Support Vector Machines

Szu-Chi Chung

Department of Applied Mathematics, National Sun Yat-sen University

Why Support Vector Machine?

- For classification task we only need to know the hyperplane. Why bother a model?
- Support Vector Machine has the following advantages
  - Effective in high dimensional spaces
  - Still effective in cases where number of dimensions is greater than the number of samples
  - Uses a subset of training points in the decision function (called support vectors), so it is also *memory efficient*
  - Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels
- The disadvantage is that it does not directly provide probability estimates. In addition, if $p \gg n$, the regularization is crutial

Support Vector Machines

- Here we approach the two-class classification problem in a direct way:
  - We try and find a plane that separates the classes in feature space

- If we cannot, we get creative in two ways:
  - We soften what we mean by "separates", and
  - We enrich and enlarge the feature space so that separation is possible

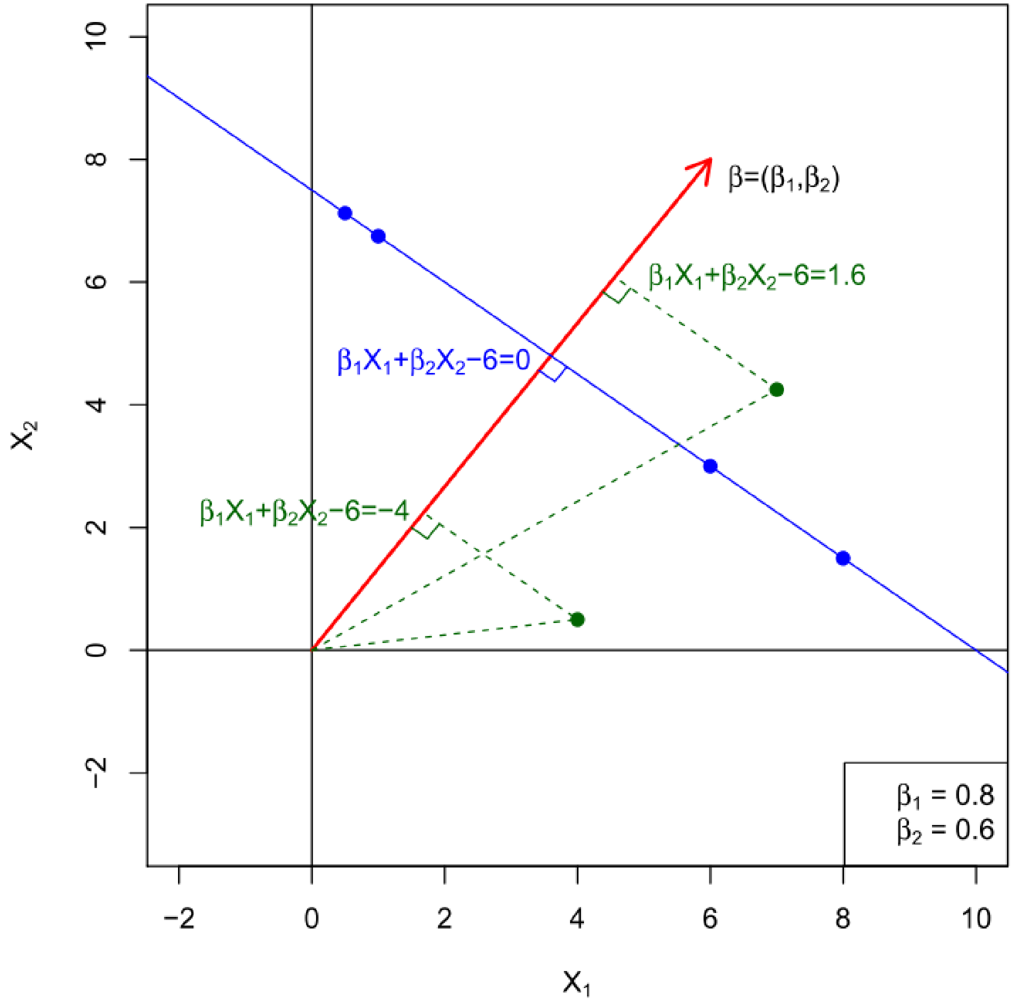
What is a Hyperplane?

- A hyperplane in p dimensions is a flat affine subspace of dimension $p - 1$
  - In general the equation for a hyperplane has the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

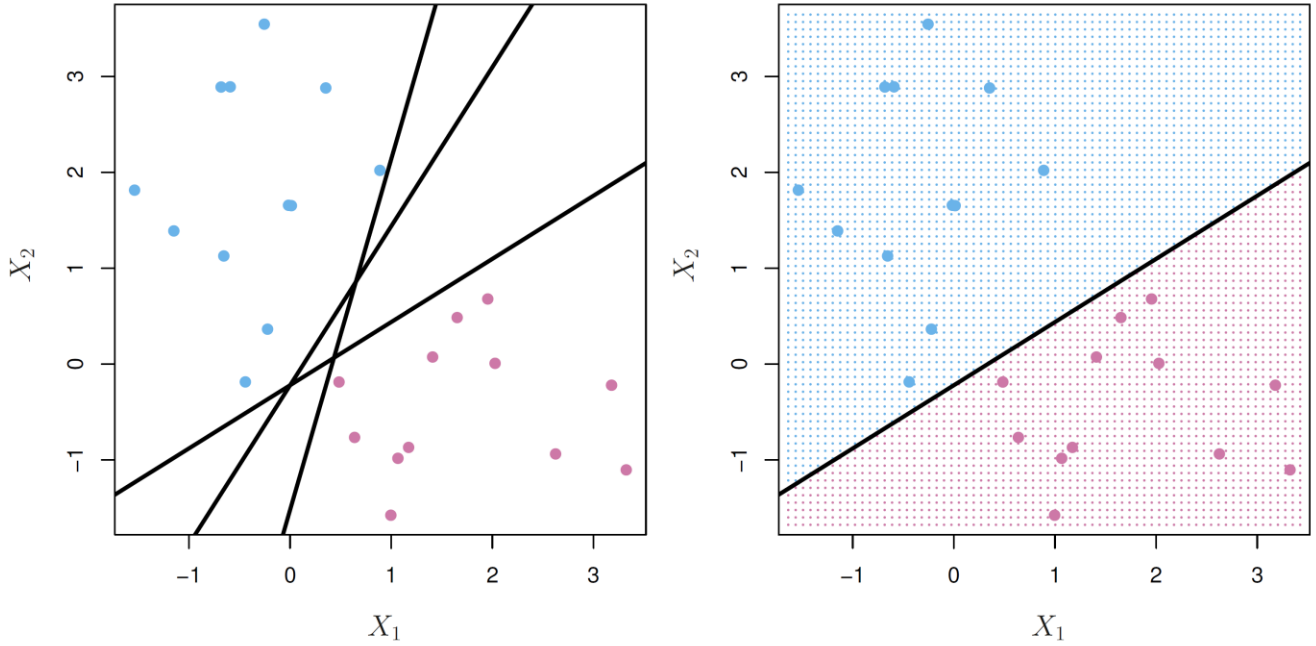
- If $p = 2$ dimensions a hyperplane is a line
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not
- The vector $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane

