at 15.28 (.36) and 18.57 (.36) respectively.*

also reveals important non-linear relationships between the predictors and the ozone levels. Temperature is the most important variable in predicting ozone levels followed by inversion height (invHt) and humidity (hum). These variables are predictive of ozone in largely independent manners as seen by the steady slopes of the feature estimates. When the variables barometric pressure and inversion temperature enter the model, the feature weights for temperature, inversion height and humidity shift, indicating that there are co-linearities, in the non-linear kernel space, between these two sets of variables. Five-fold cross-validation for KNIFE on the full data set gives non-zero weight to all but two predictors, wind speed and millibar pressure.

5 Discussion

We have presented an algorithm to minimize a feature-regularized loss function, thus achieving automatic feature selection in non-linear kernel spaces.

Computationally, the KNIFE algorithm compares favorably to existing kernel feature selection methods with the exception of simple feature filtering methods. The KNIFE al-

gorithm iterates between an optimization problem in n-dimensional feature space and then in p-dimensional feature space. This is similar to the computational burden of the adaptive scaling procedure of Grandvalet and Canu (2002), but is more efficient than the DC programming approach of Argyriou et al. (2006). In addition, one can fun the KNIFE algorithm for a limited number of iterations to further restrict computational costs.

While we have presented specific examples with kernel ridge regression and kernel SVMs, the KNIFE method is applicable to a variety of kernels problems that can be written with a loss function. These include kernel logistic regression, which has a binomial deviance loss; and kernel principal component analysis, kernel discriminant analysis and kernel canonical correlation, which all can be written with a Frobenius norm loss. Thus, the KNIFE method has many potential future uses for feature selection in a variety of kernel methods. Also, we have presented the KNIFE problem for general kernel regression or classification problems. We save problem-specific versions of the KNIFE algorithm for future work.

In conclusion, we have presented a coherent criterion for selecting features in kernel problems via the feature-regularized kernel loss function. Our methods are applicable to general kernel problems, are computationally feasible in high-dimensional settings, and give visual representations of the relationships between variables. Thus, KNIFE provides a valuable tool for feature selection with non-linear kernel problems.

6 Acknowledgments

We are grateful to Robert Tibshirani for the helpful suggestions and advice in developing and testing this method. We would also like to thank Stephen Boyd for suggesting kernel linearization, Rahul Mazumder and Holger Hoefling for discussions on algorithm convergence, and Trevor Hastie for the helpful suggestions. We also thank three anonymous reviewers, the editor, and associate editor for suggestions that led to several improvements in this paper.

References

Alon, U., N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine (1999, June). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences 96*(12), 6745–6750.

Argyriou, A., R. Hauser, C. A. Micchelli, and M. Pontil (2006). A dc-programming algorithm for kernel selection. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pp. 41–48.

Bach, F. R., G. R. G. Lanckriet, and M. I. Jordan (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*.

Bertin, K. and G. Lecué (2008). Selection of variables and dimension reduction in high-dimensional non-parametric regression. *Electronic Journal of Statistics 2*, 1224–1241.

Bertsekas, D., A. Nedi, and A. Ozdaglar (2003). *Convex analysis and optimization*. Athena Scientific.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge Univ Pr.

Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics 37*(4), 373–384.

Breiman, L. and J. Friedman (1985). Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association 80*(391), 580–598.

Candes, E. and Y. Plan (2009). Near-ideal model selection by ℓ_1 minimization. *Annals of Statistics 37*(5A), 2145–2177.

Cao, B., D. Shen, J. Sun, Q. Yang, and Z. Chen (2007). Feature selection in a kernel space. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pp. 121–128.

Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani (2004). Least angle regression. *Annals of Statistics 32*, 407–499.

Fan, J. and J. Lv (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal Of The Royal Statistical Society Series B 70*(5), 849–911.

Friedman, J., T. Hastie, and R. Tibshirani (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software 33*(1), 1–22.

Grandvalet, Y. and S. Canu (2002). Adaptive scaling for feature selection in svms. In *Advances in Neural Information Processing Systems 15*.

