Support Vector Machines

Héctor Corrada Bravo

University of Maryland, College Park, USA

Fannie Mae: 2017-08-04

Support Vector Machines

State-of-the-art classification and regression method

Flexible and efficient framework to learn classifers.

Support Vector Machines

State-of-the-art classification and regression method

Flexible and efficient framework to learn classifers.

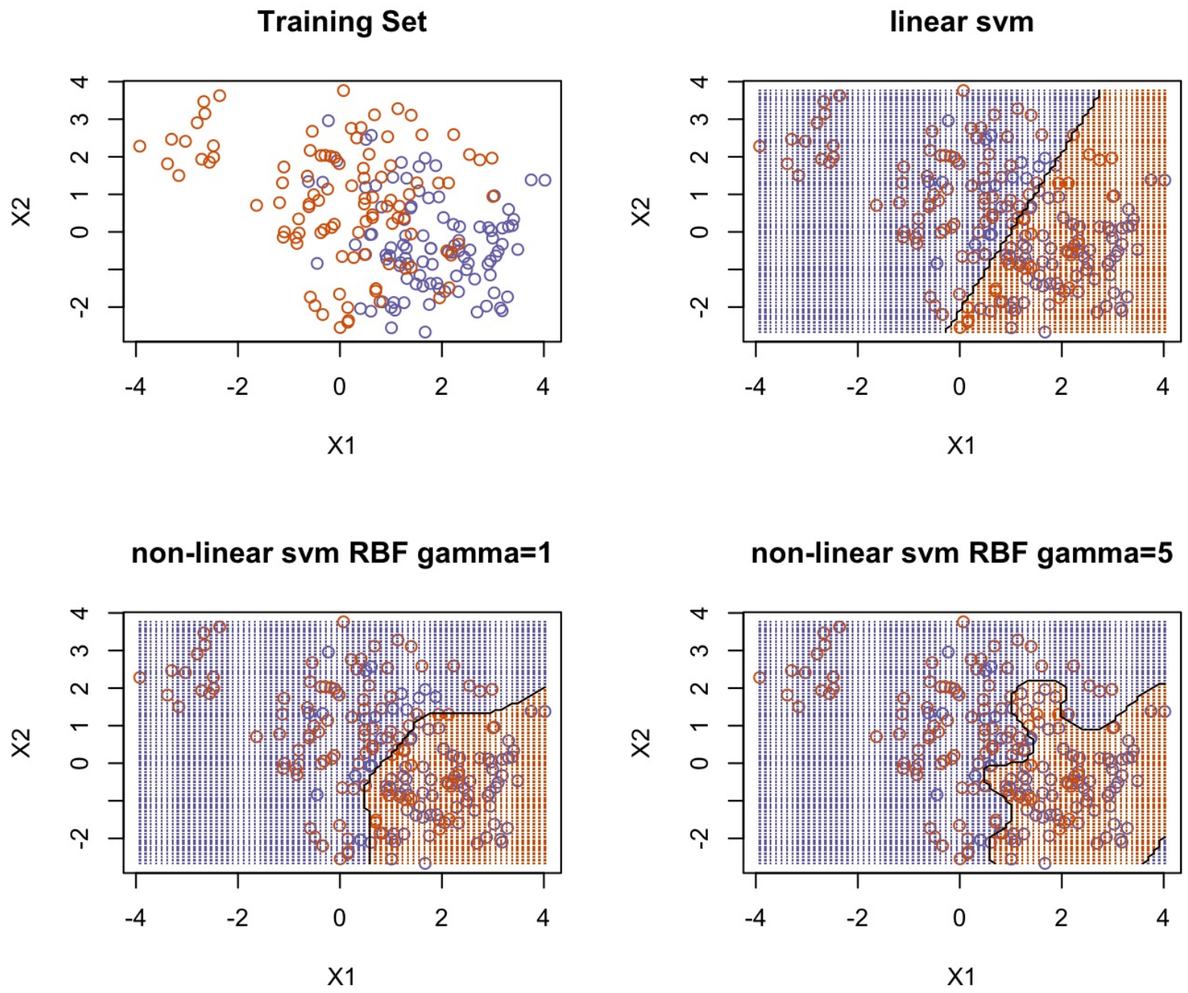
Build upon linear methods we have discussed previously and have a nice geometric interpretation of how they are trained (based maximum margin arguments).

Support Vector Machines

Can be estimated over similarities between observations (more on this later) rather than standard data in tabular form.

E.g., applications where string similarities, or network similarities are readily available.