Hyperplane in 2 Dimensions



Separating Hyperplanes

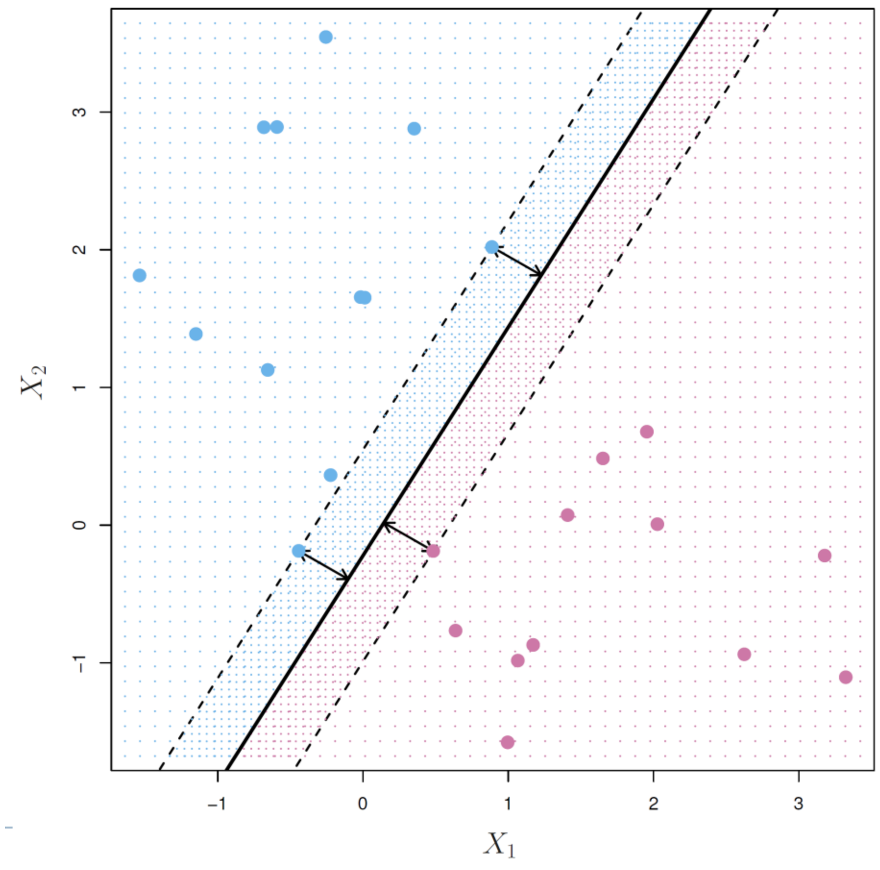
- If $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for purple, then if $Y_i \cdot f(X_i) = +1$ for all i, $f(X) = 0$ defines a separating hyperplane



1. Maximal Margin Classifier

- Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes

Note the distance between point (x_0, y_0) to line $ax + by + c = 0$ is $\frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$



- Constrained optimization problem

$$\max_{\beta_0,\beta_1,\ldots,\beta_p} M$$

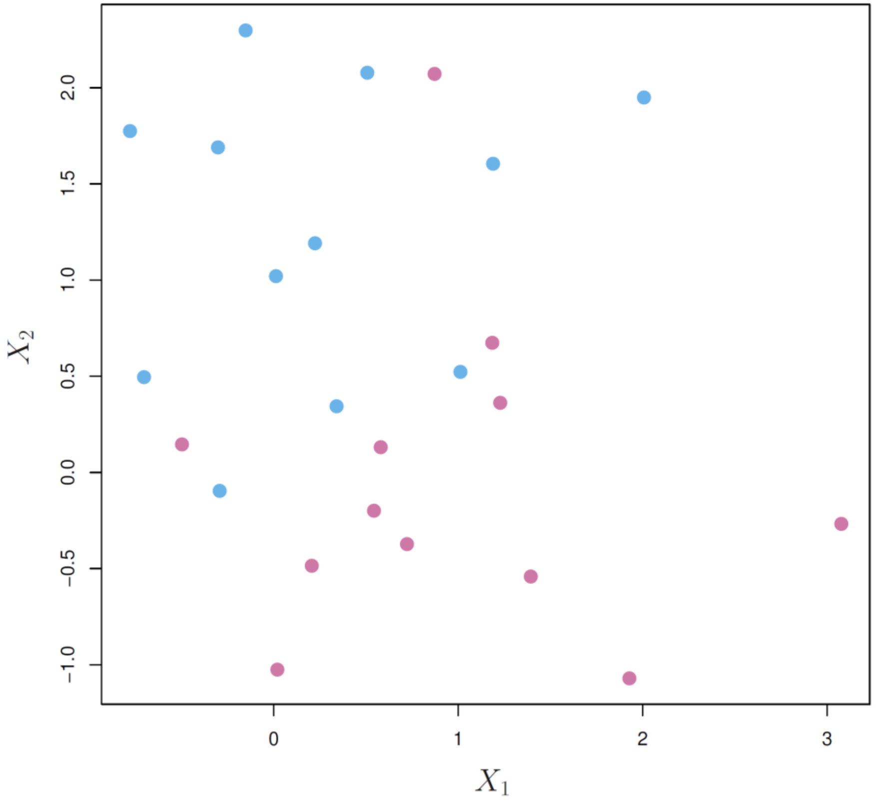
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M$$

