Machine Learning for OR & FE

Support Vector Machines (and the Kernel Trick)

Martin Haugh

Department of Industrial Engineering and Operations Research
Columbia University
Email: `martin.b.haugh@gmail.com`

Additional References: Christopher Bishop's *PRML*

Outline

Introduction to Support Vector Machines

Support vector machines are **non-probabilistic** binary linear classifiers.

The use of basis functions and the kernel trick mitigates the constraint of the SVM being a linear classifier

- in fact SVMs are particularly associated with the kernel trick.

Only a subset of data-points are required to define the SVM classifier

- these points are called support vectors.

SVMs are very popular classifiers and applications include

- text classification
- outlier detection
- face detection
- database marketing
- and many others.

SVMs are also used for multi-class classification.

Also have support vector regression.

The Separable Case

There are two classes which are assumed (for now) to be linearly separable.

Training data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with corresponding targets, $t_1, \ldots, t_n$ with $t_i \in \{-1, 1\}$.

We consider a classification rule of the form of the form

$$\begin{aligned} h(\mathbf{x}) &= \text{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right) \\ &= \text{sign}\left(y(\mathbf{x})\right) \end{aligned}$$

where $y(\mathbf{x}) := \mathbf{w}^\top \mathbf{x} + b$.

Note we can re-scale $(\mathbf{w}, b)$ without changing the decision boundary.

Therefore choose $(\mathbf{w}, b)$ so that training points closest to boundary satisfy $y(\mathbf{x}) = \pm 1$

- see left-hand component of Figure 7.1 from Bishop.

Let $\mathbf{x}_1$ be closest point from class with $t_1 = -1$ so that $\mathbf{w}^\top \mathbf{x}_1 + b = -1$.

And let $\mathbf{x}_2$ be closest point from class with $t_2 = 1$ so that $\mathbf{w}^\top \mathbf{x}_2 + b = 1$.

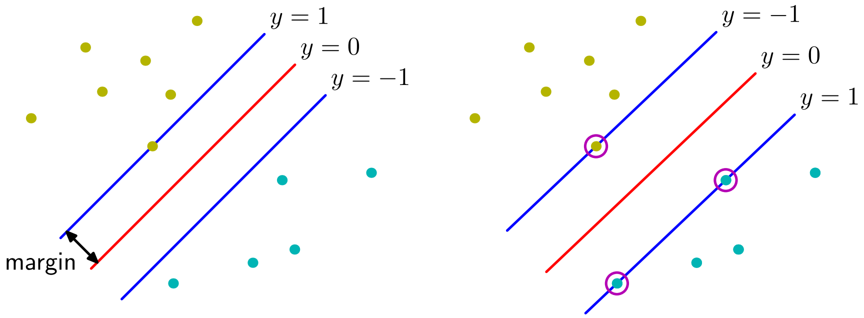


Figure 7.1 from Bishop: The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

Geometry of Maximizing the Margin

Recall the perpendicular distance of a point $\mathbf{x}$ from the hyperplane, $\mathbf{w}^\top \mathbf{x} + b = 0$, is given by

$$|\mathbf{w}^\top \mathbf{x} + b| / ||\mathbf{w}||.$$

Therefore distance of closest points in each class to the classifier is $1/||\mathbf{w}||$.

An SVM seeks the maximum margin classifier that separates all the data

- seems like a good idea
- but can also be justified by statistical learning theory.

Maximizing the margin, $1/||\mathbf{w}||$, is equivalent to minimizing $f(\mathbf{w}) := \frac{1}{2}\mathbf{w}^\top \mathbf{w}$.

Therefore obtain the following primal problem for the separable case:

$$\min_{\mathbf{w},b} \quad f(\mathbf{w}) = \tfrac{1}{2}\mathbf{w}^\top \mathbf{w} \tag{1}$$

$$\text{subject to} \quad t_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1, \quad i = 1, \ldots, n \tag{2}$$

Note that (2) ensures that all the training points are correctly classified.

The Primal Problem

The primal problem is a quadratic program with linear inequality constraints

- moreover it is convex and therefore has a unique minimum.

From the problem's geometry should be clear that only the points closest to the boundary are required to define the optimal hyperplane

- see right-hand component of Figure 7.1 from Bishop.
- these are called the support vectors
- and will see that the solution can be expressed using only these points.

Could stop here but will go to the corresponding dual problem to fully understand SVMs and the kernel trick.