Support Vector Machines



SVMs follow the "predictor space partition" framework

Separating Hyperplanes

Training data: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

- $\mathbf{x}_i$ is a vector of p predictor values for ith observation,
- y_i is the class label (we're going to use +1 and -1)

Separating Hyperplanes

Training data: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

- $\mathbf{x}_i$ is a vector of p predictor values for ith observation,
- y_i is the class label (we're going to use +1 and -1)

Build a classifier by defining a discriminative function such that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \, \text{if} \, y_i = 1$$

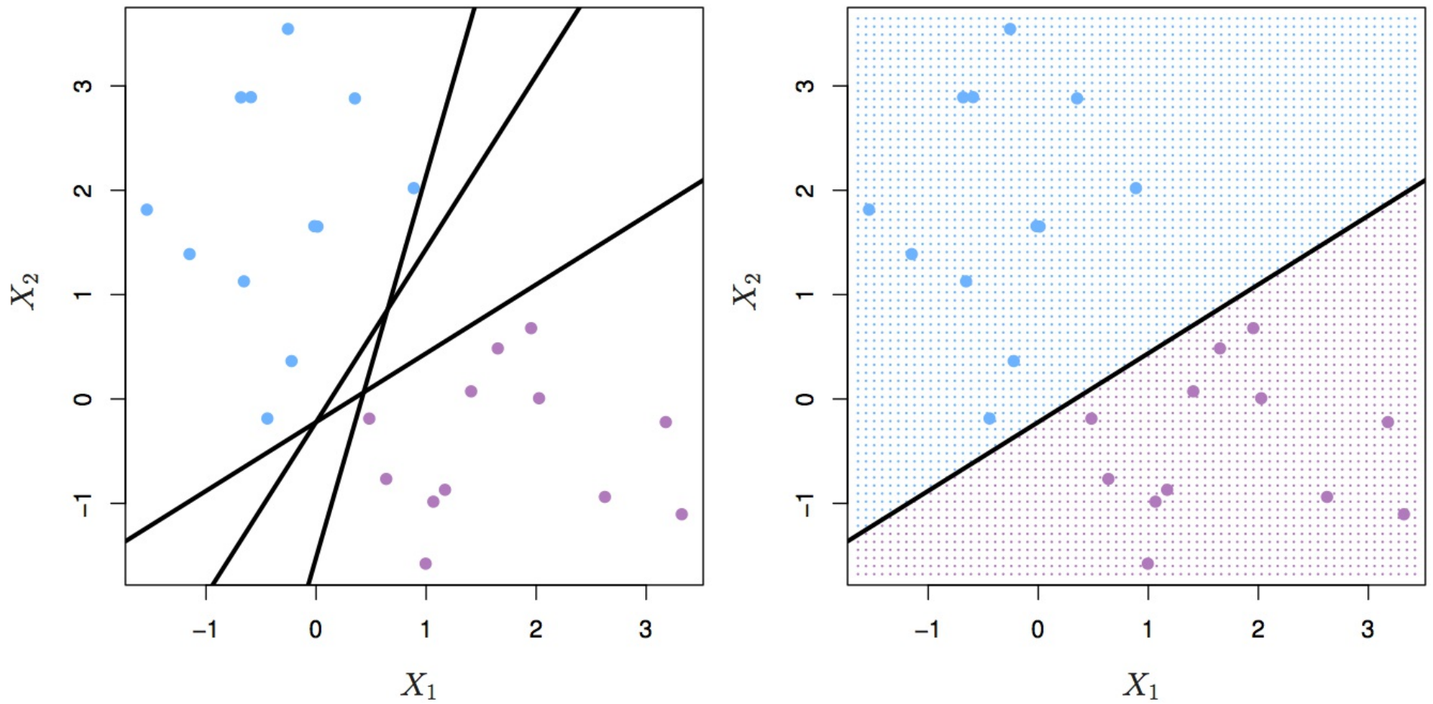
and

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \, \text{if} \, y_i = -1$$

Separating Hyperplanes

Points where the discriminative function equals 0 form a hyper-plane (i.e., a line in 2D)

$$\{x : \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = 0\}$$



Separating Hyperplanes

Hyper-plane partitions the predictor space into two sets on each side of the line.

Denote β is the vector $(\beta_1, \beta_2, \ldots, \beta_p)$

Separating Hyperplanes

Hyper-plane partitions the predictor space into two sets on each side of the line.

Denote β is the vector $(\beta_1, \beta_2, \ldots, \beta_p)$

Restrict estimates to those for which $\beta'\beta = ||\beta||^2 = 1$

Separating Hyperplanes

Hyper-plane partitions the predictor space into two sets on each side of the line.

