SUPPORT VECTOR MACHINES

Support Vector Machines

- *Support vector machine* (SVM), is an approach for classification developed around 90s
- SVMs is versatile approach that has been applied to a variety of settings including binary classification, multi-nominal classification, regression.
- The basic idea is behind SVM is based on separating hyperplanes.

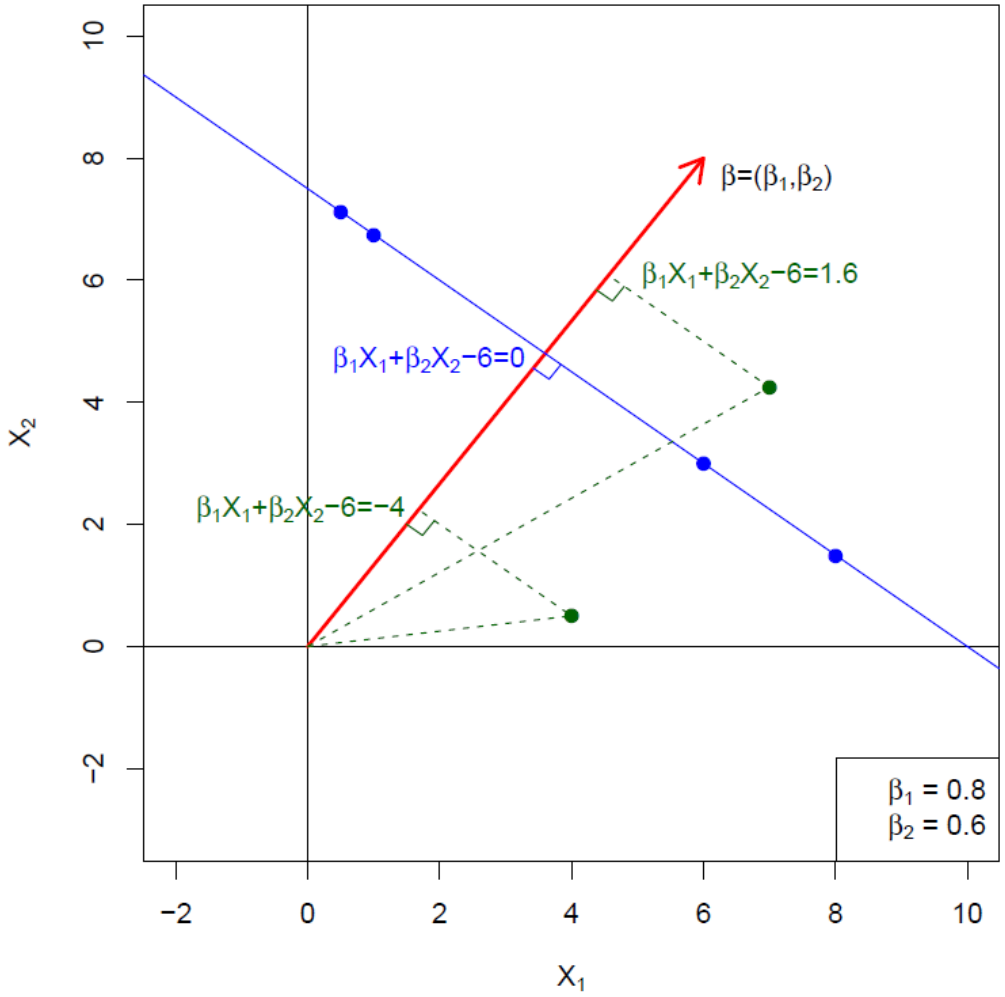
Hyperplanes

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