The Dual Problem in Separable Case

We use a Lagrange multiplier $\alpha_i \geq 0$ for each constraint in (2). Lagrangian then given by

$$L(\mathbf{w}, b; \alpha) \;=\; \frac{1}{2}\mathbf{w}^\top\mathbf{w} \;+\; \sum_{i=1}^{n} \alpha_i \left(1 - t_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right)\right) \tag{3}$$

We now wish to solve for $g(\boldsymbol{\alpha}) := \min_{\mathbf{w},b} L(\mathbf{w}, b; \boldsymbol{\alpha})$.

Note that (why?) $g(\boldsymbol{\alpha}) \leq f(\mathbf{w}^*)$ where $(\mathbf{w}^*, b^*)$ is the optimal solution to the primal problem.

Can therefore formulate the dual problem:

$$\max_{\boldsymbol{\alpha} \geq 0} g(\boldsymbol{\alpha}) \tag{4}$$

Since primal problem is convex it follows that minimum of primal equals maximum of dual problem

- i.e. $g(\boldsymbol{\alpha}^*) = f(\mathbf{w}^*)$ where $\boldsymbol{\alpha}^*$ is the optimal solution to the dual problem
- this is strong duality.

The Dual Problem in Separable Case

To solve (4) first need to solve for $g(\boldsymbol{\alpha})$: the first order conditions (FOC) are

$$\begin{aligned} \frac{\partial L}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{n} \alpha_i t_i = 0 \\ \frac{\partial L}{\partial \mathbf{w}} &= 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{x}_i \end{aligned} \tag{5}$$

The FOC are necessary and sufficient (why?) for optimality. Can substitute them into (3) to obtain

$$\begin{aligned} L(\mathbf{w}, b; \alpha) &= \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - t_i \left(\sum_{j=1}^{n} \alpha_j t_j \mathbf{x}_j^\top \mathbf{x}_i + b \right) \right) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\top \mathbf{x}_j \end{aligned}$$

The Dual Problem in Separable Case

Then dual problem in the separable case reduces to

$$\max_{\boldsymbol{\alpha} \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\top \mathbf{x}_j \tag{6}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i t_i = 0 \tag{7}$$

- also a convex quadratic program, but now with a single linear constraint.

The complementary slackness conditions imply that only the support vectors will have non-zero α's in the optimum solution.

Let $\boldsymbol{\alpha}^*$ be the optimal solution to the dual problem. Then (5) yields

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* t_i \mathbf{x}_i$$

and we obtain b^* by noting that $t_i \left(\mathbf{w}^{*\top} \mathbf{x}_i + b^* \right) = 1$ for any i with $\alpha_i^* > 0$

- so find such an i and then solve for b^*.

The Kernel Trick

The kernel-trick is a very commonly used technique in regression, classification, PCA etc. that allows the problem to be easily embedded in much higher dimensional spaces and often infinite dimensional spaces

- but without having to do an infinite amount of work.

Suppose instead of using $\mathbf{x} \in \mathbb{R}^m$ to describe the inputs we instead use a feature map, $\boldsymbol{\phi}(\mathbf{x})^\top \in \mathbb{R}^M$, often with $M >> m$.

Then if the data is linearly separable in $\mathbb{R}^M$ can solve the same dual problem as (6) and (7) except we replace $\mathbf{x}_i^\top \mathbf{x}_j$ with $\boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j)$.

Can obtain corresponding optimal b^* via

$$b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j) \quad \text{for any } \alpha_j^* > 0 \tag{8}$$

and, for a new data-point $\mathbf{x}$, the prediction

$$\text{sign}\left(\mathbf{w}^{*\top} \boldsymbol{\phi}(\mathbf{x}) + b^*\right) \;=\; \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* t_i \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}) + b^*\right). \tag{9}$$

A Detour on Kernels

Define the Gram matrix, $\mathbf{K} = \boldsymbol{\phi}\boldsymbol{\phi}^\top$ to be the $n \times n$ matrix with

$$\mathbf{K}_{ij} := \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j) =: k(\mathbf{x}_i, \mathbf{x}_j) \tag{10}$$

For any set of points, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the kernel matrix $\mathbf{K}$ is positive semi-definite so that $\mathbf{z}^\top \mathbf{K} \mathbf{z} \geq 0$ for all $\mathbf{z} \in \mathbb{R}^n$.

