Support Vector Machines

Introduction

Fisher's linear discriminant function (LDF) and related classifiers for binary and multiclass learning problems have performed well for many years and for many data sets.

Recently, a brand-new learning methodology, **support vector machines (SVMs)**, has emerged which has matched the performance of the LDF and, in many instances, has proved to be superior to it.

Development and implementation of algorithms for SVMs are currently of great interest to theoretical researchers and applied scientists in machine learning, data mining, and bioinformatics.

Huge numbers of research articles, tutorials, and textbooks have been published on the topic, and annual workshops, new research journals, courses, and websites are now devoted to the subject.

SVMs have been successfully applied to classification problems as diverse as handwritten digit recognition, text categorization, cancer classification using micro-array expression data, protein secondary-structure prediction, and cloud classification using satellite-radiance profiles.

SVMs, which are available in both linear and nonlinear versions, involve optimization of a convex loss function under given constraints and so are unaffected by problems of local minima.

This gives SVMs quite a strong competitive advantage over methods such as neural networks and decision trees.

SVMs are computed using well-documented, general-purpose, mathematical programming algorithms, and their performance in many situations has been quite remarkable.

Even in the face of massive data sets, extremely fast and efficient software is being designed to compute SVMs for classification.

By means of the new technology of kernel methods, SVMs have been very successful in building highly nonlinear classifiers.

The kernel method enables us to construct linear classifiers in high-dimensional feature spaces that are nonlinearly related to input space and to carry out those computations in input space using very few parameters.

SVMs have also been successful in dealing with situations in which there are many more variables than observations.

Although these advantages hold in general, we have to recognize that there will always be applications in which SVMs can get beaten in performance by a hand-crafted classification method.

Linear Support Vector Machines

Assume we have available a learning set of data,

$$\mathcal{L} = \{(\mathbf{x}_i, y_i) : \; i = 1, 2, \ldots, n\}.$$

- $\mathbf{x}_i \in \mathcal{R}^r$;
- $y_i \in \{-1, +1\}$.

The **binary classification problem** is to use $\mathcal{L}$ to construct a function

$$f: \ \mathcal{R}^r \Longrightarrow \mathcal{R}$$

so that

$$C(\mathbf{x}) = \text{sign}\{f(\mathbf{x})\}$$

is a classifier.

The **separating function** f then classifies each new point $\mathbf{x}$ in a test set $\mathcal{T}$ into one of two classes, Π_+ or Π_-, depending on whether $C(\mathbf{x})$ is $+1$ (if $f(\mathbf{x}) > 0$) or -1 (if $f(\mathbf{x}) < 0$), respectively.

The goal is to have f assign all positive points in $\mathcal{T}$ (i.e., those with $y = +1$) to Π_+ and all negative points in $\mathcal{T}$ (i.e., those with $y = -1$) to Π_-.

Note

In practice, we recognize that 100% correct classification may not be possible.

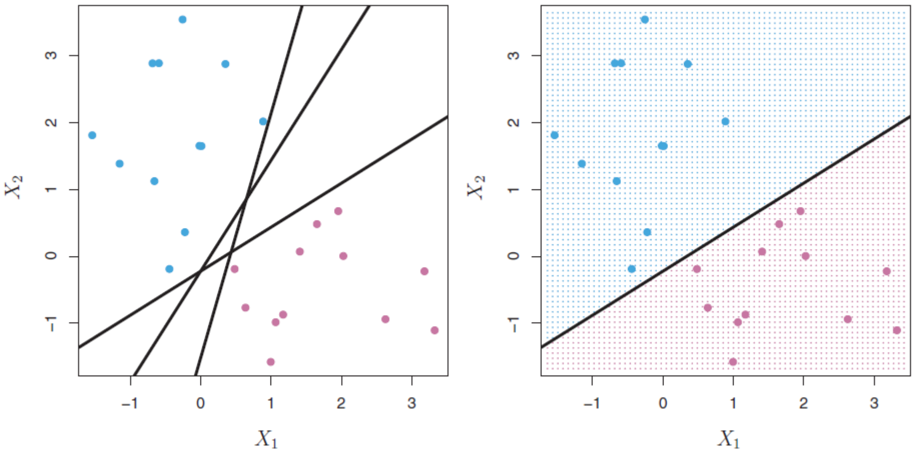

FIGURE: Hyperplanes with two variables (straight lines).

Note

- Left: There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.

- Right: A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.

The Linearly Separable Case

First, consider the simplest situation: suppose the positive ($y_i = +1$) and negative ($y_i = -1$) data points from the learning set $\mathcal{L}$ can be separated by a hyperplane,

$$\{\mathbf{x} : \ f(\mathbf{x}) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta} = 0\}.$$

- $\boldsymbol{\beta}$ is the **weight vector** with Euclidean norm $||\boldsymbol{\beta}||$,
- β_0 is the **bias**,
- $b = -\beta_0$ is the **threshold**.

If this hyperplane can separate the learning set into the two given classes without error, the hyperplane is termed a **separating hyperplane**.

Clearly, there is an infinite number of such separating hyperplanes.

Question

How do we determine which one is the best?

Consider any separating hyperplane.

Let d_- be the shortest distance from the separating hyperplane to the nearest negative data point, and let d_+ be the shortest distance from the same hyperplane to the nearest positive data point. Then, the **margin** of the separating hyperplane is defined as $d = d_- + d_+$.