$$\text{for all } i = 1, \ldots, N$$

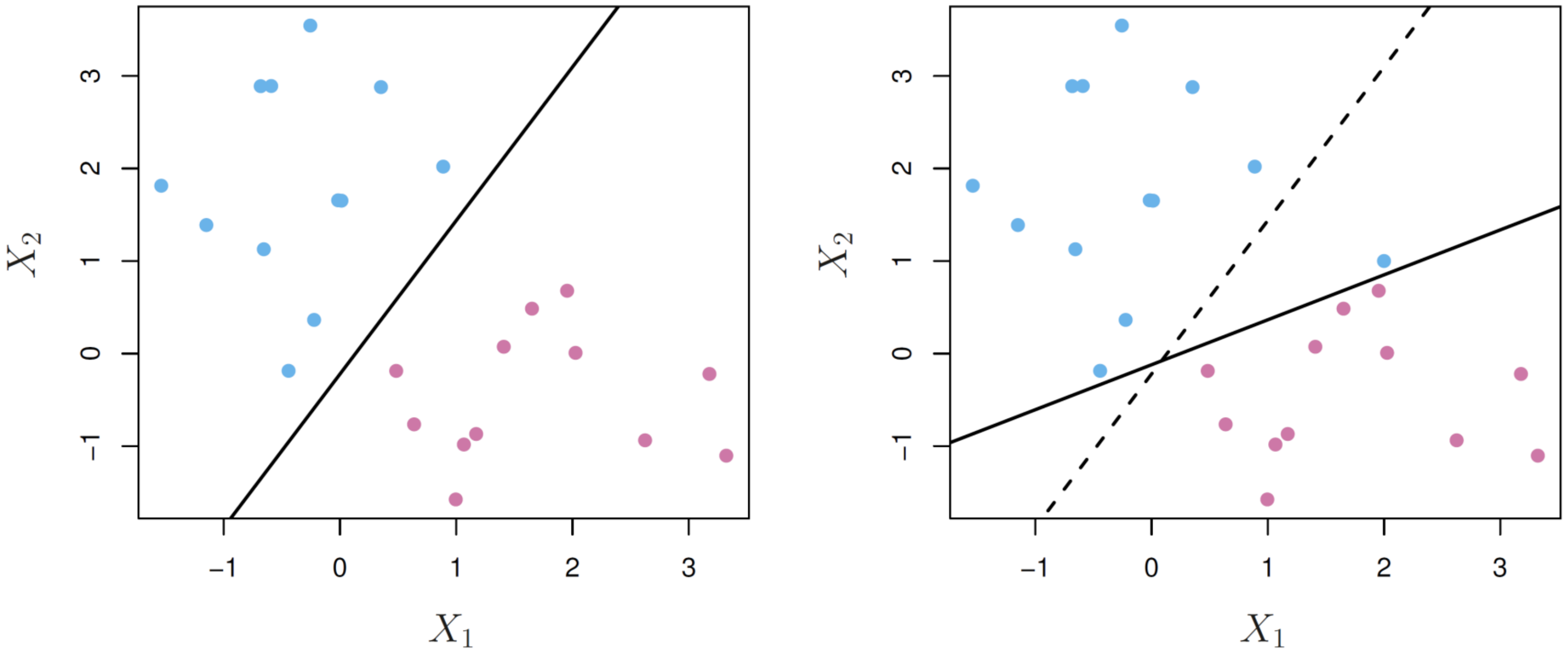
Non-separable Data

- The data below are not separable by a linear boundary
- This is often the case, unless n $< p$

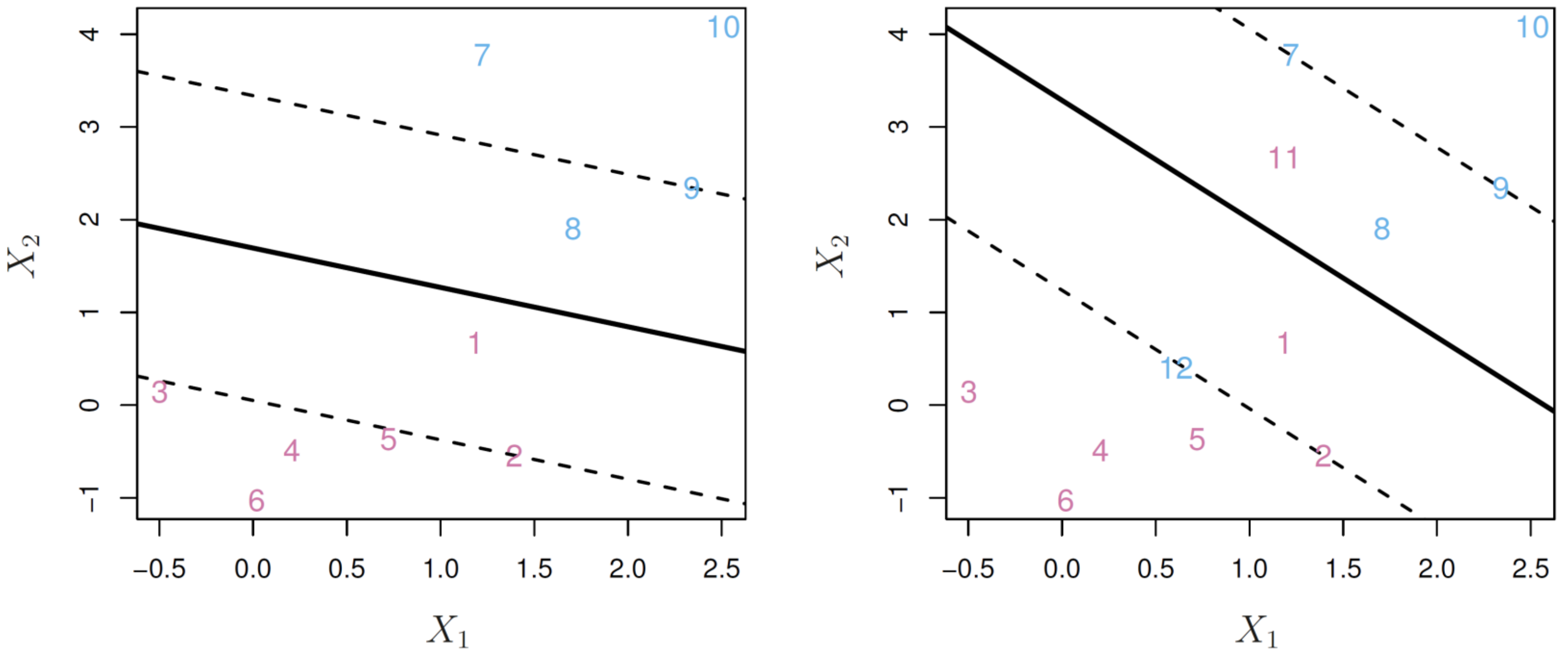


Noisy Data

- Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier
- The support vector classifier maximizes a soft margin