Denote β is the vector $(\beta_1, \beta_2, \ldots, \beta_p)$

Restrict estimates to those for which $\beta'\beta = ||\beta||^2 = 1$

Then, the signed distance of any point x to the decision boundary L is $\beta_0 + \beta' x$.

Separating Hyperplanes

With this we can easily describe the two partitions as

$$L^+ = x : \beta_0 + \beta' x > 0,$$
$$L^- = x : \beta_0 + \beta' x < 0$$

Separating Hyperplanes

The β we want as an estimate is one that separates the training data as perfectly as possible.

Separating Hyperplanes

The β we want as an estimate is one that separates the training data as perfectly as possible.

Describe this requirement as

$$y_i(\beta_0 + \beta' x_i) > 0, i = 1, \ldots, N$$

Rosenblatt's algorithm

Algorithm to find vector β that satisfies the separation requirement as much as possible.

Penalize β by how far into the wrong side misclassified points are:

$$D(\beta_0, \beta) = -\sum_{i \in \mathcal{M}} y_i(\beta_0 + \beta' x_i)$$

$\mathcal{M}$: set of points misclassified by β (on the wrong side of the hyper-plane).

Rosenblatt's Algorithm

Estimate β by minimizing D.

Assuming $\mathcal{M}$ is fixed, the gradient of D is

$$\frac{\partial D(\beta_0, \beta)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i$$

and

$$\frac{\partial D(\beta_0, \beta)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

Rosenblatt's Algorithm

Rosenblatt's algorithm uses stochastic gradient descent:

- Initialize parameters β_0 and β
- Cycle through training points i, if it is misclassified, update parameters as

$$\beta \leftarrow \beta + \rho y_i x_i$$

and

$$\beta_0 \leftarrow \beta_0 + \rho y_i$$

- Stop when converged (or get tired of waiting)

Rosenblatt's Algorithm

Update Rule:

$$\beta \leftarrow \beta + \rho y_i x_i$$

Learning rate parameter ρ is used to control how much we update β in each step.

Rosenblatt's Algorithm

Update Rule:

$$\beta \leftarrow \beta + \rho y_i x_i$$

Learning rate parameter ρ is used to control how much we update β in each step.

This basic algorithm will form the basis of our work on neural networks and deep learning later on.

Rosenblatt's Algorithm

There are a few problems with this algorithm:

If there exists β_0 and β that separates the training points perfectly,

Rosenblatt's Algorithm

There are a few problems with this algorithm:

If there exists β_0 and β that separates the training points perfectly,

then there are an infinite number of β_0 and βs that also separate the data perfectly

Rosenblatt's Algorithm

Algorithm will converge in a finite number of steps if the training data is separable

Rosenblatt's Algorithm

Algorithm will converge in a finite number of steps if the training data is separable