Definition. We say a function, $k(\mathbf{x}, \mathbf{x}')$, is a kernel if it corresponds to a scalar product, $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ in some feature space, $\mathbb{R}^M$, possibly with $M = \infty$.

Mercer's Theorem. A necessary and sufficient condition for a function, $k(\mathbf{x}, \mathbf{x}')$, to be a kernel is that the corresponding Gram matrix, $\mathbf{K}$, be positive semi-definite for all possible choices of $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

A Detour on Kernels

Key implication of theorem is possibility of implicitly defining a (possibly infinite-dimensional) feature map, $\phi(\cdot)$, using a kernel function $k(\cdot, \cdot)$.

Note that $\phi(\cdot)$ is not explicitly required to state the dual problem, nor is it required in (8) and (9)

- only $k(\cdot, \cdot)$ is required!
- a big advantage since far less work may be required to compute $k(\cdot, \cdot)$.

e.g. Let $m = 2$ and define $k(\mathbf{x}, \mathbf{x}') := (\mathbf{x}^\top \mathbf{x}')^2$. Easy to check that $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ where

$$\boldsymbol{\phi}(\mathbf{x}) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right).$$

But calculating $k(\mathbf{x}, \mathbf{x}')$ requires $O(m)$ ($= \dim(\mathbf{x})$) work whereas calculating $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ requires $O(M) = O(m^2)$ work.

More generally, we could define $k(\mathbf{x}, \mathbf{x}') := (\mathbf{x}^\top \mathbf{x}' + c)^p$

- computing it will still be $O(m)$ but working with corresponding feature mapping will be $O(m^p)$.

Constructing New Kernels (Bishop)

We assume:

- $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$ are valid kernels.
- $c > 0$ is a constant.
- $f(\cdot)$ is any function.
- $q(\cdot)$ is a polynomial with nonnegative coefficients.
- $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$.
- $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$.
- $\mathbf{A}$ is a symmetric positive semi-definite matrix.
- $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$.
- k_a and k_b are valid kernel functions over their respective spaces.

Constructing New Kernels (Bishop)

Then the following are all valid kernels:

$$
\begin{aligned}
k(x,x') &= ck_1(\mathbf{x},\mathbf{x}') \\
k(x,x') &= f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}') && (11) \\
k(x,x') &= q(k_1(\mathbf{x},\mathbf{x}')) \\
k(x,x') &= \exp(k_1(\mathbf{x},\mathbf{x}')) && (12) \\
k(x,x') &= k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}') \\
k(x,x') &= k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}') \\
k(x,x') &= k_3(\phi(\mathbf{x}),\phi(\mathbf{x}')) \\
k(x,x') &= \mathbf{x}^\top \mathbf{A}\mathbf{x}' \\
k(x,x') &= k_a(\mathbf{x}_a,\mathbf{x}'_a) + k_b(\mathbf{x}_b,\mathbf{x}'_b) \\
k(x,x') &= k_a(\mathbf{x}_a,\mathbf{x}'_a)k_b(\mathbf{x}_b,\mathbf{x}'_b)
\end{aligned}
$$

The Gaussian kernel

The Gaussian kernel is given by:

$$k(\mathbf{x}, \mathbf{x}') \;=\; \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) \tag{13}$$

It is a valid kernel because

$$\begin{aligned}
\exp\left(\frac{-||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) &= \exp\left(\frac{-\mathbf{x}^\top\mathbf{x}}{2\sigma^2}\right)\exp\left(\frac{\mathbf{x}^\top\mathbf{x}'}{\sigma^2}\right)\exp\left(\frac{-\mathbf{x}'^\top\mathbf{x}'}{2\sigma^2}\right) \\
&= f(\mathbf{x})\exp\left(\frac{\mathbf{x}^\top\mathbf{x}'}{\sigma^2}\right)f(\mathbf{x}')
\end{aligned}$$

and now we can apply (11) and (12).

Constructing Kernels for Other Objects

The kernel trick can be extended to inputs that are symbolic and not just vectors of real numbers.

Examples of such inputs are graphs, sets, strings, and text documents.

e.g. Consider a fixed set and define the space consisting of all possible subsets of this set. If A_1 and A_2 are two such subsets then let

$$k(A_1, A_2) := 2^{|A_1 \cap A_2|}$$

where $|A|$ denotes the number of subsets in A.

$k(\cdot, \cdot)$ is a valid kernel because it can be shown to correspond to an inner product in a feature space

- so we could easily use SVMs to classify these sets.