If, in addition, the distance between the hyperplane and its closest observation is maximized, we say that the hyperplane is an **optimal separating hyperplane** (also known as a **maximal margin classifier**).

If the learning data from the two classes are linearly separable, there exists β_0 and $\boldsymbol{\beta}$ such that

$$\begin{cases} \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta} \geq +1, & \text{if } y_i = +1, \\ \\ \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta} \leq -1, & \text{if } y_i = -1. \end{cases}$$

- If there are data vectors in $\mathcal{L}$ such that equality holds in the first formula, then these data vectors lie on the hyperplane H_{+1}: $\beta_0 + \mathbf{x}^T\boldsymbol{\beta} = 1$.

- If there are data vectors in $\mathcal{L}$ such that equality holds in the second formula, then these data vectors lie on the hyperplane H_{-1}: $\beta_0 + \mathbf{x}^T\boldsymbol{\beta} = -1$.

Points in $\mathcal{L}$ that lie on either one of the hyperplanes H_{-1} or H_{+1}, are said to be **support vectors**.

The support vectors typically consist of a small percentage of the total number of sample points.

If $\mathbf{x}_{-1}$ lies on the hyperplane H_{-1}, and if $\mathbf{x}_{+1}$ lies on the hyperplane H_{+1}, then,

$$\beta_0 + \mathbf{x}_{-1}^T \boldsymbol{\beta} = -1, \quad \beta_0 + \mathbf{x}_{+1}^T \boldsymbol{\beta} = +1.$$

The difference of these two equations is $\mathbf{x}_{+1}^T \boldsymbol{\beta} - \mathbf{x}_{-1}^T \boldsymbol{\beta} = 2$, and it is easy to obtain

$$\beta_0 = -\frac{1}{2}\{\mathbf{x}_{+1}^T \boldsymbol{\beta} + \mathbf{x}_{-1}^T \boldsymbol{\beta}\}.$$

The perpendicular distances of the hyperplane $\beta_0 + \mathbf{x}^T\boldsymbol{\beta} = 0$ from the points $\mathbf{x}_{-1}$ and $\mathbf{x}_{+1}$ are

$$d_- = \frac{|\beta_0 + \mathbf{x}_{-1}^T\boldsymbol{\beta}|}{||\boldsymbol{\beta}||} = \frac{1}{||\boldsymbol{\beta}||}, \quad d_+ = \frac{|\beta_0 + \mathbf{x}_{+1}^T\boldsymbol{\beta}|}{||\boldsymbol{\beta}||} = \frac{1}{||\boldsymbol{\beta}||}.$$

So, the margin of the separating hyperplane $d = 2/||\boldsymbol{\beta}||$.

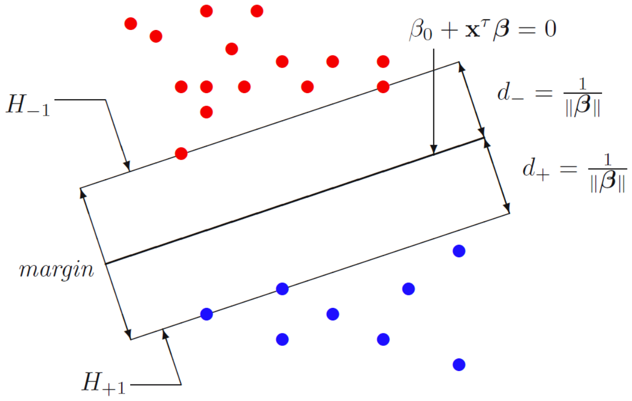
$\beta_0 + \mathbf{x}^T\boldsymbol{\beta} = 0$
H_{-1}
$d_- = \frac{1}{||\boldsymbol{\beta}||}$
$d_+ = \frac{1}{||\boldsymbol{\beta}||}$
margin
H_{+1}

FIGURE: Support vector machines: the linearly separable case. The red points correspond to data points with $y_i = -1$, and the blue points correspond to data points with $y_i = +1$. The separating hyperplane is the line $\beta_0 + \mathbf{x}^T\boldsymbol{\beta} = 0$. The support vectors are those points lying on the hyperplanes H_{-1} and H_{+1}. The margin of the separating hyperplane is $d = 2/||\boldsymbol{\beta}||$.

The above inequalities can be combined into a single set of

$$y_i(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}) \geq +1, \quad i = 1, 2, \ldots, n.$$

The quantity $y_i(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta})$ is called the **margin of $(\mathbf{x}_i, y_i)$ with respect to the hyperplane**, $i = 1, 2, \ldots, n$.

We see that $\mathbf{x}_i$ is a support vector with respect to the hyperplane if its margin equals one; that is, if

$$y_i(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}) = 1.$$

The support vector are those points lying on the hyperplanes H_{-1} and H_{+1}.

The empirical distribution of the margins of all the observations in $\mathcal{L}$ is called the **margin distribution of a hyperplane with respect to $\mathcal{L}$**.

The minimum of the empirical margin distribution is the **margin of the hyperplane with respect to $\mathcal{L}$**.

The problem is to find the optimal separating hyperplane; namely, find the hyperplane that maximizes the margin, $2/||\boldsymbol{\beta}||$, subject to the conditions.