Guyon, I. (2003). Multivariate nonlinear feature selection with kernel multiplicative updates and gram-schmidt relief. In *BISC FLINT-CIBI 2003 workshop*.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*.

Hastie, T., S. Rosset, R. Tibshirani, and J. Zhu (2004). The entire regularization path for the support vector machine. *The Journal of Machine Learning Research 5*, 1391–1415.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *Elements of Statistical Learning* (2 ed.). Springer New York Inc.

Lafferty, J. and L. Wasserman (2008). Rodeo: Sparse, greedy nonparametric regression. *Ann. Stat. 36*(arXiv:0803.1709. IMS-AOS-AOS0318. 1), 28–63.

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

Li, F., Y. Yang, and E. Xing (2006). From lasso regression to feature vector machine. In *Advances in Neural Information Processing Systems 18*, pp. 779–786.

Lin, Y. and H. H. Zhang (2007, February). Component selection and smoothing in multivariate nonparametric regression. *Annals of Statistics 34*(5), 2272–2297.

Navot, A. and N. Tishby (2004). Margin based feature selection - theory and algorithms. In *International Conference on Machine Learning (ICML*, pp. 43–50.

Neumann, J., C. Schnörr, and G. Steidl (2005). Combined svm-based feature selection and classification. *Machine Learning 61*, 129–150.

Osborne, M., B. Presnell, and B. Turlach (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis 20*(3), 389.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological) 58*(1), 267–288.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal Optimization Theory and Applications 109*(3), 475–494.

Wahba, G., Y. Lin, and H. Zhang (1999). Generalized approximate cross validation for support vector machines, or, another way to look at margin-like quantities. Technical report, University of Wisconsin.

Wang, L. (2008). Feature selection with kernel class separability. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*(9), 1534–1546.

Wang, L., J. Zhu, and H. Zou (2008). Hybrid huberized support vector machines for microarray classification and gene selection. *Bioinformatics 24*(3), 412–419.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik (2000). Feature selection for svms. In *Advances in Neural Information Processing Systems 13*, pp. 668–674.

Xu, Z., R. Jin, J. Ye, M. R. Lyu, and I. King (2009). Non-monotonic feature selection. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*.

Zhang, H. H. and C.-Y. Lin (2010). *COSSO*. R package version 1.0-1.

Zhu, J., S. Rosset, T. Hastie, and R. Tibshirani (2003). 1-norm support vector machines. In *Neural Information Processing Systems*, pp. 16.

Zou, H. (2006, December). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association 101*, 1418–1429.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B 67*, 301–320.

A Appendix: Convergence of KNIFE

We will show that the KNIFE algorithm, Algorithm 1, is an approximation to a descent algorithm that under certain conditions on the loss function and kernel, converges to a local minimum of the KNIFE objective function. We also discuss the properties of this descent algorithm and the KNIFE algorithm, showing through numerical examples the converge of the KNIFE algorithm. In special cases, we also show that the KNIFE algorithm converges to a stationary point of the KNIFE objective.

Again, we seek to minimize the following problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha},\mathbf{w}}{\text{minimize}} \quad & f(\boldsymbol{\alpha},\mathbf{w}) = L(\mathbf{Y},\mathbf{K}_{\mathbf{w}}\,\boldsymbol{\alpha}) + \lambda_1\,\boldsymbol{\alpha}^T\,\mathbf{K}_{\mathbf{w}}\,\boldsymbol{\alpha} + \lambda_2 \mathbf{1}^T\,\mathbf{w} \\ \text{subject to} \quad & 0 \le w_j < 1, \text{ for all } j = 1 \dots p. \end{aligned}$$

Consider the following algorithm:

Algorithm 3 KNIFE Descent Algorithm

1. Initialize $\boldsymbol{\alpha}^{(0)}$ and $\mathbf{w}^{(0)}$ where $0 < w_j^{(0)} < 1$ for $j = 1 \dots p$.
2. Set $\boldsymbol{\alpha}^{(t)} = \text{argmin}_{\boldsymbol{\alpha}} \left\{ L(\mathbf{Y},\mathbf{K}_{\mathbf{w}^{(t-1)}}\,\boldsymbol{\alpha}) + \lambda_1\,\boldsymbol{\alpha}^T\,\mathbf{K}_{\mathbf{w}^{(t-1)}}\,\boldsymbol{\alpha} \right\}$.
3. Estimate $\mathbf{w}^{(t)}$:
   (a) Define $\tilde{f}(\boldsymbol{\alpha}^{(t)},\mathbf{w}^{(t)}) = L(\mathbf{Y},\mathbf{B}\,\boldsymbol{\alpha}^{(t)} + \mathbf{A}\,\mathbf{w}) + \lambda_1(\boldsymbol{\alpha}^{(t)})^T\,\mathbf{A}\,\mathbf{w} + \lambda_2 \mathbf{1}^T\,\mathbf{w}$ where $\mathbf{B} \in \Re^{n\times n} : \mathbf{B}_{ii'} = k_{\mathbf{w}^{(t-1)}}(i,i') - \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i,i')^T\,\mathbf{w}^{(t-1)}$, $\mathbf{A} \in \Re^{n\times p} : \mathbf{A}_{ii'} = \sum_{i'=1}^{n} \boldsymbol{\alpha}_{i'}^{(t)} \bigtriangledown k_{\mathbf{w}^{(t-1)}}(i,i')^T$, and $\nabla k_{\mathbf{w}^{(t-1)}}(i,i')$ is the gradient of the (i,i') element of $\mathbf{K}_{w^{(t-1)}}$ with respect to $\mathbf{w}^{(t-1)}$.
   (b) Compute a descent direction, $\triangle\,\mathbf{w}$, for $\tilde{f}(\boldsymbol{\alpha}^{(t)},\mathbf{w}^{(t)})$ with respect to $\mathbf{w}^{(t)}$ over the set $\{\mathbf{w} : 0 \le \mathbf{w}_j^{(t)} < 1 \text{ for } j = 1, \dots p\}$.
   (c) Conduct a line search: $s = \text{argmin}_{u\ge 0} f(\boldsymbol{\alpha},\mathbf{w}^{(t-1)} + u \triangle\,\mathbf{w})$
   (d) Set $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + s \triangle\,\mathbf{w}$.
   (e) Repeat steps (a) - (d) until convergence.
4. Repeat steps 2-3 until convergence.

Proposition 1. *Let the loss function, $L(\mathbf{Y},\mathbf{K}_{\mathbf{w}}\,\boldsymbol{\alpha})$, be a convex and continuously differentiable function of $\boldsymbol{\alpha}$ and let the weighted kernel, $k_{\mathbf{w}}(\mathbf{x}_i,\mathbf{x}_{i'})$, be a convex or concave and continuously differentiable function of $\mathbf{w}$. Then, Algorithm 3, is a descent algorithm and converges to a local minimum of the objective $f(\boldsymbol{\alpha},\mathbf{w})$.*

Proof. We will show that estimation with respect to $\boldsymbol{\alpha}$, Step 2, and with respect to $\mathbf{w}$, Step 3, in the KNIFE descent algorithm monotonically decreases the objective $f(\boldsymbol{\alpha},\mathbf{w})$. Since this objective is bounded below by zero, this ensures convergence. We will also show that the solution to this algorithm is a local minimum, meaning that at the solution $(\boldsymbol{\alpha}^*,\mathbf{w}^*)$, $\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^*,\mathbf{w}^*) = 0$ and $\nabla_{\mathbf{w}} f(\boldsymbol{\alpha}^*,\mathbf{w}^*) = 0$.

It is easy to see that block-wise minimization with respect to $\boldsymbol{\alpha}$, Step 2, solves a convex problem. Therefore Step 2 decreases the objective and satisfies $\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^{(t)}, \mathbf{w}^{(t)}) = 0$ after each step.