The Kernel-Separated Dual

Returning to SVMs, when the data is kernel-separated our dual problem becomes:

$$\begin{array}{rl} \max_{\alpha \geq \mathbf{0}} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^{n} \alpha_i t_i = 0. \end{array}$$

Given a solution $\boldsymbol{\alpha}^*$ to the dual, can obtain corresponding optimal b^* via

$$b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for any } \alpha_j^* > 0$$

and, for a new data-point $\mathbf{x}$, the prediction

$$\text{sign}\left({\mathbf{w}^*}^\top \boldsymbol{\phi}(\mathbf{x}) + b^* \right) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* t_i k(\mathbf{x}_i, \mathbf{x}) + b^* \right).$$

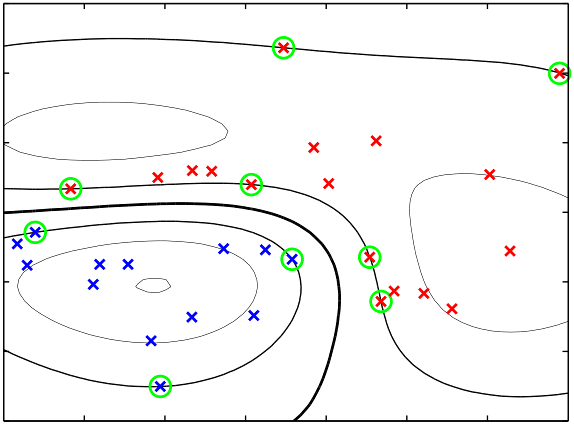
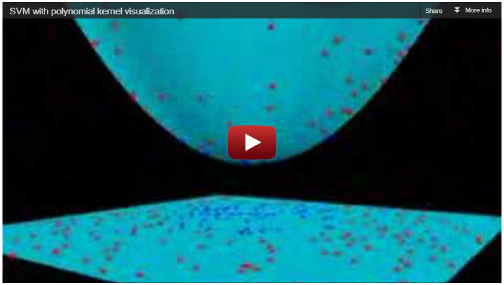


Figure 7.2 from Bishop: Example of synthetic data from two classes in two dimensions showing contours of constant $y(x)$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.

- Note that the data is linearly separable in the Gaussian-kernel space but not in the original space.



A Demo of SVM Classification with Polynomial Kernel by Udi Aharoni

The Non-Separable Case

In general the data will be non-separable so the primal problem of (1) and (2) will be infeasible.

Several ways to proceed: **e.g.** minimize the number of misclassified points, but this is NP-hard.

Instead we allow points to violate the margin constraints and penalize accordingly in the objective function. This yields the more general non-separable primal problem:

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \ \frac{1}{2}\mathbf{w}^\top \mathbf{w} \ + \ C\sum_{i=1}^{n} \xi_i \tag{14}$$

$$\begin{aligned} \text{subject to} \quad t_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) &\geq 1 - \xi_i, \quad i = 1, \ldots, n \\ \xi_i &\geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{15}$$

- again a convex quadratic programming problem with linear constraints
- the penalty C usually chosen by cross-validation.

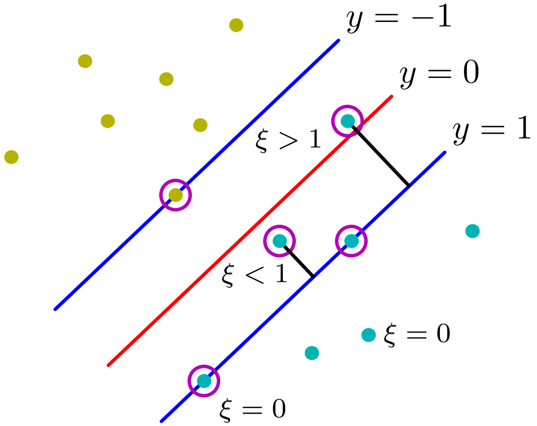


Figure 7.3 from Bishop: Illustration of the slack variables in $\xi_n \geq 0$. Data points with circles around them are support vectors.

Note that the slack variables allow points to be misclassified.

The Non-Separable Dual Problem

As with the separable case, it's more convenient to work with the dual.

Because the primal problem is convex the dual and primal have equal optimal objective functions.