Equivalently, we wish to find β_0 and $\boldsymbol{\beta}$ to

$$\text{minimize} \quad \frac{1}{2}||\boldsymbol{\beta}||^2,$$

$$\text{subject to} \quad y_i(\beta_0 + \mathbf{x}_i^T\boldsymbol{\beta}) \geq 1, \quad i = 1, 2, \ldots, n.$$

This is a convex optimization problem: minimize a quadratic function subject to linear inequality constraints.

Convexity ensures that we have a global minimum without local minima.

The resulting optimal separating hyperplane is called the **maximal (or hard) margin solution**.

The Linearly Nonseparable Case

In real applications, it is unlikely that there will be such a clear linear separation between data drawn from two classes. More likely, there will be some overlap.

We can generally expect some data from one class to infiltrate the region of space perceived to belong to the other class, and vice versa.

The overlap will cause problems for any classification rule, and, depending upon the extent of the overlap, we should expect that some of the overlapping points will be misclassified.

The **nonseparable case** occurs if either the two classes are separable, but not linearly so, or that no clear separability exists between the two classes, linearly or nonlinearly.

One reason for overlapping classes is the high noise level (i.e., large variances) of one or both classes.

As a result, one or more of the constraints will be violated.

The way we cope with overlapping data is to create a more flexible formulation of the problem, which leads to a **soft-margin solution**.

To do this, we introduce the concept of a nonnegative **slack variable**, ξ_i, for each observation, $(\mathbf{x}_i, y_i)$, in $\mathcal{L}$, $i = 1, 2, \ldots, n$.

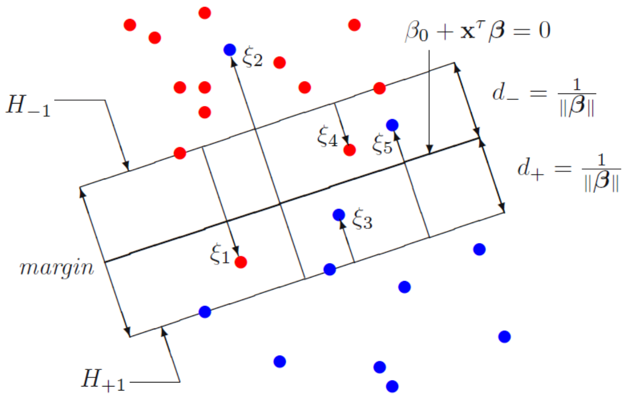
$\beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0$
ξ_2
H_{-1}
$d_- = \frac{1}{\|\boldsymbol{\beta}\|}$
ξ_4
ξ_5
$d_+ = \frac{1}{\|\boldsymbol{\beta}\|}$
ξ_3
ξ_1
margin
H_{+1}

FIGURE: Support vector machines: the linearly nonseparable case. The slack variables ξ_1 and ξ_4 are associated with the red points that violate the constraint of hyperplane H_{-1}, and points marked by ξ_2, ξ_3, and ξ_5 are associated with the blue points that violate the constraint of hyperplane H_{+1}. Points that satisfy the constraints of the appropriate hyperplane have $\xi_i = 0$.

Let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^T \geq \mathbf{0}$.

The constraints now become

$$y_i(\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}) + \xi_i \geq 1, \quad i = 1, 2, \ldots, n.$$

Note

Some data points may have $\xi_i = 0$.

The classifier now has to find the optimal hyperplane that controls both the margin, $2/||\boldsymbol{\beta}||$, and some computationally simple function of the slack variables, such as

$$g_\sigma(\boldsymbol{\xi}) = \sum_{i=1}^{n} \xi_i^\sigma,$$

subject to certain constraints.

Here, we discuss the case of $\sigma = 1$.

The **1-norm soft-margin optimization problem** is to find β_0, $\boldsymbol{\beta}$, and $\boldsymbol{\xi}$ to

$$\text{minimize} \quad \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n}\xi_i,$$

$$\text{subject to} \quad \xi_i \geq 0, \quad y_i(\beta_0 + \mathbf{x}_i^T\boldsymbol{\beta}) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, n.$$

- $C > 0$ is a **regularization parameter**. C takes the form of a tuning constant that controls the size of the slack variables and balances the two terms in the minimizing function.

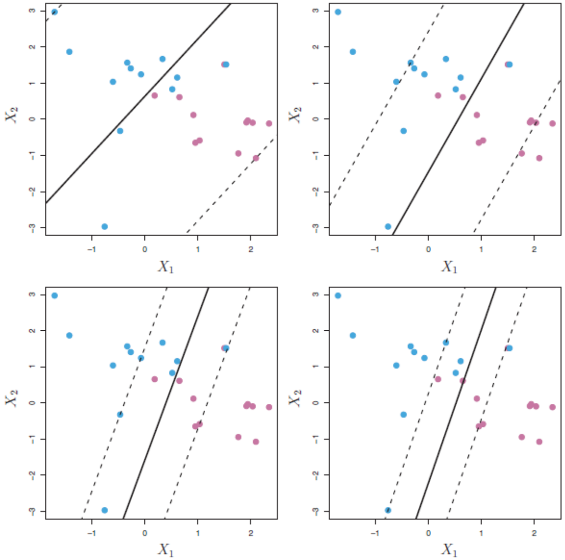

FIGURE: A support vector classifier was fit using four different values of the tuning parameter C.