However, the number of finite steps can be very large

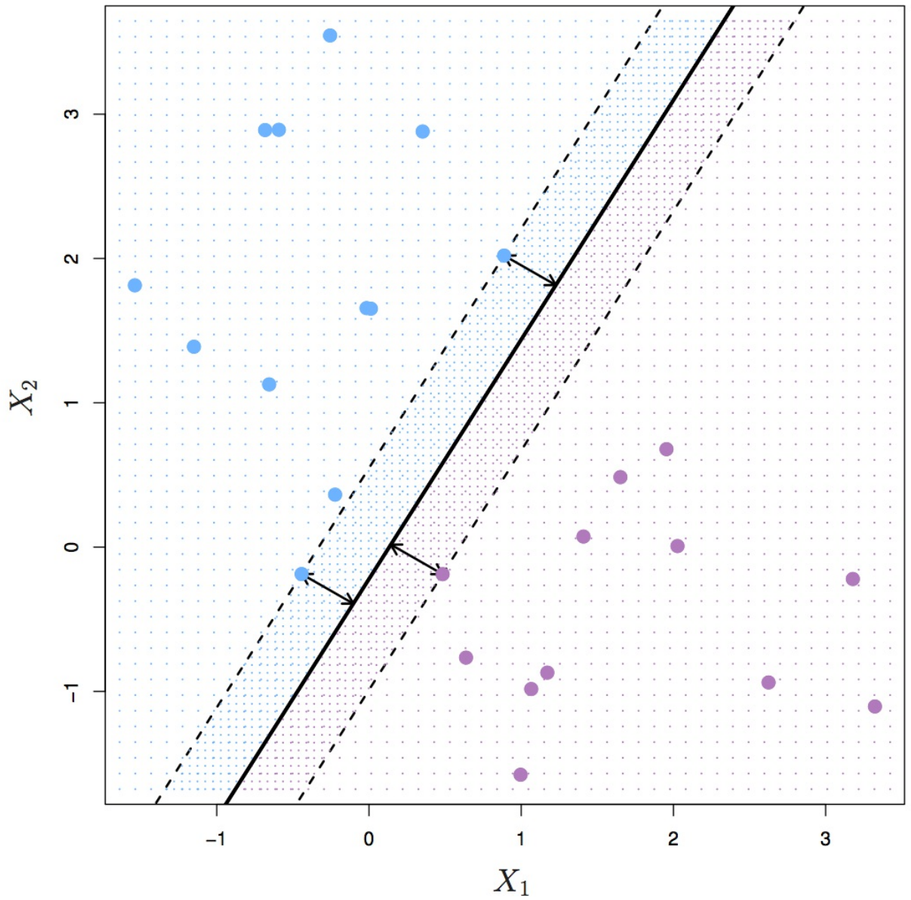
Rosenblatt's Algorithm

When the training data is not separable, the algorithm will not converge.

Support Vector Machines

Support Vector Machines (SVMs) are designed to directly address these problems.

Support Vector Machines



A central concept in SVMs that we did not see in logistic regression is **the margin**: the distance between the separating plane and its nearest datapoints.

Support Vector Machines

When the data are separable, SVMs will choose the single optimal β that maximizes the distance between the decision boundary and the closest point in each class.

Support Vector Machines

When the data are separable, SVMs will choose the single optimal β that maximizes the distance between the decision boundary and the closest point in each class.

Why is this a good idea?

Support Vector Machines

SVMs are designed from three key insights:

1. **Look for the maximum margin hyper-plane**
2. Only depend on pair-wise "similarities" of observations
3. Only depend on a subset of observations (support vectors)

Let's see these in turn.

Maximum margin hyper-planes

Goal: find the hyper-plane that separates training data with largest margin.

This will tend to generalize better since new observations have room to fall within margin and still be classified correctly.

Maximum margin hyper-planes

This can be cast as optimization problem:

$$\max_{\beta_0,\beta} M$$
$$\text{s.t.} |\beta|^2 = 1$$
$$y_i(\beta_0 + \beta' x_i) \geq M \, \forall i$$

Maximum margin hyper-planes

Rewrite optimization problem setting $M = 1/||\beta||^2$ and using a little bit of algebra (see Section 4.5 of Hastie and Tibshirani):

$$\min_{\beta_0,\beta} \frac{1}{2}|\beta|^2$$
$$\text{s.t.} y_i(\beta_0 + \beta' x_i) \geq 1 \forall i$$

Maximum margin hyper-planes

$$\min_{\beta_0,\beta} \frac{1}{2}|\beta|^2$$
$$\text{s.t.} y_i(\beta_0 + \beta' x_i) \geq 1 \forall i$$

This is a constrained optimization problem

Minimize the norm of β under the constraint that it classifies every observation correctly.

Maximum-margin hyper-planes

We can switch between equivalent constrained minimization and constrained maximization problems.

In the maximum-margin hyper-plane case, the equivalent constrained maximization problem (the dual problem) is:

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i' x_k$$
$$\text{s.t.} \alpha_i = 0 \, \forall i$$

Maximum margin hyper-planes

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i' x_k$$
$$\text{s.t.} \alpha_i = 0 \, \forall i$$

This quadratic optimization problem is usually easier to optimize than the original problem (notice there is only positivity constraints on α).

Support Vector Machines

Key insight no. 2: **SVMs only depend on pairwise "similarity" functions of observations**

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i' x_k$$
$$\text{s.t.} \alpha_i = 0 \, \forall i$$

Only inner products between observations are required as opposed to the observations themselves.

Support Vector Machines

Also, we can write the discriminant function in equivalent form

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i x' x_i$$

Support Vector Machines

Also, we can write the discriminant function in equivalent form

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i x' x_i$$

Geometrically, we can think of the inner product between observations as a "similarity" measure.

Support Vector Machines

Also, we can write the discriminant function in equivalent form

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i x' x_i$$

Geometrically, we can think of the inner product between observations as a "similarity" measure.

Therefore, we can fit these models with other measures that works as "similarities".

Support Vector Machines

Key insight no. 3: **SVMs only depend on a subset of observations (support vectors)**

Optimial solutions β and α must satisfy the following condition:

$$\alpha_i[y_i(\beta_0 + \beta' x_i) - 1] = 0 \, \forall i.$$

Support Vector Machines

$$\alpha_i[y_i(\beta_0 + \beta' x_i) - 1] = 0 \, \forall i.$$

Case 1: $\alpha_i > 0$, then the signed distance between observation x_i and the decision boundary is 1.