The non-separable dual problem reduces to

$$\begin{aligned} \max_{\boldsymbol{\alpha}\geq 0,\, \boldsymbol{\lambda}\geq 0} \quad & \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{n} \alpha_i t_i = 0 \\ & C - \alpha_i - \lambda_i = 0, \qquad i = 1, \ldots, n \end{aligned} \tag{16}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ are Lagrange multipliers for the constraints (15)

- again a convex quadratic program with linear constraints
- the original dual plus the additional linear constraints of (16).

Can remove $\boldsymbol{\lambda}$ from the dual by replacing (16) with $\alpha_i \leq C$ for $i = 1, \ldots, n$.

Kernelizing the Dual

As with the separable case, we can easily apply the kernel trick to obtain the following general non-separable dual problem:

$$\max_{\boldsymbol{\alpha} \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{17}$$

subject to

$$\sum_{i=1}^{n} \alpha_i t_i = 0 \tag{18}$$

$$\alpha_i \leq C, \quad i = 1, \ldots, n \tag{19}$$

Given an optimal solution, $\boldsymbol{\alpha}^*$, can recover the SVM classifier as:

$$b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for any } C > \alpha_j^* > 0$$

and, for a new data-point $\mathbf{x}$, the prediction

$$\text{sign}\left(\mathbf{w}^{*\top} \boldsymbol{\phi}(\mathbf{x}) + b^*\right) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* t_i k(\mathbf{x}_i, \mathbf{x}) + b^*\right). \tag{20}$$

Numerical Optimization of the SVM Problems

Since both primal and dual are convex quadratic problems we could (in theory) solve either one using convex optimization techniques.

Traditionally we preferred to solve the dual because constraints in the dual were easier to handle.

But standard convex optimization techniques not suitable for the dual problem because the "Q" matrix in (17) is often too large

- training sets on the order of $20,000$ points not uncommon
- standard gradient methods would require storing a $20k \times 20k$ matrix!

So special purpose solvers were used instead. **e.g.** the sequential minimization optimization (SMO) algorithm

- which avoids the need for storing the entire Q matrix.

Remark: The kernel trick not the motivation for working with the dual since can also implement kernels directly on the primal problem. In fact for very large-scale problems the primal problem is now the preferred formulation and it is solved via first-order methods – see later slides.

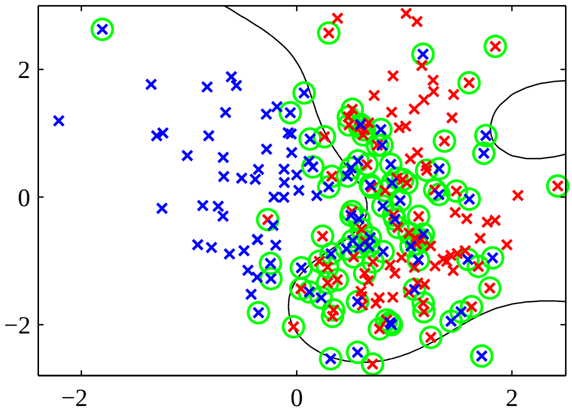


Figure 7.4 from Bishop: Illustration of the ν-SVM applied to a nonseparable data set in two dimensions. The support vectors are indicated by circles.

Comparing SVM and Logistic Loss Functions

Interesting to compare the error functions used by various classifiers.

The primal objective function of the SVM classifier may be written (why?) as

$$\begin{aligned}\text{Obj. Fun.} &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ &\equiv \frac{1}{2C}||\mathbf{w}||^2 + \sum_{i=1}^{n} E_{sv}(t_i y_i)\end{aligned}$$

where

$$E_{sv}(t_i y_i) := [1 - t_i y_i]^+ \tag{21}$$

and where $y_i := y(\mathbf{x}_i)$.

The error function $E_{sv}(\cdot)$ is known as the hinge error function.

It is this error function that induces the sparsity, i.e. $\alpha_i^* = 0$ for many i's, in the optimal SVM dual.

It is an approximation to the $0-1$ loss function and is compared with the loss function used by logistic regression in Figure 7.5 from Bishop.