Next, we consider estimation with respect to $\mathbf{w}$ in Step 3. It is obvious that if $\nabla_{\mathbf{w}} f(\boldsymbol{\alpha}^{(t)}, \mathbf{w}^{(t-1)}) = 0$, Step 3 returns $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)}$ and this step is non-increasing. Then, we consider the case where $\mathbf{w}^{(t-1)}$ is not an optimal point. Without loss of generality and to avoid notational complexities we can consider minimizing an objective $f(\mathbf{w}) = h(g(\mathbf{w}))$ where $h()$ is a convex and continuously differentiable function of $\mathbf{w}$ and $g()$ is a convex or concave and continuously differentiable function of $\mathbf{w}$. In this notation, if we denote the previous solution as $\mathbf{w}'$, Step 3 computes a descent direction of $\tilde{f}_{\mathbf{w}'}(\mathbf{w}) = h(g(\mathbf{w}') + \nabla g(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}'))$. Applying standard definitions, it is easy to see that since $h()$ and $g()$ are continuously differentiable and hence locally Lipshcitz continuous, that $f()$ is also locally Lipshcitz in $\mathbf{w}$. Furthermore, since $h()$ and $g()$ are convex or concave, there exists an open neighborhood of any $\mathbf{w}'$ such that $\nabla g(\mathbf{w}') \neq 0$ and $\nabla h(\mathbf{w}') \neq 0$ in which $h()$ and $g()$ are monotonic (Bertsekas et al., 2003). Putting these two facts together, we have that there exists an open neighborhood of $\mathbf{w}'$, $\mathcal{N}(\mathbf{w}')$, for all $\mathbf{w}'$ such that $\nabla_{\mathbf{w}} f(\mathbf{w}') \neq 0$ in which $f(\mathbf{w})$ for $\mathbf{w} \in \mathcal{N}(\mathbf{w}')$ is monotonic. We will then restrict our consideration to four cases in this monotonic neighborhood: $f()$ is strictly decreasing or non-decreasing with $g()$ convex or $f()$ is strictly increasing or non-increasing with $g()$ concave.

Let us first consider the strictly decreasing case with $g()$ convex. Letting $\mathbf{w}^*$ be a minimum of $\tilde{f}_{\mathbf{w}'}(\mathbf{w})$ in $\mathcal{N}(\mathbf{w}')$, we have

$$
\begin{aligned}
g(\mathbf{w}) &\geq g(\mathbf{w}') + \nabla g(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}') \quad \forall\, \mathbf{w}, \mathbf{w}' && \because g() \text{ is convex.} \\
h(g(\mathbf{w}^*)) &\leq h\left(g(\mathbf{w}') + \nabla g(\mathbf{w}')^T(\mathbf{w}^* - \mathbf{w}')\right) && \because h() \text{ is strictly decreasing.} \\
f(\mathbf{w}^*) &\leq \tilde{f}_{\mathbf{w}'}(\mathbf{w}^*) \leq \tilde{f}_{\mathbf{w}'}(\mathbf{w}') && \because \text{by definition of } \mathbf{w}^*. \\
&\leq f(\mathbf{w}') && \because \tilde{f}_{\mathbf{w}'}(\mathbf{w}') = f(\mathbf{w}').
\end{aligned}
$$

Therefore, Step 3 decreases the original objective when $f()$ is strictly decreasing at $\mathbf{w}^{(t-1)}$. (Note that the line search ensures that $\mathbf{w}^{(t)}$ must remain in a neighborhood in which $f()$ is strictly decreasing).

Now, when $f(\mathbf{w}')$ is non-decreasing and $g()$ is convex, we have that $f()$ is locally convex (Bertsekas et al., 2003), which can be seen by considering the Hessian of $f()$ (Boyd and Vandenberghe, 2004). Then, taking the descent direction $\triangle \mathbf{w}$ to be $\triangle \mathbf{w} = -\nabla \tilde{f}_{\mathbf{w}'}(\mathbf{w})$, we have a gradient descent step on $\tilde{f}()$. But, since $\nabla \tilde{f}_{w'}(\mathbf{w}) = \nabla f(\mathbf{w}')$, this also is a gradient descent step on $f()$. Again, the line search ensures that only descent steps are taken and that $\mathbf{w}^{(t)}$ remains in a neighborhood which is non-decreasing. The proof of convergence for Step 3, then follows the convergence analysis of gradient descent algorithms (Boyd and Vandenberghe, 2004).