This means that observation x_i is on the margin

Support Vector Machines

$$\alpha_i[y_i(\beta_0 + \beta' x_i) - 1] = 0 \, \forall i.$$

Case 2: $y_i(\beta_0 + \beta' x_i) > 1$, then observation x_i is not on the margin and $\alpha_i = 0$.

Support Vector Machines

To define the discriminant function in terms of αs we only need observations that are on the margin,

i.e., those for which $\alpha_i > 0$.

Support Vector Machines

To define the discriminant function in terms of αs we only need observations that are on the margin,

i.e., those for which $\alpha_i > 0$.

These are called support vectors.

Support Vector Machines

To define the discriminant function in terms of αs we only need observations that are on the margin,

i.e., those for which $\alpha_i > 0$.

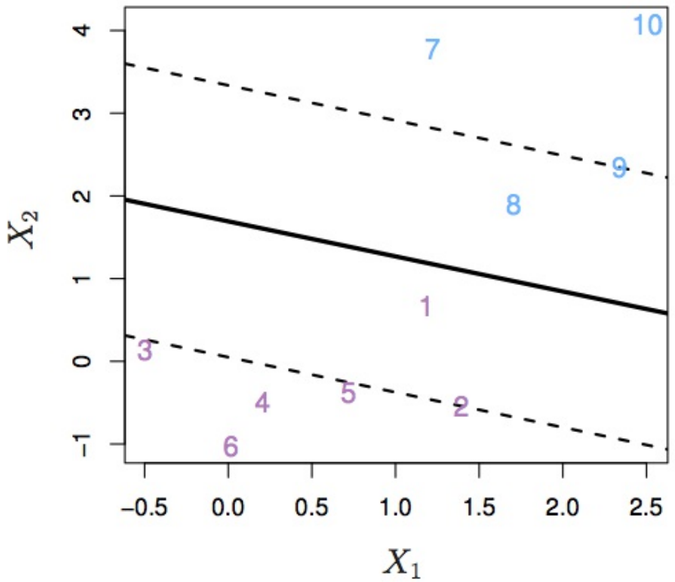
These are called support vectors.

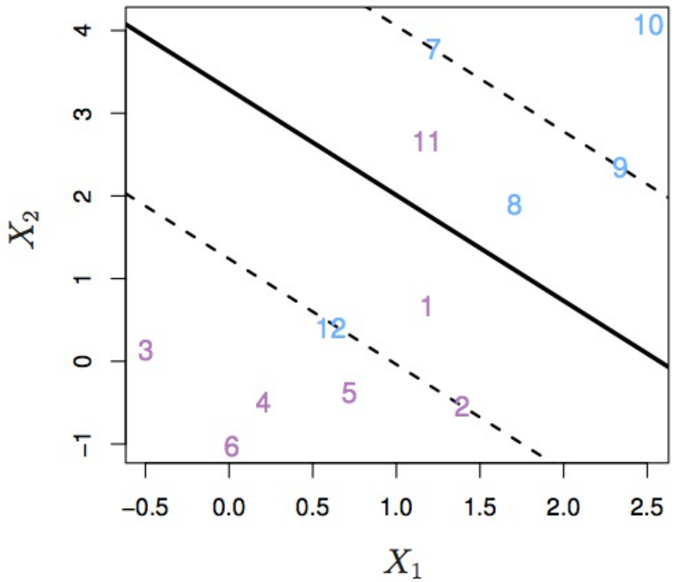
Also implies we only need Support Vectors to make predictions.

Non-separable data

The method we have discussed so far runs into an important complication:

What if there is no separating hyper-plane?.





Non-separable data

The solution is to penalize observations on the **wrong side of the margin** by introducing slack variables to the optimization problem.

$$\min_{\beta_0,\beta,\xi} C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\|\beta\|^2$$
$$\text{s. t } y_i(\beta_0 + \beta' x_i) \geq 1 - \xi_i \, \forall i$$
$$\xi_i \geq 0 \, \forall i$$

Non-separable data

$$\min_{\beta_0,\beta,\xi} C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\|\beta\|^2$$
$$\text{s. t } y_i(\beta_0 + \beta' x_i) \geq 1 - \xi_i \,\forall i$$
$$\xi_i \geq 0 \,\forall i$$

C is a parameter that tradeoffs the width of the margin vs. the penalty on observations on the wrong side of the margin.