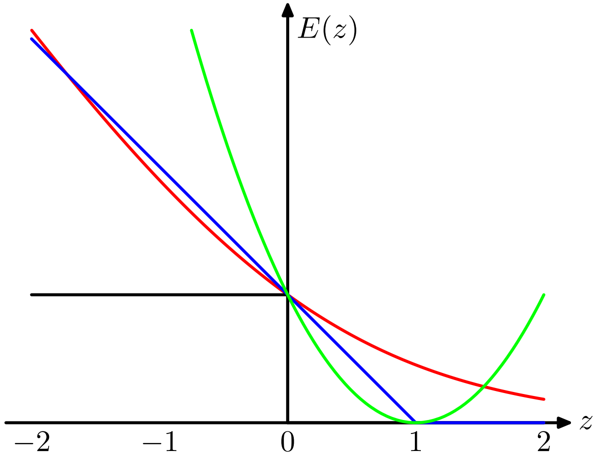


Figure 7.5 from Bishop: Plot of the 'hinge' error function used in support vector machines, shown in blue, along with the error function for logistic regression, rescaled by a factor of $1/\ln(2)$ so that it passes through the point $(0, 1)$, shown in red. Also shown are the misclassification error in black and the squared error in green.

Multiclass SVMs

SVMs are designed for binary classification but they can also be used for multi-class problem with K classes, $C_1, \ldots, C_k$.

There are two commonly used approaches:

1. one-versus-the-rest: train K different SVM's with the k^{th} SVM trained on "C_k" versus "not C_k"
    - not a good idea (why?) to use majority voting among the K classifiers
    - instead final classification usually determined by taking

    $$y(\mathbf{x}) = \max_k y_k(\mathbf{x}) \tag{22}$$

    where $y_k(\mathbf{x})$ is the linear classifier of k^{th} SVM
    - but there are scaling problems with using (22) and the balance of the training points in each of the K SVMs generally poor.
2. one-versus-one: train all $K(K-1)/2$ two-class SVMs and then use majority voting to obtain final classifier
    - a lot of computational work when K is large.

Application of SVMs to multi-class problems is ad-hoc with many limitations.

Regression and Support Vector Machines

SVMs have also been proposed for regression – quite popular in practice.

We replace quadratic error function with an ϵ-insensitive error function, $E_\epsilon(\cdot)$:

$$E_\epsilon\left(y(\mathbf{x}) - t\right) := \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases} \tag{23}$$

where $y(\mathbf{x}) := \mathbf{w}^\top \mathbf{x}_i + b$ and t is the dependent variable.

$E_\epsilon(\cdot)$ only penalizes predictions that are more than ϵ away from the target

- results in sparse solutions where only a subset of the training points matter.

Regularized objective function of SVM regression then given by

$$C \sum_{i=1}^{N} E_\epsilon\left(\mathbf{w}^\top \mathbf{x}_i + b - t_i\right) + \frac{1}{2}||\mathbf{w}||^2$$

– C can be chosen via cross-validation.

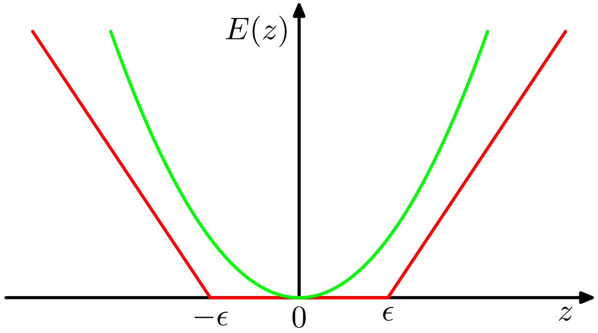


Figure 7.6 from Bishop: Plot of an ϵ-insensitive error function (in red) in which the error increases linearly with distance beyond the insensitive region. Also shown for comparison is the quadratic error function (in green).

Regression and Support Vector Machines

Can reformulate primal regression SVM problem as

$$\min_{\mathbf{w},b,\boldsymbol{\xi}\geq 0,\hat{\boldsymbol{\xi}}\geq 0} \quad C\sum_{i=1}^{n}\left(\xi_i+\hat{\xi}_i\right) \; + \; \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to} \quad t_i \;\leq\; y(\mathbf{x}_i)+\epsilon+\xi_i \tag{24}$$

$$t_i \;\geq\; y(\mathbf{x}_i)-\epsilon-\hat{\xi}_i \tag{25}$$

Slack variables $\boldsymbol{\xi}$ and $\hat{\boldsymbol{\xi}}$ non-zero only for predictions outside ϵ-insensitive tube.