2. Support Vector Classifier (SVC)



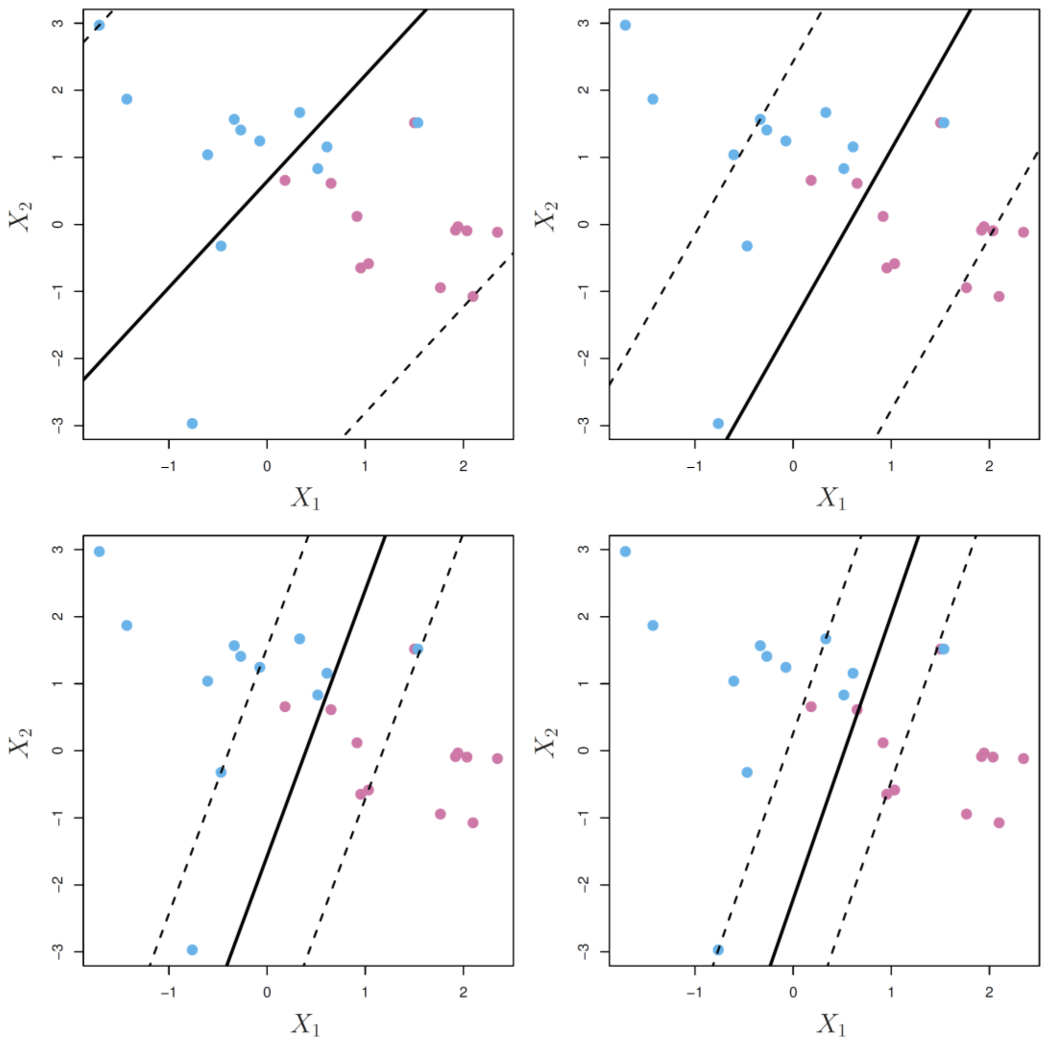
$$\max_{\beta_0,\beta_1,\ldots,\beta_p} M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

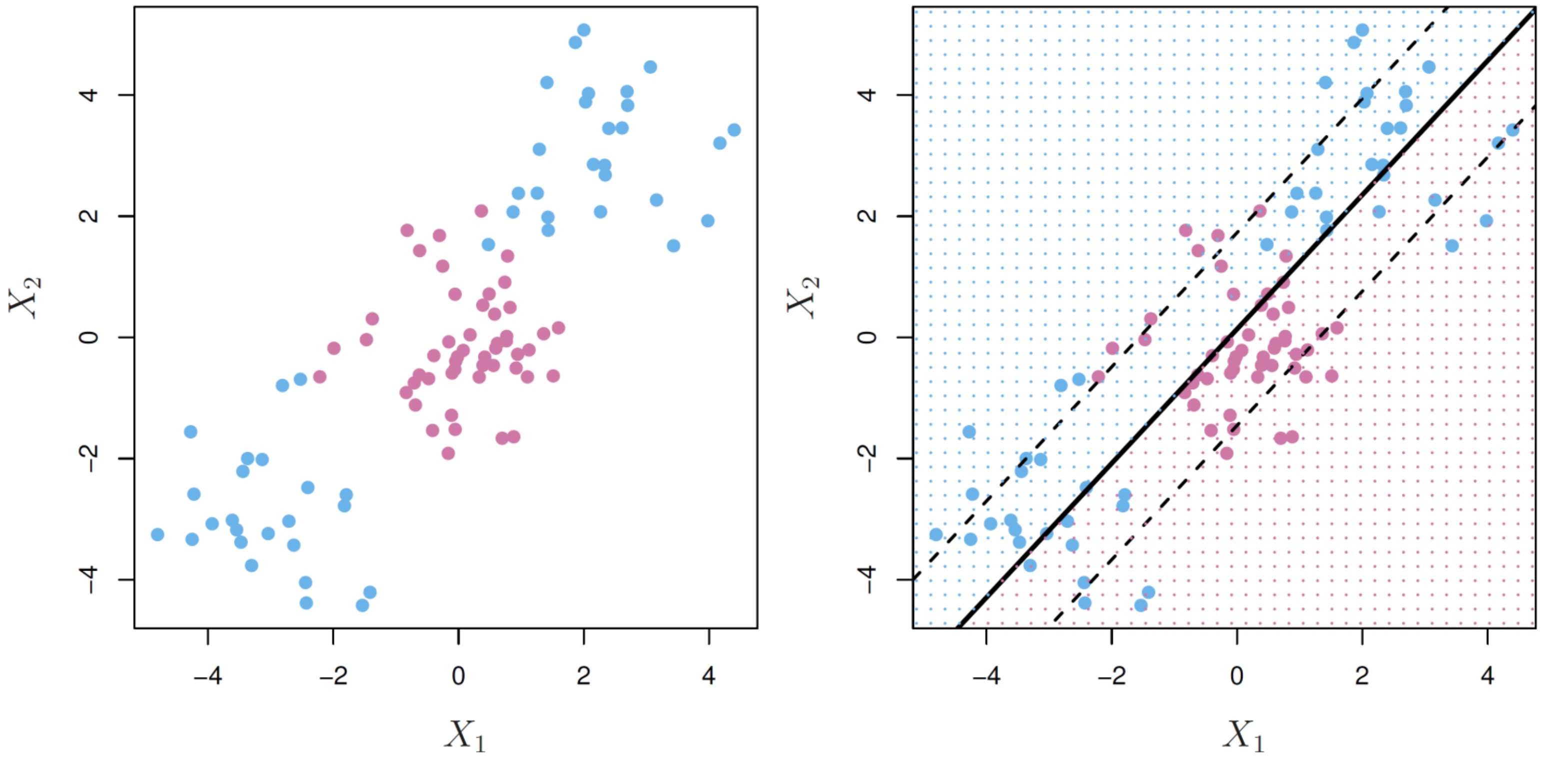
$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq const, \text{ for all } i = 1, \ldots, N$$

C is a regularization parameter



Linear boundary can fail

- Sometime a linear boundary simply won't work, no matter what value of C
- The example below is such a case
- What to do?



Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2, X_1^3, X_1X_2, X_1X_2^3, \ldots$. Hence go from a p-dimensional space to a $M > p$ dimensional space
- Fit a support-vector classifier in the enlarged space
- This results in non-linear decision boundaries in the original space
- Example: Suppose we use $(X_1X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1X_2). Then the decision boundary would be of the form

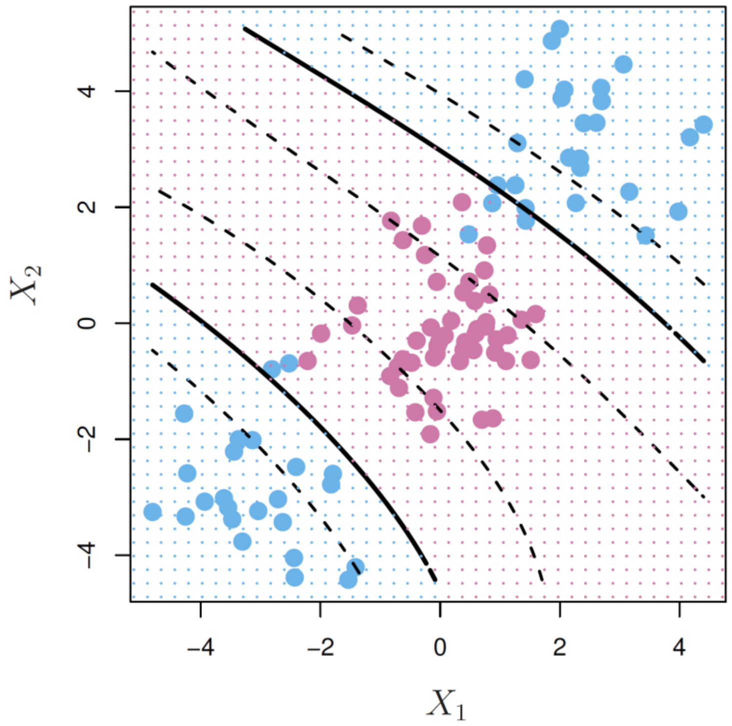
$$\beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_1^2 + \beta_4X_2^2 + \beta_5X_1X_2 = 0$$

  - This leads to nonlinear decision boundaries in the original space (quadratic conic sections)

Cubic Polynomials

- Here we use a basis expansion of cubic polynomials. From 2 variables to 9
- The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 + \beta_7 X_2^3 + \beta_8 X_1 X_2^2 + \beta_9 X_1^2 X_2 = 0$$



Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of kernels
- Main idea: feature mapping to a high dimensional space

$$x \rightarrow \Phi(x) = (\varphi_1(x), \varphi_2(x), \varphi_3(x), \dots), \Phi(x)'\Phi(x) = K(x, u)$$

Kernel Trick: We do not really need to know $\Phi(x)$. Instead, we work on the Kernel

- Before we discuss these, we must understand the role of inner products in support-vector classifiers

Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

- The linear support vector classifier can be represented as (n parameters)

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

- To estimate the parameters $\alpha_1, \dots, \alpha_n$ and β_0, all we need are the $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.
  - It turns out that most of the $\hat{\alpha}_i$ can be zero (Points x_i associated with nonzero $\hat{\alpha}_i$ are called support vectors (SVs)):

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i y_i \langle x, x_i \rangle$$

S is the support set of indices i such that $\hat{\alpha}_i > 0$

Computing the Support Vector Classifier

$$\max_{\beta_0,\beta,|\beta|=1} M$$

$$y_i\left(x_i^T\beta+\beta_0\right)\geq M(1-\epsilon_i),$$

$$\epsilon_i\geq 0, \sum_{i=1}^{n}\epsilon_i\leq const, \text{ for all } i=1,\ldots,N$$

Is equivalent to

$$\max_{\beta_0,\beta} M$$

$$\frac{1}{|\beta|}y_i\left(x_i^T\beta+\beta_0\right)\geq M(1-\epsilon_i),$$

$$\epsilon_i\geq 0, \sum_{i=1}^{n}\epsilon_i\leq const, \text{ for all } i=1,\ldots,N$$