The fact that only support vectors affect the classifier is in line with our previous assertion that C controls the bias-variance trade-off of the support vector classifier.

When the tuning parameter C is large, then the margin is wide, many observations violate the margin, and so there are many support vectors.

In this case, many observations are involved in determining the hyperplane.

The top left panel illustrates this setting: this classifier has low variance (since many observations are support vectors) but potentially high bias.

In contrast, if C is small, then there will be fewer support vectors and hence the resulting classifier will have low bias but high variance.

The bottom right panel in the illustrates this setting, with only eight support vectors.

Optimization Problem

See the textbook.

Nonlinear Boundary

The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear.

However, in practice we are sometimes faced with non-linear class boundaries.

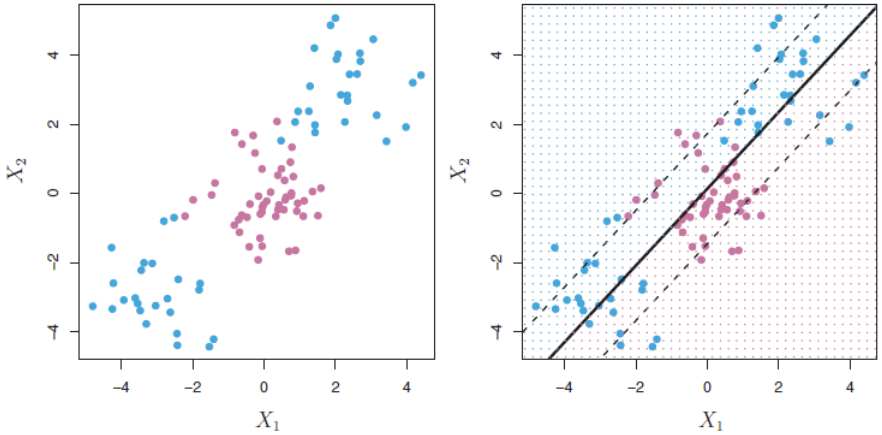

FIGURE: Left: The observations fall into two classes, with a non-linear boundary between them. Right: The support vector classifier seeks a linear boundary, and consequently performs very poorly.

For instance, consider the data in the left-hand panel of the figure.

It is clear that a support vector classifier or any linear classifier will perform poorly here.

Indeed, the support vector classifier shown in the right-hand panel of the figure is useless here.

Nonlinear Support Vector Machines

So far, we have discussed methods for constructing a linear SVM classifier.

But what if a linear classifier is not appropriate for the data set in question? Can we extend the idea of linear SVM to the nonlinear case?

The key to constructing a nonlinear SVM is to observe that the observations in $\mathcal{L}$ only enter the dual optimization problem through the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j,\ i, j = 1, 2, \ldots, n$.

Nonlinear Transformations

Suppose we transform each observation, $\mathbf{x}_i \in \mathcal{R}^r$, in $\mathcal{L}$ using some nonlinear mapping $\boldsymbol{\Phi}$: $\mathcal{R}^r \to \mathcal{H}$, where $\mathcal{H}$ is an $N_{\mathcal{H}}$-dimensional feature space.

The nonlinear map $\boldsymbol{\Phi}$ is generally called the **feature map** and the space $\mathcal{H}$ is called the **feature space**. The space $\mathcal{H}$ may be very high-dimensional, possibly even infinite dimensional.

We will generally assume that $\mathcal{H}$ is a Hilbert space of real-valued functions on $\mathcal{R}$ with inner product $\langle \cdot, \cdot \rangle$ and norm $|| \cdot ||$.

Let

$$\mathbf{\Phi}(\mathbf{x}_i) = \big(\phi_1(\mathbf{x}_i), \ldots, \phi_{N_{\mathcal{H}}}(\mathbf{x}_i)\big)^T \in \mathcal{H}, \quad i = 1, 2, \ldots, n.$$

The transformed sample is then $\{\mathbf{\Phi}(\mathbf{x}_i), y_i\}$, where $y_i \in \{-1, +1\}$ identifies the two classes.

If we substitute $\Phi(\mathbf{x}_i)$ for $\mathbf{x}_i$ in the development of the linear SVM, then data would only enter the optimization problem by way of the inner products $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$.

The difficulty in using nonlinear transformations in this way is computing such inner products in high-dimensional space $\mathcal{H}$.

The Kernel Trick

The idea behind nonlinear SVM is to find an optimal separating hyperplane (with or without slack variables, as appropriate) in high-dimensional feature space $\mathcal{H}$ just as we did for the linear SVM in input space.

Of course, we would expect the dimensionality of $\mathcal{H}$ to be a huge impediment to constructing an optimal separating hyperplane (and classification rule) because of the curse of dimensionality.

The fact that this does not become a problem in practice is due to the kernel trick.

The so-called kernel trick is a wonderful idea that is widely used in algorithms for computing inner products of the form $\langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle$ in feature space $\mathcal{H}$.

The trick is that instead of computing these inner products in $\mathcal{H}$, which would be computationally expensive because of its high dimensionality, we compute them using a nonlinear kernel function, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle$, in input space, which helps speed up the computations.

Then, we just compute a linear SVM, but where the computations are carried out in some other space.

Kernels and Their Properties

See the textbook.