The arugment when $g()$ is concave is analogous. Consider the case when $f()$ is strictly increasing:

$$g(\mathbf{w}) \leq g(\mathbf{w}') + \nabla g(\mathbf{w}')^T(\mathbf{w} - \mathbf{w}') \quad \forall\, \mathbf{w}, \mathbf{w}' \qquad \because g() \text{ is concave.}$$

$$h(g(\mathbf{w}^*)) \leq h\left(g(\mathbf{w}') + \nabla g(\mathbf{w}')^T(\mathbf{w}^* - \mathbf{w}')\right) \qquad \because h() \text{ is strictly increasing,}$$

and the remainder of the argument is identical to the above case. Similarly, when $f(w')$ is non-increasing and $g()$ is concave, we have that $f()$ is locally convex (Bertsekas et al., 2003). Again, the remainder of the argument is identical to the above case.

Finally, while $\nabla_{\mathbf{w}} f(\boldsymbol{\alpha}^{(t)}, \mathbf{w}^{(t)})$ may not necessarily equal zero after each step, this gradient condition will be satisfied at the solution. This occurs as from the above argument, if $\mathbf{w}^{(t-1)}$ is not an optimal point, then there exists a feasible descent direction and convergence has not been achieved. Therefore, we have shown that estimation of $\mathbf{w}^{(t)}$ in Step 3 necessarily decreases the objective $f(\boldsymbol{\alpha}^{(t)}, \mathbf{w})$, and that Algorithm 3 is a descent algorithm that converges to a local minimum of $f(\boldsymbol{\alpha}, \mathbf{w})$.

□

Before discussing how this descent algorithm relates to the KNIFE algorithm, Algorithm 1, some additional remarks are warranted. First, the exact line search can be replaced by a backtracking or other line searching method. The result and proof remain unchanged as the argument proving convergence of gradient descent methods with the various line searches is employed. Second, notice that the arguments in the proof indicate that for all $\mathbf{w}^{(t-1)}$ that are not optimal points ($\nabla_{\mathbf{w}} f(\boldsymbol{\alpha}^{(t)}, \mathbf{w}^{(t-1)}) = 0$), there exists a neighborhood of $\mathbf{w}^{(t-1)}$ in which minimizing $\tilde{f}_{\mathbf{w}^{(t-1)}}(\boldsymbol{\alpha}, \mathbf{w})$ with respect to $\mathbf{w}$ will decrease the objective $f(\boldsymbol{\alpha}, \mathbf{w})$. In the KNIFE descent algorithm, the line search ensures that each step remains in this neighborhood.

The KNIFE algorithm replaces the descent direction and line search for estimating $\mathbf{w}^{(t)}$ with a full minimization of the linearized objective $\tilde{f}_{\mathbf{w}^{(t-1)}}(\boldsymbol{\alpha}, \mathbf{w})$ with respect to $\mathbf{w}$. Thus, the KNIFE algorithm is an approximation to the descent algorithm, and is not guaranteed to strictly decrease the objective at each iteration. Since there always exists a neighborhood of $\mathbf{w}^{(t-1)}$ in which the objective will decrease, however, one can restrict the range considered either explicitly, $||\, \mathbf{w}^{(t-1)} - \mathbf{w}^{(t)}\, || \leq c$ or implicitly through adding a penalty, $\mu|| \mathbf{w}^{(t-1)} - \mathbf{w}^{(t)} ||$ to the objective and find c or μ through a line search. In practice, however, we have observed that these are rarely needed as the KNIFE algorithm almost always strictly decreases the objective at each iteration for common kernels and loss functions. We note that we have opted to present the KNIFE approximation algorithm as the approach is faster than conducting a full line search as in the KNIFE descent algorithm. Our implementation of the KNIFE algorithm them employs the KNIFE approximation, checking the objective at each step to ensure it is decremented. If this is not the case, a line search as in the

KNIFE descent algorithm is employed.

For certain loss functions and kernels, however, we can obtain stronger convergence results for both the KNIFE algorithm and KNIFE descent algorithm.

