Support Vector Machine

2012-10-18

Hao Wu

Introduction

Figures for the slides are obtained from Hastie *et al.* **The Elements of Statistical Learning**.

Problem setting:

- Given training data pairs $(x_1, y_1), \ldots, (x_N, y_N)$. x_i's are p-vector predictors. $y_i \in \{-1, 1\}$ are outcomes.
- Our goal: to find a classifier based on x.
- Such classifier is defined as a function of x, $G(x)$. G is estimated based on the training data (x, y) pairs.
- Once G is obtained, it can be used for future predictions.

There are many ways to construct $G(x)$, and Support Vector Machine (SVM) is one of them. We'll first consider the simple case: $G(x)$ is based on linear function of x. It's often called **linear SVM** or **support vector classifier** in Hastie book.

Simple case: perfectly separable case

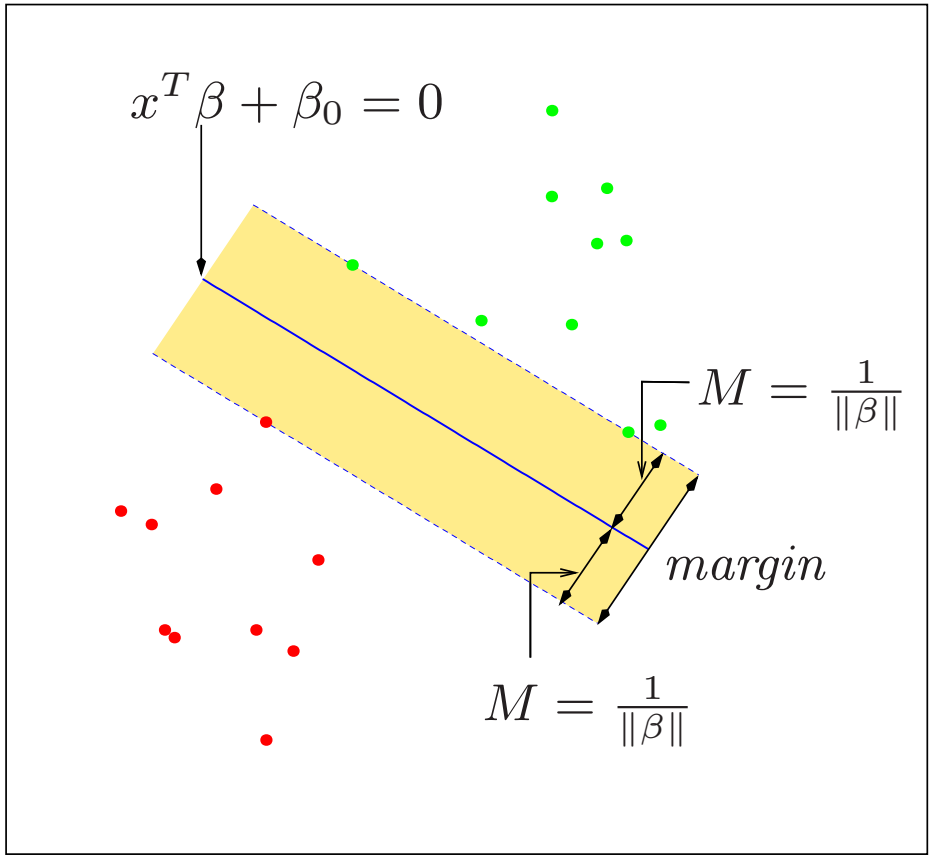
- First define a hyperplane by $\{x : f(x) = x^T b + b_0 = 0\}$, where b is a unit vector with $||b|| = 1$.
- A classification rule can be defined as $G(x) = sign[x^T b + b_0]$.
- The problem is to estimate b's.

Consider a simple case where two groups are perfectly separated. We want to find a "border" to separate two groups.

- There are infinite number of borders can perfectly separate two groups. Which one is optimal?
- Conceptually, the optimal border should separates the two classes with the largest margins.
- We define the optimal border to be the one satisfying: (1) the distances between the closest points to the border are the same in both groups, denote the distance by M; and (2) M is maximized.

M is called the "margin".

Illustration of the optimal border (solid line) with margins (dash lines).



Problem setup

Then problem to find the best border can be framed into following optimization problem:

$$\begin{aligned} \max_{\beta,\beta_0} \quad & M \\ s.t. \quad & y_i(x_i^T b + b_0) \geq M, i = 1, \ldots, N \end{aligned}$$

This is not a typical LP/QP problem so we do some transformations to make it look more familiar.