Let $M=1/|\beta|$

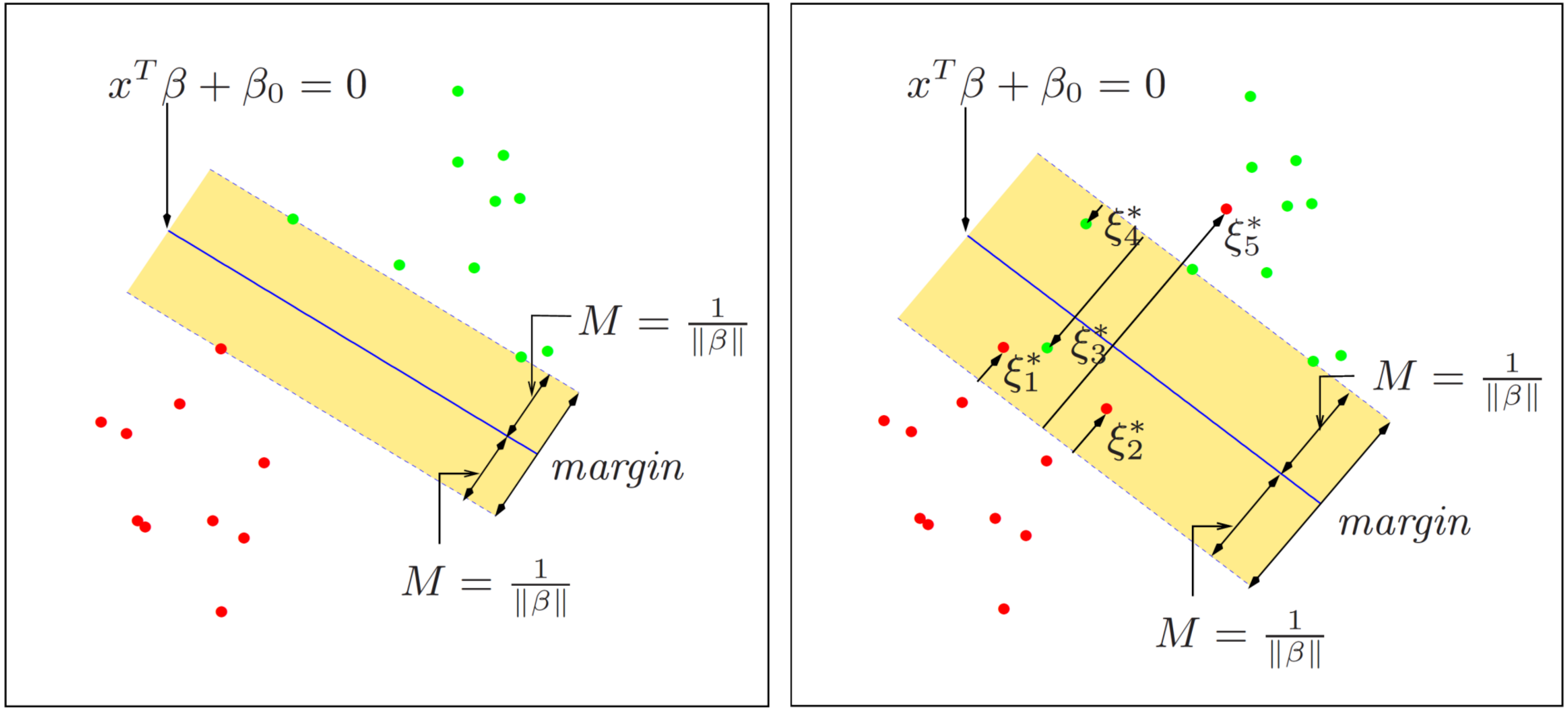
$$\min_{\beta_0,\beta} |\beta|$$

$$y_i\left(x_i^T\beta+\beta_0\right)\geq (1-\epsilon_i),$$

$$\epsilon_i\geq 0, \sum_{i=1}^{n}\epsilon_i\leq const, \text{ for all } i=1,\ldots,N$$

Computing the Support Vector Classifier

- $\sum_{i=1}^{n} \epsilon_i \leq C$



Computing the Support Vector Classifier (Assume $\epsilon_i = 0$)

- The Lagrange (primal) problem is (suitable for small p) to minimizing

$$L_P = \frac{1}{2}|\beta|^2$$

Subject to $y_i\left(x_i^T\beta + \beta_0\right) \geq 1$

$p + 1$ variables, n constraints

- The Lagrangian (dual) problem is (suitable for small n, can use the kernel trick) to maximizing

$$L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i'=1}^{n}\alpha_i\alpha_{i'}\, y_i y_{i'} x_i^T x_i$$

Subject to $0 \leq \alpha_i$ and $\sum_{i=1}^{n}\alpha_i y_i = 0$

n variables, $n + 1$ constraints

Computing the Support Vector Classifier

- The Lagrange (primal) problem is to minimizing

$$L_P = \frac{1}{2}|\beta|^2 + C\sum_{i=1}^{n} \epsilon_i$$

Subject to $\epsilon_i \geq 0, y_i\left(x_i^T\beta + \beta_0\right) \geq 1 - \epsilon_i$

$p + 1 + n$ variables, $2n$ constraints

- The Lagrangian (dual) problem is to maximizing

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

Subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

n variables, $2n + 1$ constraints

- Note the solution has the form $\hat{\beta} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i$

3. Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- The Support Vector Machines is an extension of SVC using kernels
- Some special kernel functions can do this for us. E.g.

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

computes the inner-products needed for d dimensional polynomials - $\binom{p+d}{d}$ basis functions!

- The solution has the form

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i y_i K\langle x, x_i \rangle$$

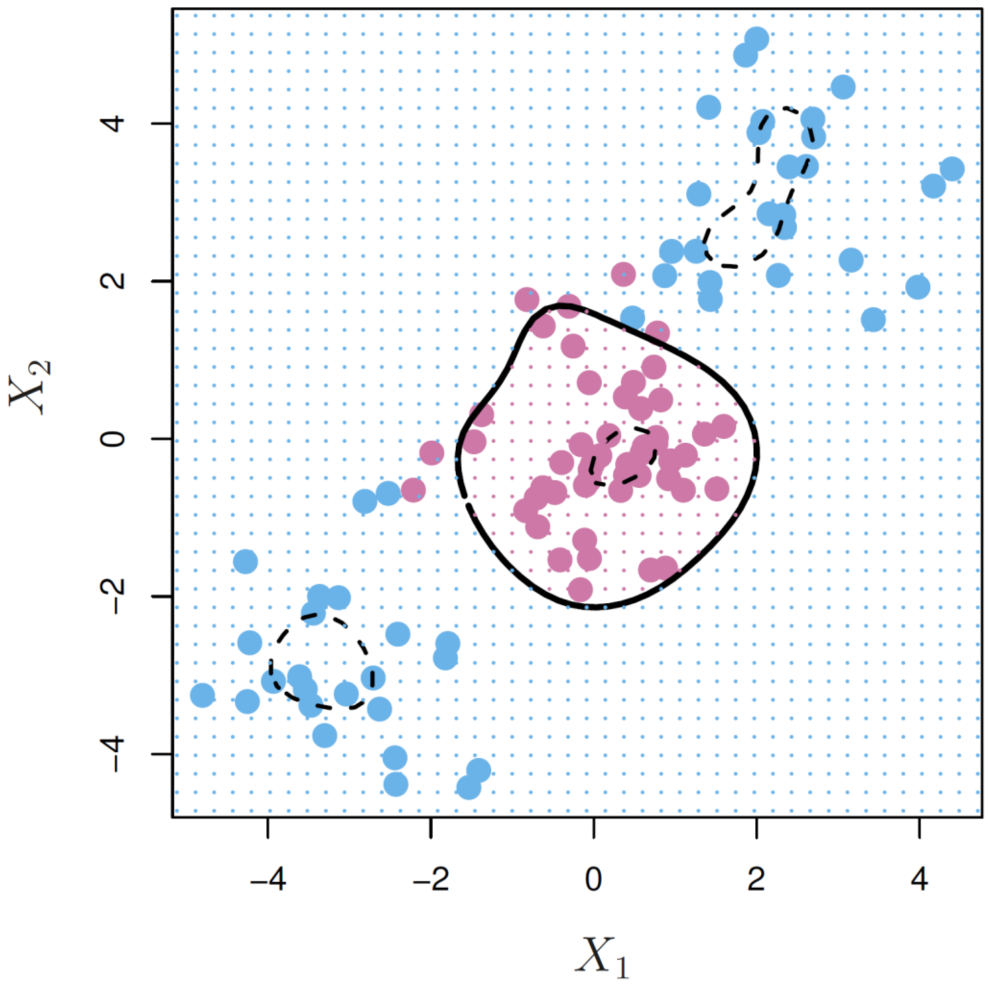
Radial Kernel

- Implicit feature space; very high dimensional
- Controls variance by squashing down most dimensions severely