Non-separable data

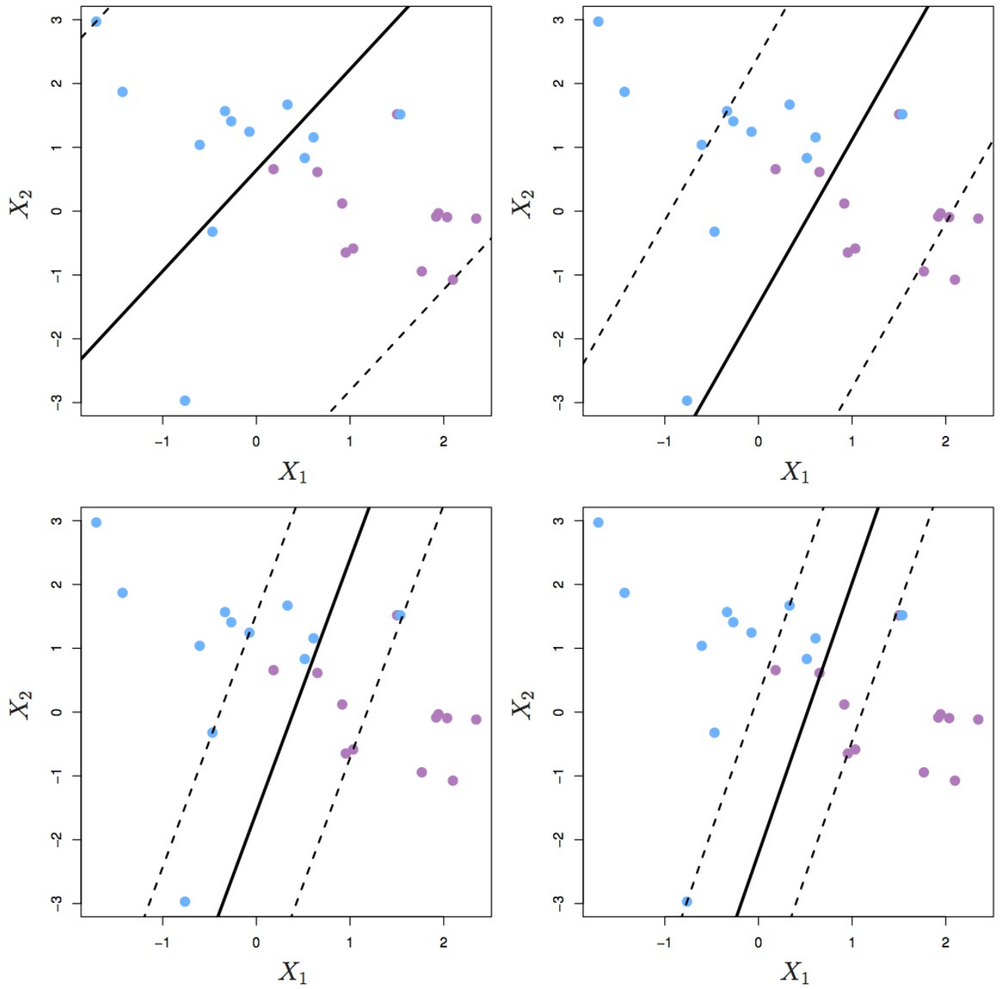
$$\min_{\beta_0,\beta,\xi} C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\|\beta\|^2$$
$$\text{s. t } y_i(\beta_0 + \beta' x_i) \geq 1 - \xi_i \, \forall i$$
$$\xi_i \geq 0 \, \forall i$$

C is a parameter that tradeoffs the width of the margin vs. the penalty on observations on the wrong side of the margin.

This is a "data fit + model complexity" learning objective.

Non-separable data

C is a hyper-parameter to be selected by the user or via cross-validation model selection methods.



Non-separable data

An elegant result is that this formulation doesn't change the dual problem we saw before very much:

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i' x_k$$
$$\text{s.t. } 0 \leq \alpha_i \leq C \forall i$$

Non-separable data

Only need support vectors, where $\alpha_i > 0$ to define the discriminant function and make predictions.

Non-separable data

Only need support vectors, where $\alpha_i > 0$ to define the discriminant function and make predictions.

The smaller the cost parameter C, the learned SVM will have fewer support vectors.