Divided both sides of the constraint by M, and define $\beta = b/M$, $\beta_0 = b_0/M$, the constraints become:

$$y_i(x_i^T \beta + \beta_0) \geq 1.$$

This means that we scale the coefficients of the border hyperplane, so that the margin lines are in the forms of $x_i^T \beta + \beta_0 + 1 = 0$ (upper margin) and $x_i^T \beta + \beta_0 - 1 = 0$ (lower margin).

Now we have

$$\|\beta\| = \|b\|/M = 1/M.$$

So the objective function (maximizing M) is equivalent to minimizing $\|\beta\|$.

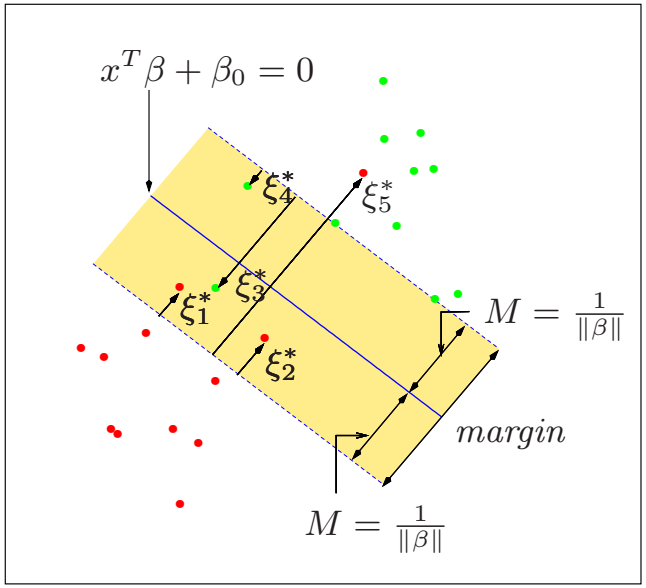
After this transformation, the optimization problem can be expressed as a simpler, more familiar form:

$$\begin{aligned} \min_{\beta,\beta_0} \quad & \|\beta\| \\ s.t. \quad & y_i(x_i^T\beta + \beta_0) \geq 1, i = 1, \ldots, N \end{aligned}$$

This is a typical quadratic program problem can be solved by interior point method.

Non-separable case

When two classes are not perfectly separable, we still want to find a border with two margins (why?). But now there will be points on the wrong sides. We introduce slack variables to account for those points.



Define slack variables $\{\xi_1, \ldots, \xi_N\}$, where $\xi_i \geq 0 \forall i$ and

- $\xi = 0$ when the point is on the correct side of the margin.
- $\xi > 1$ when the point passes the border to the wrong side.
- $0 < \xi < 1$ when the point is in the margin but still on the correct side.

Now the constraints in the original optimization problem is modified to:

$$y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, i = 1, \ldots, N$$

- ξ_i can be interpreted as the proportional amount by which the predication is on the wrong side of the margin.
- Anther constraint $\sum_i \xi_i \leq C$ is often added to bound the total number of misclassification.
- Put together, the optimization problem for this case is often written as :

$$\begin{aligned} \min_{\beta,\beta_0} \quad & \frac{1}{2}\|\beta\|^2 \\ s.t. \quad & y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \\ & \sum_i \xi_i \leq C, \quad \xi_i \geq 0 \end{aligned}$$

Again this is a quadratic programming problem. What are the unknowns?

Computation

The primal Lagrangian is:

$$L_P = \frac{1}{2}\|\beta\|^2 + \gamma \sum_i \xi_i - \sum_i \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] - \sum_i \mu_i \xi_i$$

Take derivatives of β, β_0, ξ_i then set to zero, get (the stationary conditions) :

$$\beta = \sum_i \alpha_i y_i x_i$$

$$0 = \sum_i \alpha_i y_i$$

$$\alpha_i = \gamma - \mu_i, \forall i$$

Plus these back to the primal Lagrangian (using the first two equalities), get the following dual objective function (verify):

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_i{}' \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}$$

.

The L_D needs to be maximized subject to constraints:

$$\sum_i \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq \gamma$$

The KKT conditions for the problem (in additional to the stationary conditions) include following complementary slackness and primal/dual feasibilities:

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] = 0$$

$$\mu_i \xi_i = 0$$

$$y_i(x_i^T\beta + \beta_0) - (1 - \xi_i) \geq 0$$

$$\alpha_i, \mu_i, \xi_i \geq 0$$

The QP problem can be solved using interior point method based on these.