Examples of Kernels

Kernel	$K\langle \mathbf{x}, \mathbf{y}\rangle$
Polynomial of degree d	$(\langle \mathbf{x}, \mathbf{y}\rangle + c)^d$
Gaussian radial basis function	$\exp\left\{-\frac{\lvert\lvert \mathbf{x}-\mathbf{y}\rvert\rvert^2}{2\sigma^2}\right\}$
Laplacian	$\exp\left\{-\frac{\lvert\lvert \mathbf{x}-\mathbf{y}\rvert\rvert}{\sigma}\right\}$
Thin-plate spline	$\left(\frac{\lvert\lvert \mathbf{x}-\mathbf{y}\rvert\rvert}{\sigma}\right)^2 \ln\left\{\frac{\lvert\lvert \mathbf{x}-\mathbf{y}\rvert\rvert}{\sigma}\right\}$
Sigmoid	$\tanh(a\langle \mathbf{x}, \mathbf{y}\rangle + b)$

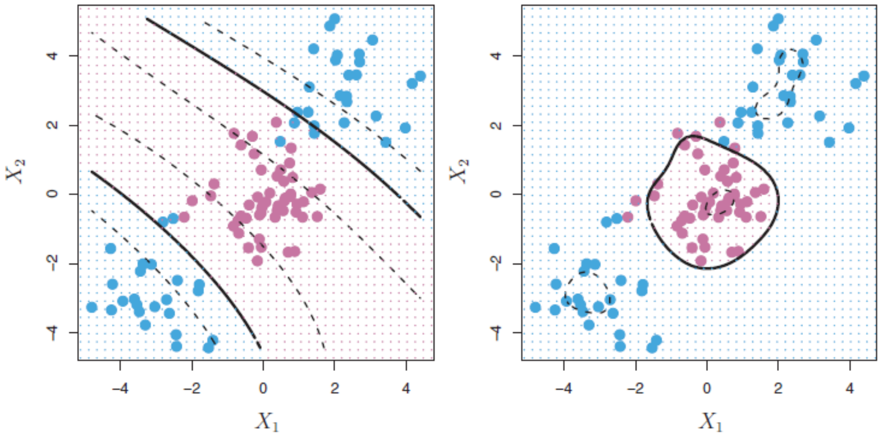

FIGURE: Left: An SVM with a polynomial kernel of degree 3 is applied to the non-linear data, resulting in a far more appropriate decision rule. Right: An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.

Optimizing in Feature Space

See the textbook.

Grid Search for Parameters

We need to determine two parameters when using a Gaussian RBF kernel, namely, the cost, C, of violating the constraints and the kernel parameter $\gamma = 1/\sigma^2$.

The parameter C in the box constraint can be chosen by searching a wide range of values of C using either CV (usually, 10-fold) on $\mathcal{L}$ or an independent validation set of observations.

In practice, it is usual to start the search by trying several different values of C, such as 10, 100, 1,000, 10,000, and so on.

A initial grid of values of γ can be selected by trying out a crude set of possible values, say, 0.00001, 0.0001, 0.001, 0.01, 0.1, and 1.0.

When there appears to be a minimum CV misclassification rate within an interval of the two-way grid, we make the grid search finer within that interval.

Armed with a two-way grid of values of (C, γ), we apply CV to estimate the generalization error for each cell in that grid.

The (C, γ) that has the smallest CV misclassification rate is selected as the solution to the SVM classification problem.

SVM as a Regularization Method

The idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, seemed distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis.

Moreover, the idea of using a kernel to expand the feature space in order to accommodate non-linear class boundaries appeared to be a unique and valuable characteristic.

However, since that time, deep connections between SVMs and other more classical statistical methods have emerged.

The SVM classifier can also be regarded as the solution to a particular regularization problem.

Let $f \in \mathcal{H}_K$, the reproducing kernel Hilbert space (rkhs) associated with the kernel K, with $||f||^2_{\mathcal{H}_K}$ the squared-norm of f in $\mathcal{H}_K$.

Classification error

Consider the classification error, $y_i - C(\mathbf{x}_i)$, where y_i and $C(\mathbf{x}_i) \in \{-1, +1\}$.

Then,

$$\left|y_i - C(\mathbf{x}_i)\right| = \left|y_i\left(1 - y_i C(\mathbf{x}_i)\right)\right| = \left|1 - y_i C(\mathbf{x}_i)\right| = \left(1 - y_i C(\mathbf{x}_i)\right)_+, \;\; i = 1, 2, \ldots, n.$$

Note

The classification error is not suitable being the loss function, however, it inspires us to consider an appropriate loss function.

Hinge Loss Function

The quantity $\left(1 - y_i f(\mathbf{x}_i)\right)_+$ is called the **hinge loss function** which could be zero if all $\mathbf{x}_i$ are correctly classified.

The hinge loss plays a vital role in SVM methodology; indeed, it has been shown to be Bayes consistent for classification in the sense that minimizing the loss function yields the Bayes rule.