Proposition 2. *If the KNIFE algorithm or KNIFE descent algorithm finds a unique minimum for the coefficients, $\boldsymbol{\alpha}$, and the weights, $\mathbf{w}$, in each step and the loss function and kernel are continuously differentiable, then the algorithm monotonically decreases the objective and converges to a stationary point of $f(\boldsymbol{\alpha}, \mathbf{w})$.*

Proof. Differentiability of the loss function and kernel implies that $f(\boldsymbol{\alpha}, \mathbf{w})$ is regular on its domain. This along with unique minima in both blocks of coordinates satisfies conditions for monotonic convergence to a stationary point for non-convex functions. Differentiability can be relaxed to weaker conditions for regularity (Tseng, 2001). □

The conditions of Proposition 2 require that the objective be strictly convex in both $\boldsymbol{\alpha}$ and $\mathbf{w}$. One such example is the squared error loss with linear kernel given in (3). We have mentioned that these examples are bi-convex, and thus the KNIFE algorithm simply iterates between minimization with respect to the coefficients and then the feature weights. As discussed in Section 3.3, these can also be written as a convex problem.

While we have discussed convergence properties of the KNIFE algorithm and descent algorithm, the solution will depend on the starting values of $\mathbf{w}$. Even if the conditions of Proposition 2 are satisfied, non-convex functions have potentially many stationary points. Thus, we recommend initializing the KNIFE algorithm at several random starting points and taking the solution which gives the minimum objective value. We investigate this as well as the convergence of the KNIFE approximation algorithm in a small numerical example in the left panel of Figure 5. Here, the KNIFE objective is shown for several iterations of the algorithm starting from random weight initializations. We see that the approximation strictly decreases the objective in this example.

Notice that both of the convergence results require the loss function to be continuously differentiable. While many loss functions such as squared error and binomial deviance are smooth, there is one notable exception, namely the hinge loss of support vector machines. Without smoothness conditions on the loss function, there may not be a feasible descent direction in Step 3 (b) of Algorithm 3 that decreases the original objective. Thus, the coordinate-wise minimizations of KNIFE for SVMs may never converge. Hence, we employ smooth approximations to the non-differentiable hinge loss such as squared error hinge and a Huberized hinge loss (Wang et al., 2008). These are shown in the right panel of Figure 5. Throughout this paper, KNIFE for SVMs is used with one of these smooth loss functions. Additionally, we note that in

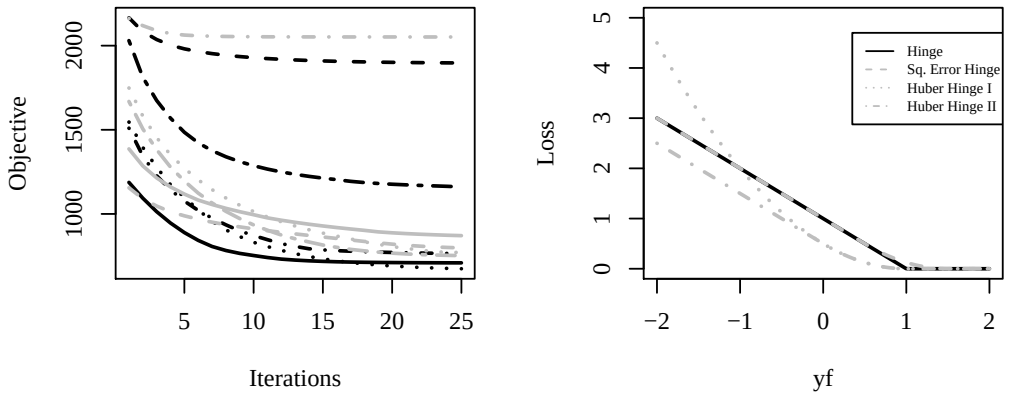

Figure 5: *(left) KNIFE objective for iterations of the KNIFE algorithm starting at 10 random sets of weights. Here, a radial kernel with squared error loss is used. (right) Smooth approximations to the non-differentiable hinge loss for the support vector machine.*

our experiments and examples, this approximation to the hinge loss generally decreases the original SVM objective function.