The support vectors

At optimal solution, β is in the form of: $\hat{\beta} = \sum_i \hat{\alpha}_i y_i x_i$.

This means $\hat{\beta}$ is a linear combination of $y_i x_i$, and only depends on those data points with $\hat{\alpha} \neq 0$. These data points are called "**support vectors**".

According to the complmentary slackness in the KKT conditions, at optimal point we have:

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] = 0, \quad \forall i$$

which means α_i could be non-zero only when $y_i(x_i^T\beta + \beta_0) - (1 - \xi_i) = 0$.

What this result tell us?

For points with non-zero α_i:

- The points with $\xi_i = 0$ will have $y_i(x_i^T\beta + \beta_0) = 1$, or these points are on the margin lines. Based on the KKT conditions we know these points will have $0 < \alpha_i < \gamma$ (why)?
- Other points with $y_i(x_i^T\beta + \beta_0) = 1 - \xi_i$ are on the wrong side of the margins.

So only the points on the margin or at the wrong side of the margin are informative for the separating hyperplane. That's why these points are called the "**support vectors**", because they provide "support" for the decision rule.

This makes sense, because the points that can be correctly separated and "far away" from the margin (those "easy" points) don't tell us anything about the classification rule (the hyperplane).

Support Vector Machine

We have discussed **support vector classifier**, which uses hyperplane to separate two groups. **Support Vector Machine** enlarges the feature space to make the procedure more flexible.

To be specific, we transform the input data x_i using some basis functions $h_m(x), m = 1, \ldots, M$. Now the input data become $h(x_i) = (h_1(x_i), \ldots, h_M(x_i))$. This basically transform the data to another space, which could be nonlinear in the original space.

We then find SV classifier in the transformed space using the same procedure, e.g., find optimal

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0.$$

And the decision is made by: $\hat{G}(x) = sign(\hat{f}(x))$.

Note: the classifier is linear in the transformed space, but nonlinear in the original one.

Choose basis function?

Now the problem becomes the choice of basis function, or do we even need to choose basis function.

Recall in the linear space, β is in the form of:

$$\beta = \sum_i \alpha_i y_i x_i.$$

In the transformed space, it becomes:

$$\beta = \sum_i \alpha_i y_i h(x_i).$$

So the decision boundary is:

$$f(x) = h(x)^T \sum_i \alpha_i y_i h(x_i) + \beta_0 = \sum_i \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0.$$

Moreover, the dual objective function in transformed space becomes:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_i' \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle$$

.

What does this tell us?

Both the objective function and the decision boundary in the transformed space involves only the inner products of the transformed data, not the transformation itself!

So the basis functions are not important, as long as we know $\langle h(x), h(x_i) \rangle$.

Kernel tricks

Define the kernal function $K : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$, to represent the inner product in the transformed space:

$$K(x, x') = \langle h(x), h(x') \rangle.$$

K needs to be a symmetric and positive semi-definite. With the kernel trick, the decision boundary becomes:

$$f(x) = \sum_i \alpha_i y_i K(x, x_i) + \beta_0.$$

Some popular choices of the kernel functions are:

- Polynomial with d degree: $K(x, x') = (1 + \langle x, x' \rangle)^d$.
- Radial basis: $K(x, x') = exp\{-\|x - x'\|^2/c\}$.
- Neural network: $K(x, x') = tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$.

Computation of SVM

With kernels defined, the Lagrangian dual function is:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_i{}' \alpha_i \alpha_{i'} y_i y_{i'} K(x_i, x_{i'})$$

.

Maximize L_D, with α_i's being the unknowns, subject to the same constrains:

$$\sum_i \alpha_i y_i = 0$$

$$0 < \alpha_i < \gamma$$

This is a standard QP problem can be solved easily.

With $\hat{\alpha}_i$ given, we still need to get $\hat{\beta}_0$ to construct the decision boundary. **Note**: β is not needed since the decision boundary doesn't involve it.

Remember the points on the margin line has $y_i f(x_i) = 1$, and they have $0 < \hat{\alpha}_i < \gamma$.

So we obtain $\hat{\beta}_0$ by using data points with $0 < \hat{\alpha}_i < \gamma$ to solve $y_i f(x_i) = 1$. We will get one β_0 for each of such point. In practice we often use the average of those to get a stable result.

The role of γ

To control the smoothness of boundary.

Remember γ is introduced in the primal problem to control the total misclassification, e.g., dual variable for original constraint $\sum_i \xi_i \leq C$. we can always project the original data to higher dimensional space so that they can be better separated by a linear classifier (in the transformed space), but

- Large γ: fewer error in transformed space, wiggly boundary in original space.
- Small γ: more errors in transform space, smoother boundary in original space.