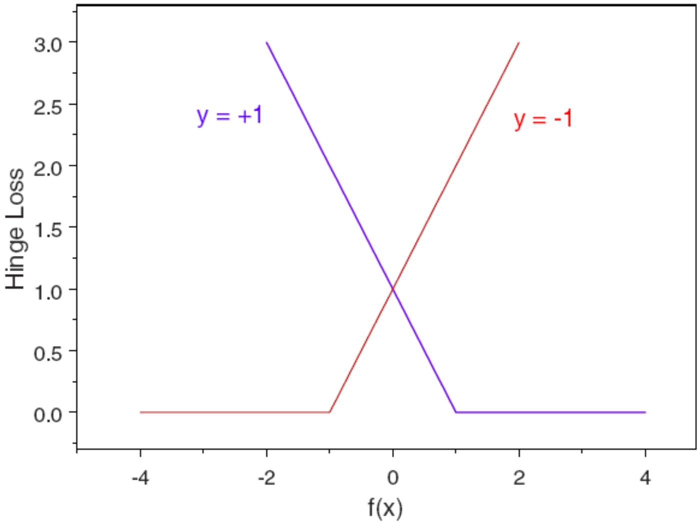

FIGURE: Hinge loss function $\left(1 - yf(x)\right)_+$ for $y = -1$ and $y = +1$.

The hinge loss is also related to the misclassification loss function

$$\mathrm{I}\big[y_i C(\mathbf{x}_i) \leq 0\big] = \mathrm{I}\big[y_i f(\mathbf{x}_i) \leq 0\big].$$

When $f(\mathbf{x}_i) = \pm 1$, the hinge loss is twice the misclassification loss; otherwise, the ratio of the two losses depends upon the sign of $y_i f(\mathbf{x}_i)$.

We wish to find a function $f \in \mathcal{H}_K$ to minimize a penalized version of the hinge loss.

Specifically, we wish to find $f \in \mathcal{H}_K$ to

$$\text{minimize} \;\; \frac{1}{n}\sum_{i=1}^{n}\big(1 - y_i f(\mathbf{x}_i)\big)_+ + \lambda ||f||^2_{\mathcal{H}_K}, \quad \lambda > 0.$$

In the above formula, the first term, $n^{-1}\sum_{i=1}^{n}\big(1 - y_i f(\mathbf{x}_i)\big)_+$, measures the distance of the data from separability, and the second term, $\lambda ||f||^2_{\mathcal{H}_K}$ penalizes over-fitting.

The tuning parameter λ balances the trade-off between estimating f (the first term) and how well f can be approximated (the second term).

After the minimizing f has been found, the SVM classifier is $C(\mathbf{x}) = \text{sign}\left\{ f(\mathbf{x}) \right\}$, $\mathbf{x} \in \mathcal{R}^r$.

If the space $\mathcal{H}_K$ consists of linear functions of the form

$$f(\mathbf{x}) = \beta_0 + \mathbf{\Phi}(\mathbf{x})^T \boldsymbol{\beta} \quad \text{with} \quad ||f||^2_{\mathcal{H}_K} = ||\boldsymbol{\beta}||^2,$$

then the problem of finding f is equivalent to one of finding β_0 and $\boldsymbol{\beta}$ to

$$\text{minimize} \;\; \frac{1}{n}\sum_{i=1}^{n}(1 - y_i f(\mathbf{x}_i))_+ + \lambda||\boldsymbol{\beta}||^2.$$

Then, the original problem can be reformulated in terms of solving the 1-norm soft-margin optimization problem.

Relationship to Logistic Regression

The loss function in SVM is known as hinge loss.

However, it hinge loss turns out that the hinge loss function is closely related to the loss function used in logistic regression.

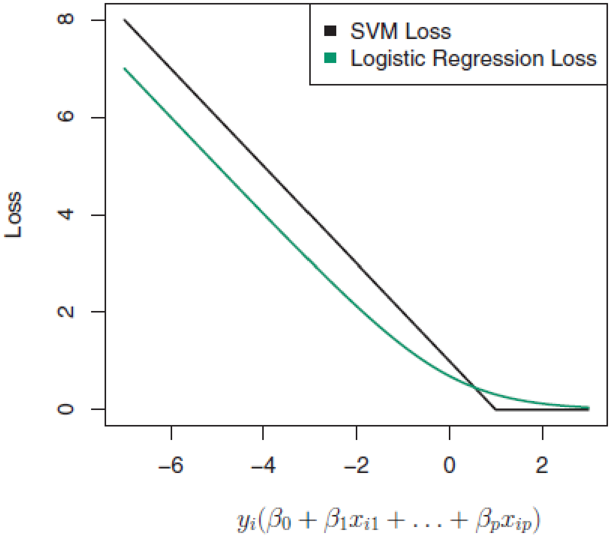

FIGURE: The SVM and logistic regression loss functions are compared, as a function of $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$.

An interesting characteristic of the support vector classifier is that only support vectors play a role in the classifier obtained; observations on the correct side of the margin do not affect it.

This is due to the fact that the loss function shown in the figure is exactly zero for observations for which $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq 1$; these correspond to observations that are on the correct side of the margin.

In contrast, the loss function for logistic regression shown in the figure is not exactly zero anywhere.

But it is very small for observations that are far from the decision boundary.

Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results.

When the classes are well separated, SVMs tend to behave better than logistic regression; in more overlapping regimes, logistic regression is often preferred.

The Role of C

When the support vector classifier and SVM were first introduced, it was thought that the tuning parameter C was an unimportant nuisance parameter that could be set to some default value, like 1.