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

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

Hyperplanes



Separating Hyperplanes

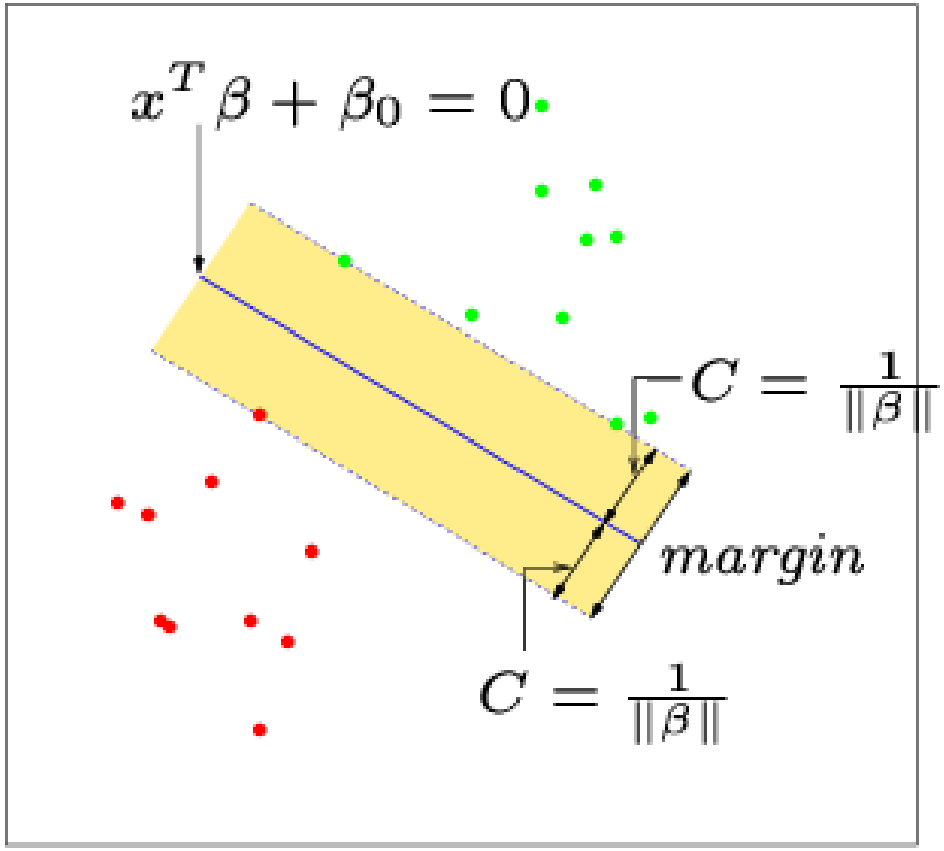
- Here we approach the two-class classification problem in a direct way:
- We try and find a plane that separates the classes in feature space.
- If we cannot, we get creative in two ways:
  - We soften what we mean by “separates”, and
  - We enrich and enlarge the feature space so that separation is possible.