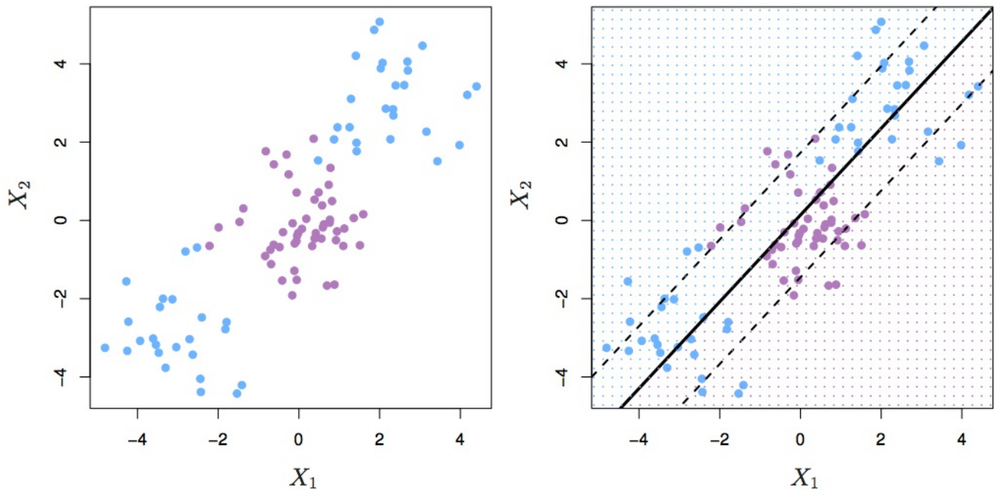
Non-separable data

Only need support vectors, where $\alpha_i > 0$ to define the discriminant function and make predictions.

The smaller the cost parameter C, the learned SVM will have fewer support vectors.

Think of the number of support vectors as a rough measure of the complexity of the SVM obtained.

Non-linear Support Vector Machine



What to do when we need non-linear partitions of predictor space to get a classifier?

Non-linear Support Vector Machine

We can define the SVM discriminant function in terms of inner products of observations.

We can generalize inner product using "kernel" functions that provide something like an inner product:

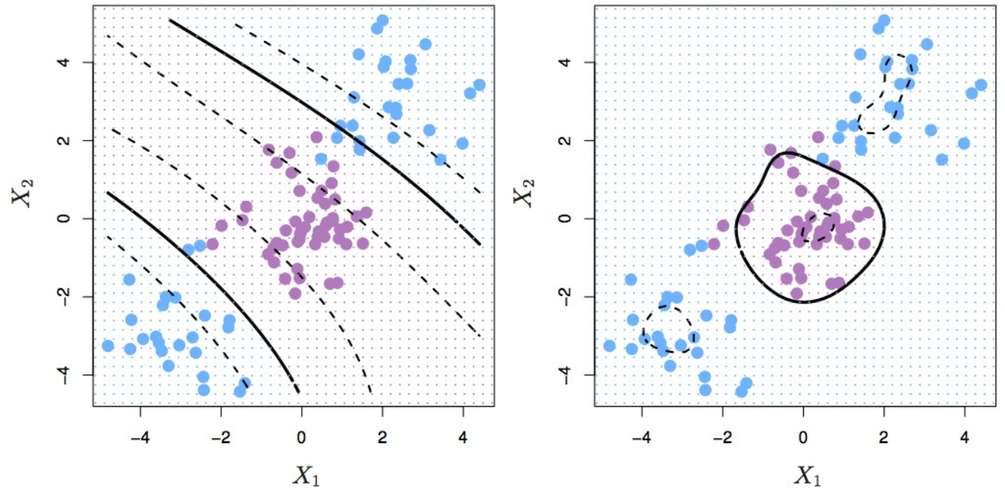
$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