$$K\langle x, x_i\rangle = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

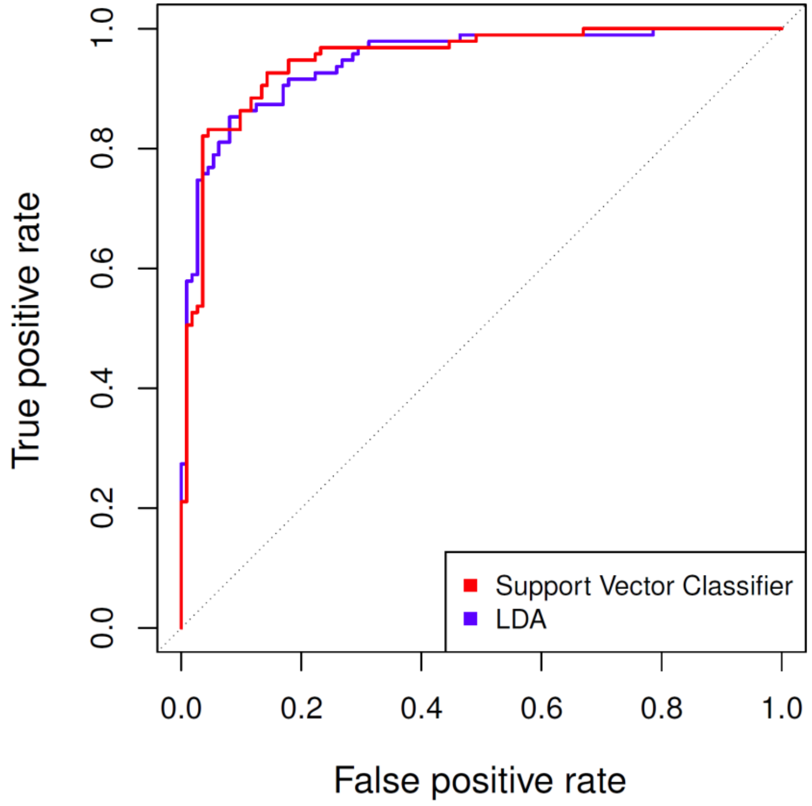
$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i y_i K\langle x, x_i\rangle$$

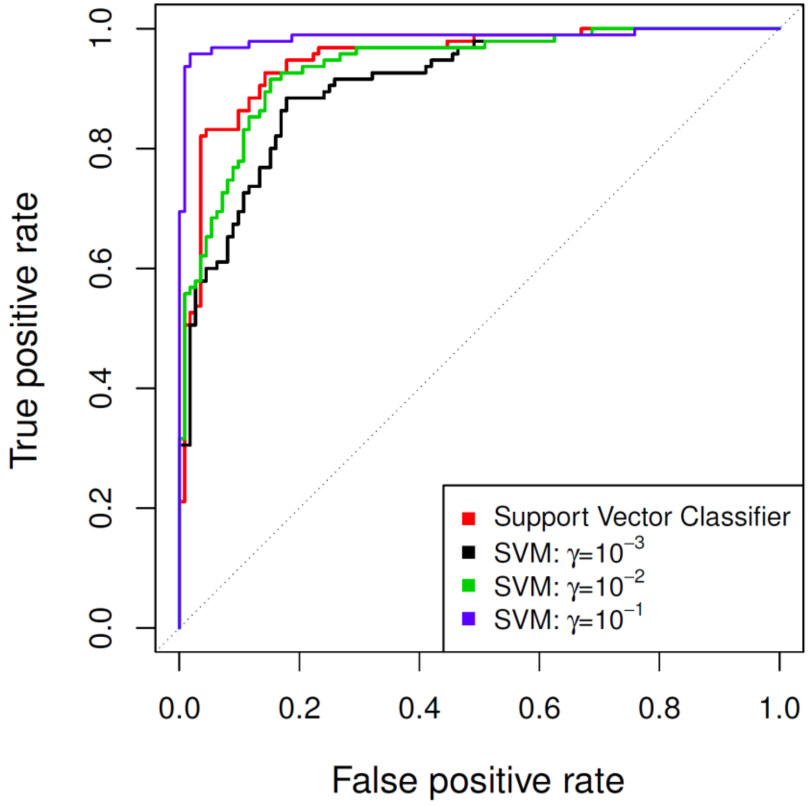
- γ is also a regularization parameter (You should reduce it if overfitting)



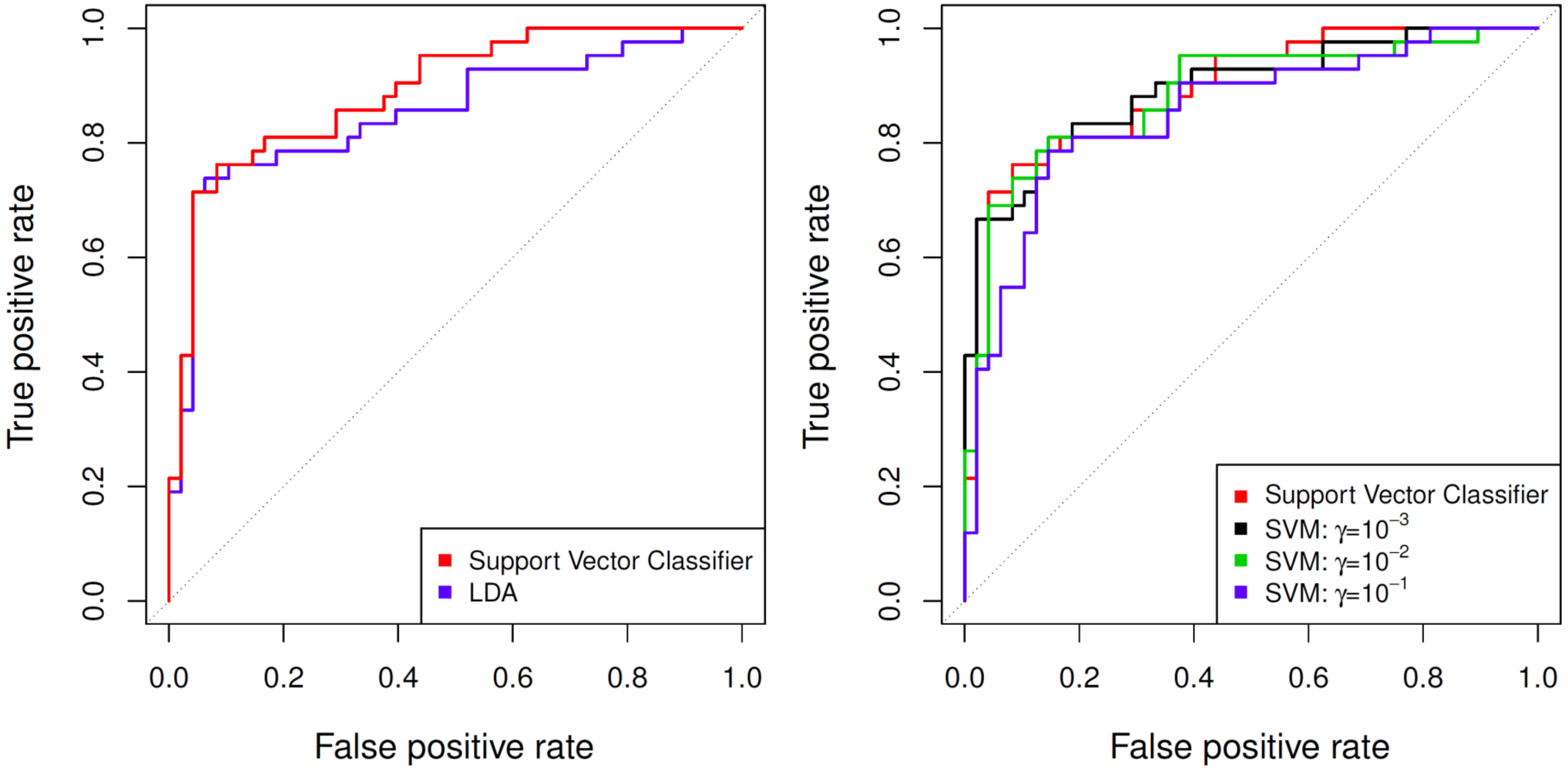
Example: Heart Data

- ROC curve is obtained by changing the threshold 0 to threshold t in $\hat{f}(X) > t$, and recording false positive and true positive rates as t varies. Here we see ROC curves on training data





Example continued: Heart Test Data



SVMs: more than 2 classes?