Separating Hyperplanes

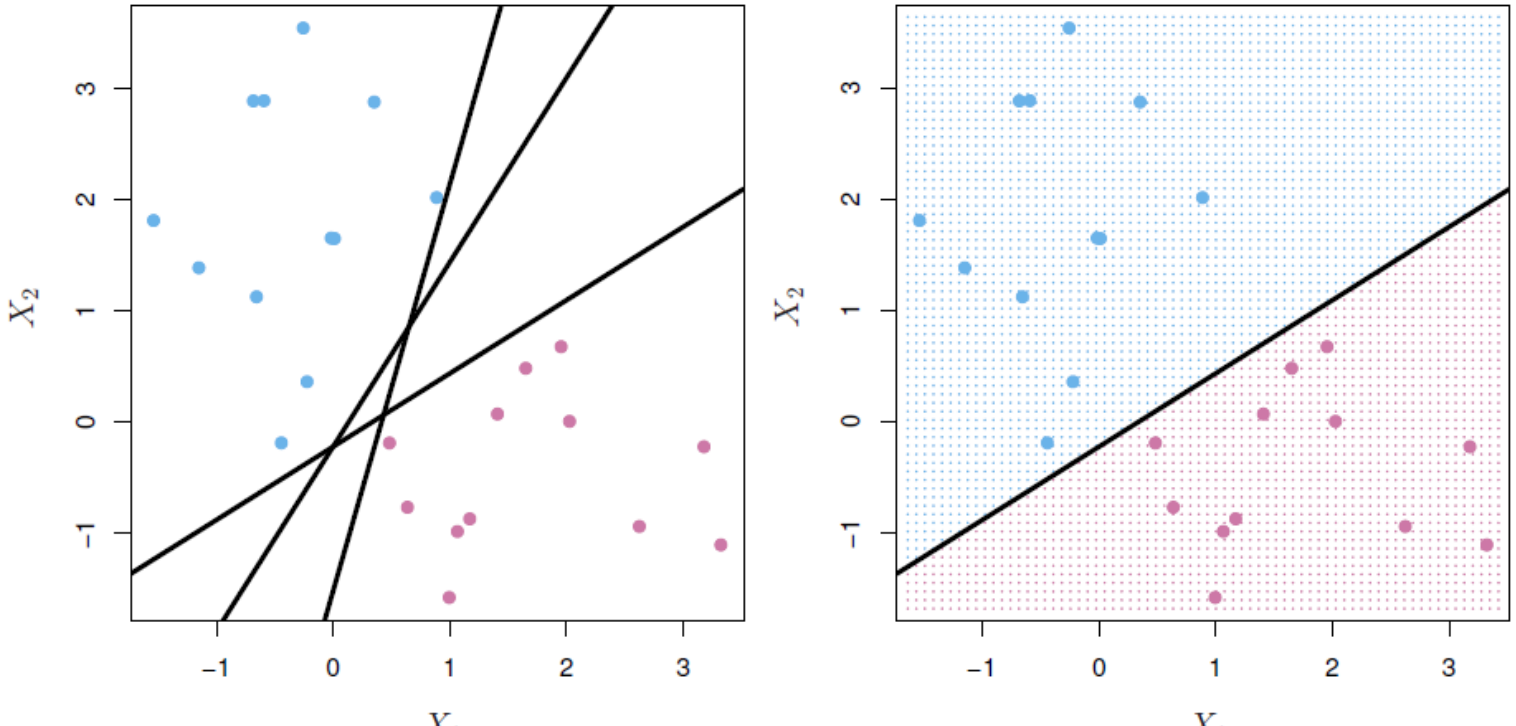
- Imagine a situation where you have a two class classification problem with two predictors X_1 and X_2.
- Suppose that the two classes are “linearly separable” i.e. one can draw a straight line in which all points on one side belong to the first class and points on the other side to the second class.
- Then a natural approach is to find the straight line that gives the biggest separation between the classes i.e. the points are as far from the line as possible
- This is the basic idea of a support vector classifier.

Separating Hyperplanes

- C is the minimum perpendicular distance between each point and the separating line.
- We find the line which maximizes C.
- This line is called the “optimal separating hyperplane”
- The classification of a point depends on which side of the line it falls on.