γ is a tuning parameter often obtained from cross-validation.

A little more about the decision rule

Recall the decision boundary only depends on support vectors, or the points with $\alpha_i \neq 0$. So $f(x)$ can be written as:

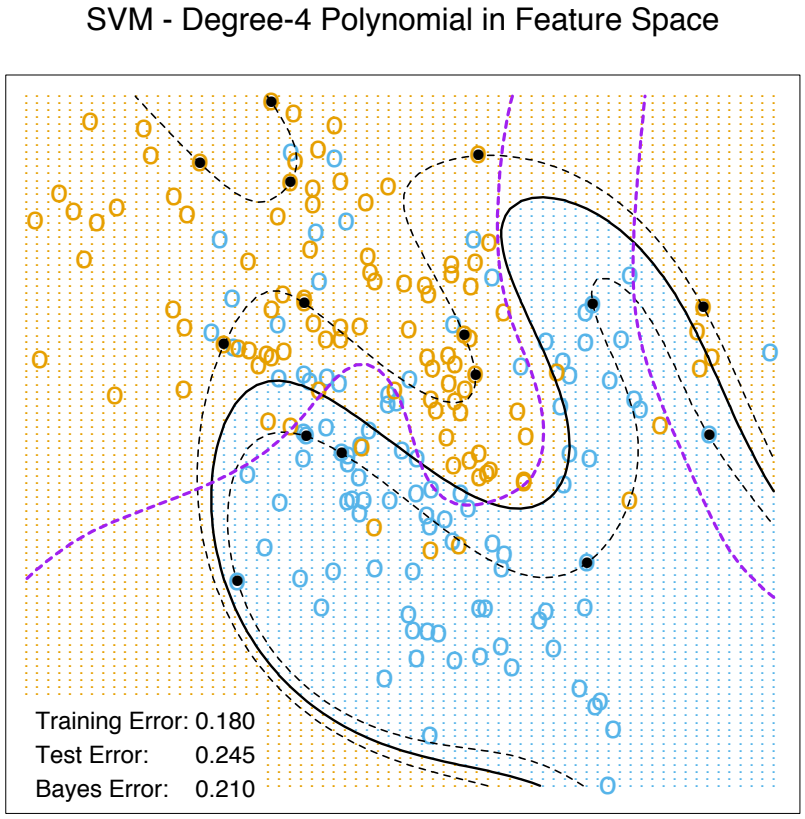
$$f(x) = \sum_{x_i \in S} \alpha_i y_i K(x, x_i) + \beta_0,$$

where S is the set of support vectors.

The kernel $K(x, x')$ can be seen as a similarity measure between x and x'. So to classify for point x, the decision is made essentially by a weighted sum of similarity of x to all the support vectors.

An example

SVM using 4-degree polynomial kernal. Decision boundary projected into 2-D space.



SVM in R

There are several `R` packages include SVM function: `e1071`, `kernlab`, `klaR`, `svmpath`, etc.

Table below summarize the R SVM functions. For more details please refer to the "*Support Vector Machines in R*" paper at the class website.

	`ksvm()` (**kernlab**)	`svm()` (**e1071**)	`svmlight()` (**klaR**)	`svmpath()` (**svmpath**)
Formulations	C-SVC, ν-SVC, C-BSVC, spoc-SVC, one-SVC, ϵ-SVR, ν-SVR, ϵ-BSVR	C-SVC, ν-SVC, one-SVC, ϵ-SVR, ν-SVR	C-SVC, ϵ-SVR	binary C-SVC
Kernels	Gaussian, polynomial, linear, sigmoid, Laplace, Bessel, Anova, Spline	Gaussian, polynomial, linear, sigmoid	Gaussian, polynomial, linear, sigmoid	Gaussian, polynomial

Summary of SVM

Strengths of SVM:

- flexibility.
- scales well for high-dimensional data.
- can control complexity and error trade-off explicitly.
- as long as a kernel can be defined, non-traditional (vector) data, like strings, trees can be input.

Weakness:

- how to choose a good kernel (a low degree polynomial or radial basis function can be a good start).

A grand review of the class

We have covered following topics in this class:

- MCMC techniques.
- EM and MM algorithms.
- Hidden Markov models: Viterbi and forward-back algorithms.
- Linear/quadratic programming with applications in quantile regression, LASSO and SVM.

We also implemented the first 3 topics in homework.