- The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?
  - OVA One versus All. Fit K different 2-class SVM classifiers $\hat{f}(x),\ k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}(x^*)$, is largest.
  - OVO One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{kl}(x)$. Classify x^* to the class that wins the most pairwise competitions

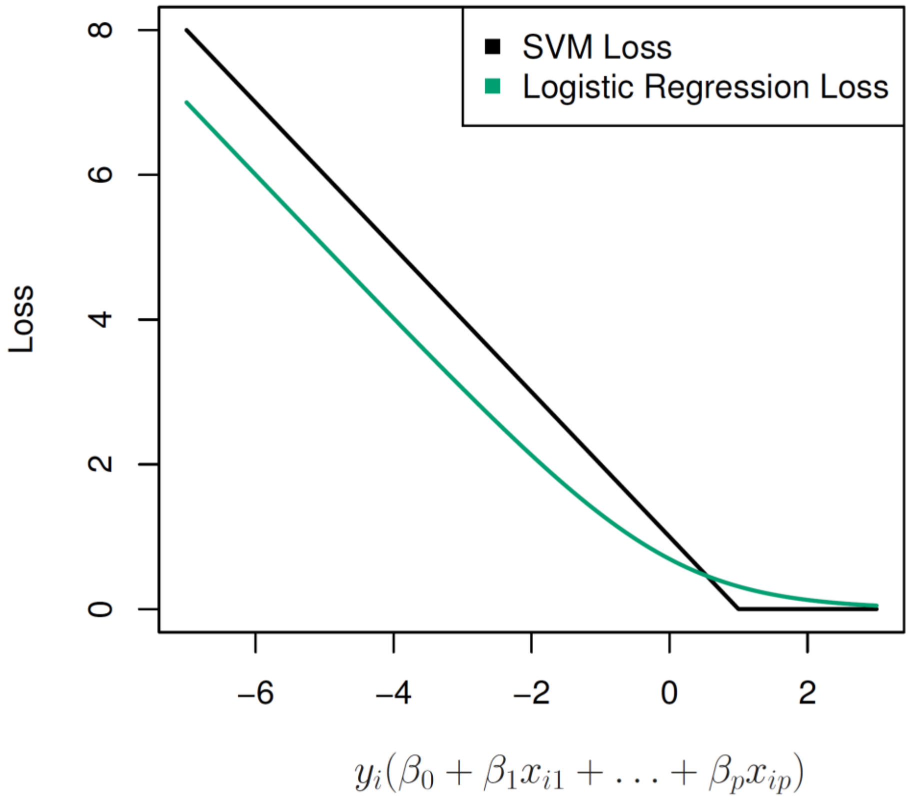
Which to choose? If K is not too large, use OVO

Support Vector versus Logistic Regression?

- With $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$ can rephrase support-vector classifier optimization as

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} \left\{ \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

  - This has the form loss plus penalty. The loss is known as the hinge loss
  - Very similar to “loss” in logistic regression (negative log-likelihood)



Which to use: SVM or Logistic Regression?

- When classes are (nearly) separable, SVM does better than LR. So does LDA
- When not, LR (with ridge penalty) and SVM very similar
- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive

Computations

- Support Vector Machine algorithms are not scale invariant, so it is highly recommended to scale your data
- For the linear case, the algorithm used in *LinearSVC* by the *liblinear* implementation. Note that *LinearSVC* does not accept parameter kernel, as this is assumed to be linear
- *SVC* (relies on *libsvm*)and *NuSVC* are similar methods, but accept slightly different sets of parameters
- *liblinear and libsvm are* efficient library developed by Chih-Jen Lin

Computations

- C is 1 by default and it's a reasonable default choice. If you have a lot of noisy observations you should decrease it: decreasing C corresponds to more regularization
- For large-scaled problem, try to use *SGDClassifier* with hinge loss

SVM

- The key features of SVMs are the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by optimizing the margin

- SVM can also be extended to regression, density estimation and novelty detection problem

Appendix

Reference

- https://www.csie.ntu.edu.tw/~htlin/mooc/
- https://scikit-learn.org/stable/modules/svm.html

Computing the support vector machine

- The problem above is quadratic with linear inequality constraints, hence it is a convex optimization problem. We describe a quadratic programming solution using Lagrange multipliers

Statistical Modeling Y ~ X:

- Y continuous, X continuous: Regression Analysis
- Y discrete, X continuous:
  - X is Gaussian: Linear Discriminant Analysis
  - X is non-Gaussian: Logistic Regression
- Y discrete, X discrete: Discriminant Correspondent Analysis
- Y continuous, X discrete: ANOVA

Perceptron: I only need to know the hyperplane! Why bother a model?

- Perceptron: On-Line learning Classification
  - The hyperplane is sequentially updated by the training set. Calculate the actual output and then update the weight whose increment is proportional to the difference of the actual output and the desired output

- Given an initial w, for each training pair (x_i, y_i) where $1 \le i \le n$, do the two steps:

$$o_i = f(w^T(t)x_i) \text{ (the decision rule)}$$
$$w(t+1) = w(t) + \alpha(o_i - y_i)x_i$$

From Perceptron to SVM

- Introduce the concept of Margin → a unique solution
- Introduce the slack variables → soft margin
- Introduce Kernel method → non-linear feature mapping

Mercer's theorem and common kernels

- Linear kernel
  - Pros: safe, fast (QP solvers), explainable (by *w* and SVs)
  - Cons: restricted (not always separable)
  - A basic tool
- Polynomial kernel
  - Pros: less restricted than linear, strong physical control by knowing the degree
  - Cons: numerical difficulty for large degree
  - Perhaps small degree only
- Gaussian kernel
  - Pros: more powerful than linear and poly ones, bounded (less numerical difficulty than Poly one), only one parameter to be selected
  - Cons: mysterious (no *w*), slower than linear, overfitting problem
  - Popular but be used carefully