Non-linear Support Vector Machine

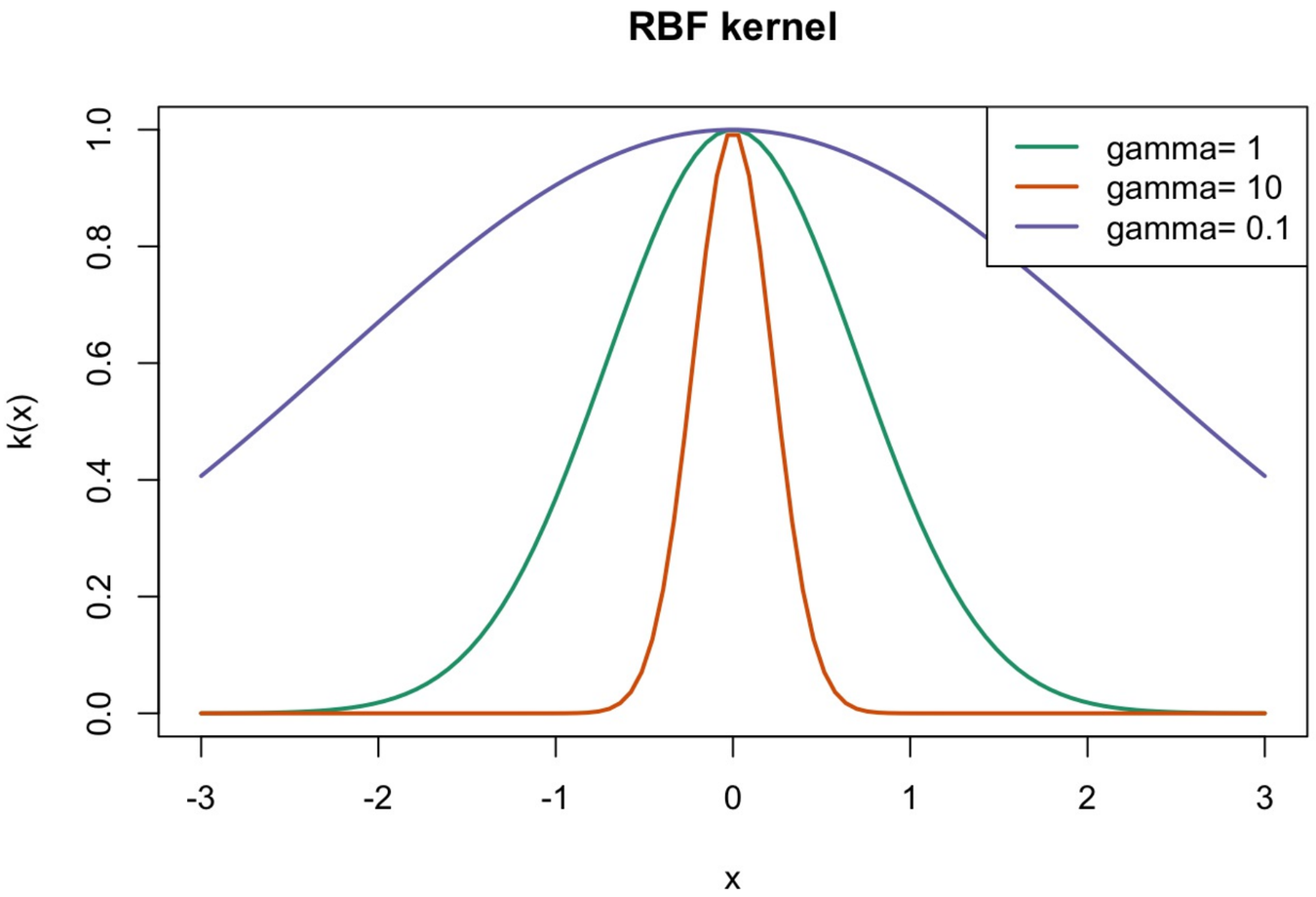
But, what is k? Let's consider two examples.

- Polynomial kernel: $k(x, x_i) = 1 + \langle x, x_i \rangle^d$
- RBF (radial) kernel:

$$k(x, x_i) = \exp\{-\gamma \sum_{j=1}^{p} (x_j - x_{ij})^2\}$$



Non-linear Support Vector Machine



Non-linear Support Vector Machine

The optimization problem is very similar

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k k(x_i, x_k)$$
$$\text{s.t. } 0 \leq \alpha_i \leq C \forall i$$

SVM classification example

Let's try fitting SVMs to the credit card default dataset we saw in previous examples.

Let's start with a linear SVM (where k is the inner product).

SVM classification example

Here we are fitting three different SVMs resulting from using three different values of cost parameter c.

cost	number_svs	train_error	test_error
1e-02	352	3.48	3.18
1e+00	359	3.48	3.18
1e+02	364	3.48	3.18

SVM classification example

Let's try now a non-linear SVM by using a radial kernel.

Notice now that we have two parameters to provide to the fitting function: cost parameter C and parameter γ of the radial kernel function.

SVM classification example

cost	gamma	number_svs	train_error	test_error
0.01	0.01	348	3.48	3.18
1.00	0.01	359	3.48	3.18
10.00	0.01	352	3.48	3.18
0.01	1.00	406	3.48	3.18
1.00	1.00	432	2.88	2.46
10.00	1.00	382	2.78	2.54
0.01	10.00	498	3.48	3.18
1.00	10.00	1131	2.62	2.88
10.00	10.00	944	2.30	3.10

Support Vector Machines

Different algorithms depending on data size

- Massive number of examples with few predictors, train with stochastic gradient descent on the primal problem
- Moderate number of examples, use quadratic optimization with kernel functions
- For quadratic version, can subset observations that could be support vectors

Support Vector Machines

State-of-the-art for many applications

RBF kernels usually work well, but tuning γ properly is very important

Very elegant formulation

Kernel trick gives a lot of flexibility