However, the Loss+Penalty formulation for the support vector classifier indicates that this is not the case.

The choice of tuning parameter is very important and determines the extent to which the model underfits or overfits the data.

We have established that the support vector classifier is closely related to logistic regression and other preexisting statistical methods.

Is the SVM unique in its use of kernels to enlarge the feature space to accommodate non-linear class boundaries?

The answer to this question is no.

We could just as well perform logistic regression or many of the other classification methods using non-linear kernels; this is closely related to some of the non-linear approaches.

However, for historical reasons, the use of non-linear kernels is much more widespread in the context of SVMs than in the context of logistic regression or other methods.

Multiclass Support Vector Machines

Often, data are derived from more than two classes.

In the multiclass situation, $\mathbf{X} \in \mathcal{R}^r$ is a random r-vector chosen for classification purposes and $Y \in \{1, 2, \ldots, K\}$ is a class label, where K is the number of classes.

Because SVM classifiers are formulated for only two classes, we need to know if (and how) the SVM methodology can be extended to distinguish between $K > 2$ classes.

There have been several attempts to define such a multiclass SVM strategy.

Multiclass SVM as a Series of Binary Problems

The standard SVM strategy for a multiclass classification problem (over K classes) has been to reduce it to a series of binary problems.

There are different approaches to this strategy.

One-versus-rest

Divide the K-class problem into K binary classification subproblems of the type kth class vs. not kth class, $k = 1, 2, \ldots, K$.

Corresponding to the kth subproblem, a classifier $\hat{f}_k$ is constructed in which the kth class is coded as positive and the union of the other classes is coded as negative.

A new $\mathbf{x}$ is then assigned to the class with the largest value of $\hat{f}_k(\mathbf{x})$, $k = 1, 2, \ldots, K$, where $\hat{f}_k(\mathbf{x})$ is the optimal SVM solution for the binary problem of the kth class versus the rest.

One-versus-one

Divide the K-class problem into $\binom{K}{2}$ comparisons of all pairs of classes.

A classifier $\hat{f}_{jk}$ is constructed by coding the jth class as positive and the kth class as negative, $j, k = 1, 2, \ldots, K$, $j \neq k$.

Then, for a new $\mathbf{x}$, aggregate the votes for each class and assign $\mathbf{x}$ to the class having the most votes.

Even though these strategies are widely used in practice to resolve multiclass SVM classification problems, one has to be cautious about their use.

The one-versus-rest approach is popular for carrying out text categorization tasks, where each document may belong to more than one class.

Although it enjoys the optimality property of the SVM method for each binary subproblem, it can yield a different classifier than the Bayes optimal classifier for the multiclass case.

Furthermore, the classification success of the one-versus-rest approach depends upon the extent of the class-size imbalance of each subproblem and whether one class dominates all other classes when determining the most-probable class for each new $\mathbf{x}$.

The one-versus-one approach, which uses only those observations belonging to the classes involved in each pairwise comparison, suffers from the problem of having to use smaller samples to train each classifier, which may, in turn, increase the variance of the solution.

A True Multiclass SVM

To construct a true multiclass SVM classifier, we need to consider all K classes, $\Pi_1, \Pi_2, \ldots, \Pi_K$, simultaneously.

Note

The classifier has to reduce to the binary SVM classifier if $K = 2$.

Let $\mathbf{v}_1, \ldots, \mathbf{v}_K$ be a sequence of K-vectors, where $\mathbf{v}_k$ has a 1 in the kth position and whose elements sum to zero, $k = 1, 2, \ldots, K$; that is, let

$$\mathbf{v}_1 = \left(1, -\frac{1}{K-1}, \ldots, -\frac{1}{K-1}\right)^T,$$

$$\mathbf{v}_2 = \left(-\frac{1}{K-1}, 1, \ldots, -\frac{1}{K-1}\right)^T,$$

$$\vdots$$

$$\mathbf{v}_K = \left(-\frac{1}{K-1}, -\frac{1}{K-1}, \ldots, 1\right)^T.$$

Note that if $K = 2$, then $\mathbf{v}_1 = (1, -1)^T$ and $\mathbf{v}_2 = (-1, 1)^T$.

Every $\mathbf{x}_i$ can be labeled as one of these K vectors; that is, $\mathbf{x}_i$ has label $\mathbf{y}_i = \mathbf{v}_k$ if $\mathbf{x}_i \in \Pi_k,\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, K$.

Next, we generalize the separating function $f(\mathbf{x})$ to a K-vector of separating functions,

$$\mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})\big)^T,$$

$$f_k(\mathbf{x}) = \beta_{0k} + h_k(\mathbf{x}), \quad h_k \in \mathcal{H}_K, \quad k = 1, 2, \ldots, K.$$

$\mathcal{H}_K$ is a reproducing-kernel Hilbert space (rkhs) spanned by the $\{K(\mathbf{x}_i, \cdot),\ i = 1, 2, \dots, n\}$.

For example, in the linear case, $h_k(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\beta}_k$, for some vector of coefficients $\boldsymbol{\beta}_k$.

We also assume, for uniqueness, that

$$\sum_{k=1}^{K} f_k(\mathbf{x}) = 0.$$

Let $\mathbf{L}(\mathbf{y}_i)$ be a K-vector with 0 in the kth position if $\mathbf{x}_i \in \Pi_k$, and 1 in all other positions; this vector represents the equal costs of misclassifying $\mathbf{x}_i$ (and allows for an unequal misclassification cost structure if appropriate).