Primal problem convex so can instead work with its dual problem

- obtained via the Lagrangian and relaxing (24), (25) and non-neg. constraints
- kernel trick can also be used with dual
- dual problem can be solved numerically to obtain a fitted function of the form

  $$y(\mathbf{x}) \;=\; \sum_{i=1}^{n}(a_i-\hat{a}_i)k(\mathbf{x},\mathbf{x}_i)+b$$

  where a_i's and $\hat{a}_i$'s are Lagrange multipliers for kernel-consistent versions of (24) and (25).

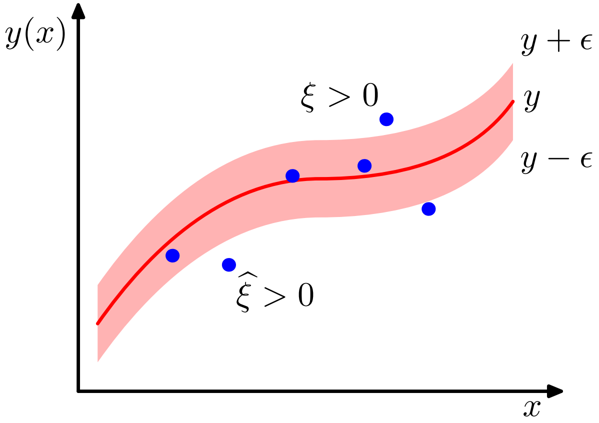


Figure 7.7 from Bishop: Illustration of SVM regression, showing the regression curve together with the ϵ-insensitive 'tube'. Also shown are examples of the slack variables ξ and $\hat{\xi}$. Points above the ϵ-tube have $\xi > 0$ and $\hat{\xi} = 0$, points below the ϵ-tube have $\xi = 0$ and $\hat{\xi} > 0$, and points inside the ϵ-tube have $\xi = \hat{\xi} = 0$.

Only points on the edge of the tube or outside the tube are support vectors.

Beyond SVMs

SVMs very popular for both classification and regression

- are also used for novelty detection.

But they do have some weaknesses

- they can only be used in an ad-hoc way for multi-class classification
- outputs of SVM classifier is a decision rather than a probability

Bishop proposes relevance vector machines (RVMs) as an alternative sparse kernel technique for regression and classification

- a Bayesian method which produces posterior class probabilities
- can handle multi-class classification
- much slower to train than SVMs but no need for cross-validation
- evaluation of test points much faster than SVMs due to generally greater sparsity.

Kernels and the Primal Problem

Kernels arose naturally in the dual problem but we can also use the kernel trick directly in the primal problem formulation.

To do this we need the concept of a reproducing kernel Hilbert space (RKHS).

Without going into too many details, a RKHS $\mathcal{H}_K$:

1. Is a (Hilbert) space of functions $f(\cdot)$
2. Is equipped with an inner product, $||\cdot||^2_{\mathcal{H}_K}$, corresponding to a kernel $K(\cdot,\cdot)$.

In fact there is a 1-1 correspondence between kernels and RKHS's.

Now consider the learning problem

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(t_i, f(\mathbf{x}_i)) + \lambda ||f||^2_{\mathcal{H}_K} \tag{26}$$

where λ is a regularization parameter that penalizes the "complexity" of f.

This is an infinite dimensional problem! But we have the following result:

The Representer Theorem

It can be shown that the solution to (26) has the form

$$f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j K(\mathbf{x}, \mathbf{x}_j). \tag{27}$$

Also, for any f of the form (27) we have

$$||f||^2_{\mathcal{H}_K} = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j). \tag{28}$$

It follows from (27) and (28) that we can write (26) as

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} L(t_i, \mathbf{K}_{i.}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \tag{29}$$

where $\mathbf{K}$ is the Gram matrix from (10) and $\mathbf{K}_{i.}$ is the i^{th} row of $\mathbf{K}$.

Note that (29) is a finite-dimensional problem!

Back to SVM's

We can now use (29) to formulate a kernelized primal SVM:

$$\min_{\boldsymbol{\alpha},b} \sum_{i=1}^{n} \left[1 - t_i(\mathbf{K}_{i.}\boldsymbol{\alpha} + b)\right]^{+} + \lambda \boldsymbol{\alpha}^{\top}\mathbf{K}\boldsymbol{\alpha} \tag{30}$$

Recently, the approach for very large problems has been to solve (30) directly using first-order methods and by smoothing the hinge function.

Remark: It should be clear from the RKHS formulation in (26) that the kernel trick does not work with $||\cdot||_1$ or Lasso-style norms.