If $K = 2$ and $\mathbf{x}_i \in \Pi_1$, then $\mathbf{L}(\mathbf{y}_i) = (0, 1)^T$, while if $\mathbf{x}_i \in \Pi_2$, then $\mathbf{L}(\mathbf{y}_i) = (1, 0)^T$.

The multiclass generalization of the optimization problem is, therefore, to find functions $\mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})\big)^T$ satisfying the uniqueness which

$$\text{minimize} \;\; \mathcal{I}_\lambda(\mathbf{f}, \mathcal{Y}) = \frac{1}{n}\sum_{i=1}^{n}[\mathbf{L}(\mathbf{y}_i)]^T(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ + \frac{\lambda}{2}\sum_{k=1}^{K}||h_k||^2.$$

- $(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ = ((f_1(\mathbf{x}_i) - y_{i1})_+, \ldots, (f_K(\mathbf{x}_i) - y_{iK})_+)^T$,
- $\mathcal{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ is a $(K \times n)$-matrix.

By setting $K = 2$.

If $\mathbf{x}_i \in \Pi_1$, then $\mathbf{y}_i = \mathbf{v}_1 = (1, -1)^T$, and

$$
\begin{aligned}
[\mathbf{L}(\mathbf{y}_i)]^T(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ &= (0, 1)\big((f_1(\mathbf{x}_i) - 1)_+, (f_2(\mathbf{x}_i) + 1)_+\big)^T, \\
&= (f_2(\mathbf{x}_i) + 1)_+ = (1 - f_1(\mathbf{x}_i))_+. \quad f_1(\mathbf{x}_i) + f_2(\mathbf{x}_i) = 0
\end{aligned}
$$

If $\mathbf{x}_i \in \Pi_2$, then $\mathbf{y}_i = \mathbf{v}_1 = (-1, 1)^T$, and

$$[\mathbf{L}(\mathbf{y}_i)]^T(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ = (1, 0)\big((f_1(\mathbf{x}_i) + 1)_+, (f_2(\mathbf{x}_i) - 1)_+\big)^T,$$

$$= \big(f_1(\mathbf{x}_i) + 1\big)_+.$$

Moreover, if $K = 2$,

$$\sum_{k=1}^{2} ||h_k||^2 = ||h_1||^2 + || - h_1||^2 = 2||h_1||^2.$$

Support Vector Regression

The SVM was designed for classification.

Questions

- Can we extend (or generalize) the idea to regression?
- How would the main concepts used in SVM - convex optimization, optimal separating hyperplane, support vectors, margin, sparseness of the solution, slack variables, and the use of kernels - translate to the regression situation?

It turns out that all of these concepts find their analogues in regression analysis and they add a different view to the topic than the views we saw in classical regression analysis.

ϵ-Insensitive Loss Functions

In SVM classification, the margin is used to determine the amount of separation between two nonoverlapping classes of points: the bigger the margin, the more confident we are that the optimal separating hyperplane is a superior classifier.

In regression, we are not interested in separating points but in providing a function of the input vectors that would track the points closely.

Thus, a regression analogue for the margin would entail forming a band or tube around the true regression function that contains most of the points.

Points not contained within the tube would be described through slack variables.

In formulating these ideas, we first need to define an appropriate loss function.

We define a loss function that ignores errors associated with points falling within a certain distance (e.g., $\epsilon > 0$) of the true linear regression function,

$$\mu(\mathbf{x}) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta}.$$

In other words, if the point $(\mathbf{x}, y)$ is such that $|y - \mu(\mathbf{x})| \leq \epsilon$, then the loss is taken to be zero; if, on the other hand, $|y - \mu(\mathbf{x})| > \epsilon$, then we take the loss to be $|y - \mu(\mathbf{x})| - \epsilon$.

Loss Functions

With this strategy in mind, we can define the following two types of loss function:

- $L_1^{\epsilon}\big(y, \mu(\mathbf{x})\big) = \max\left\{0, |y - \mu(\mathbf{x})| - \epsilon\right\}$,
- $L_2^{\epsilon}\big(y, \mu(\mathbf{x})\big) = \max\left\{0, \big(y - \mu(\mathbf{x})\big)^2 - \epsilon\right\}$.

The first loss function, L_1^ϵ, is called the **linear ϵ-insensitive loss function**, and the second, L_2^ϵ, is the **quadratic ϵ-insensitive loss function**.

We see that the linear loss function ignores all errors falling within $\pm\epsilon$ of the true regression function $\mu(\mathbf{x})$ while dampening in a linear fashion errors that fall outside those limits.

Optimization for Linear ϵ-Insensitive Loss

See the textbook.

Optimization Algorithms for SVMs

When a data set is small, general-purpose linear programming (LP) or quadratic programming (QP) optimizers work quite well to solve SVM problems; QP optimizers can solve problems having about a thousand points, whereas LP optimizers can deal with hundreds of thousands of points.

With large data sets, however, a more sophisticated approach is required.

The main problem when computing SVMs for very large data sets is that storing the entire kernel in main memory dramatically slows down computation.

Alternative algorithms, constructed for the specific task of overcoming such computational inefficiencies, are now available in certain SVM software.