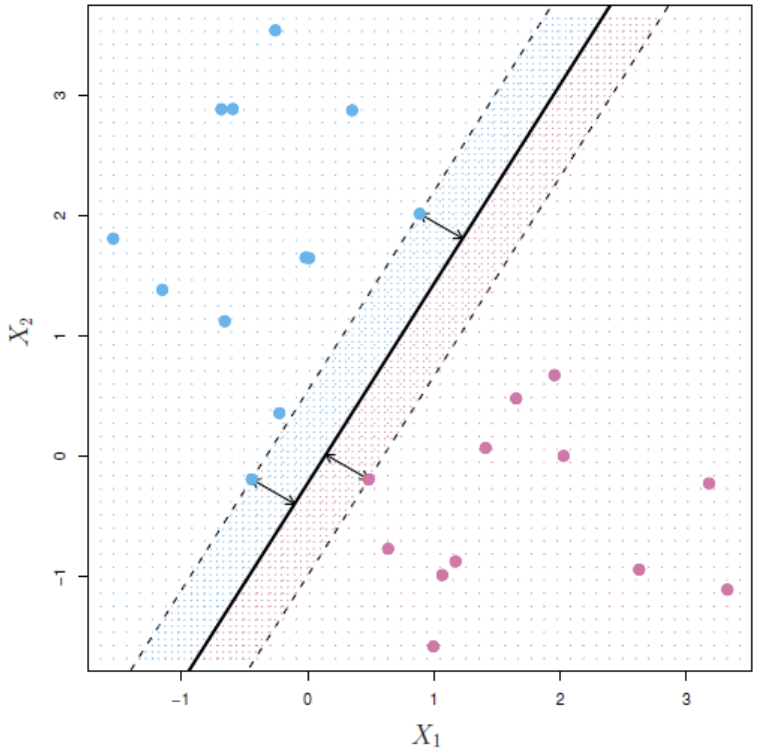
Separating Hyperplanes



- If $f(X) = \beta_0 + \beta_1 X_1 + \quad + \beta_P X_p$, 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 Yi = +1 for blue, say, and Yi =-1 for mauve, then if $Y_i f(X_i) > 0$ for all i, $f(X) = 0$ defines a separating hyperplane.

Maximal Margin Classier

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



Constrained optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

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

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

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

More Than Two Predictors

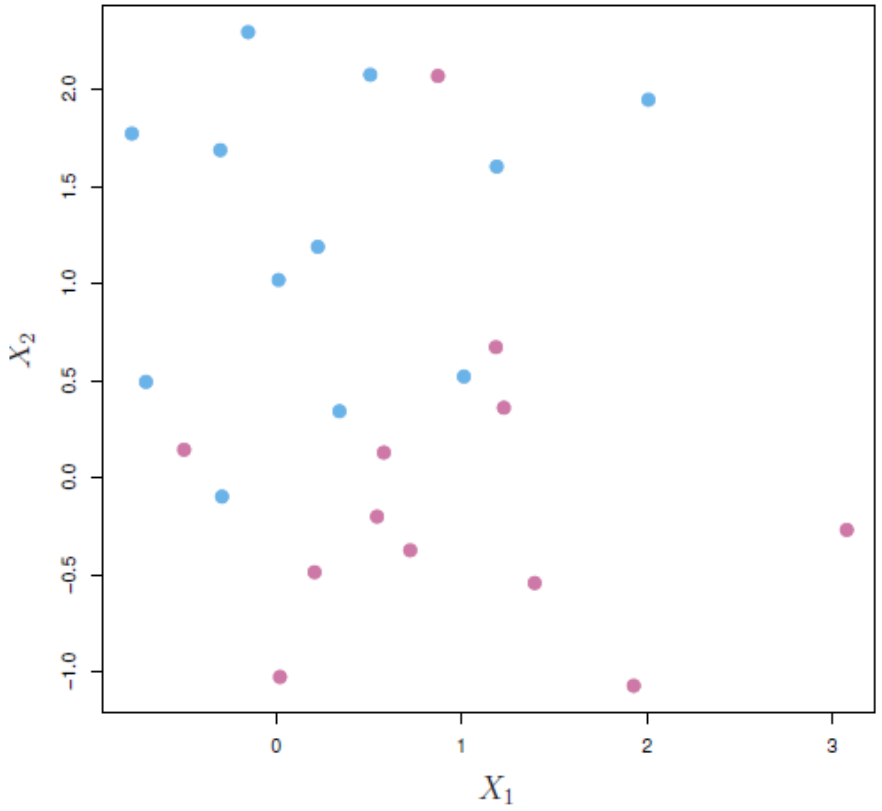
- This idea works just as well with more than two predictors.
- For example, with three predictors you want to find the plane that produces the largest separation between the classes.
- With more than three dimensions it becomes hard to visualize a plane but it still exists. In general they are called hyperplanes.

Non-Separating Classes

- Of course in practice it is not usually possible to find a hyper-plane that perfectly separates two classes.
- In other words, for any straight line or plane that I draw there will always be at least some points on the wrong side of the line.
- That means that the optimization problem above have no feasible solution.

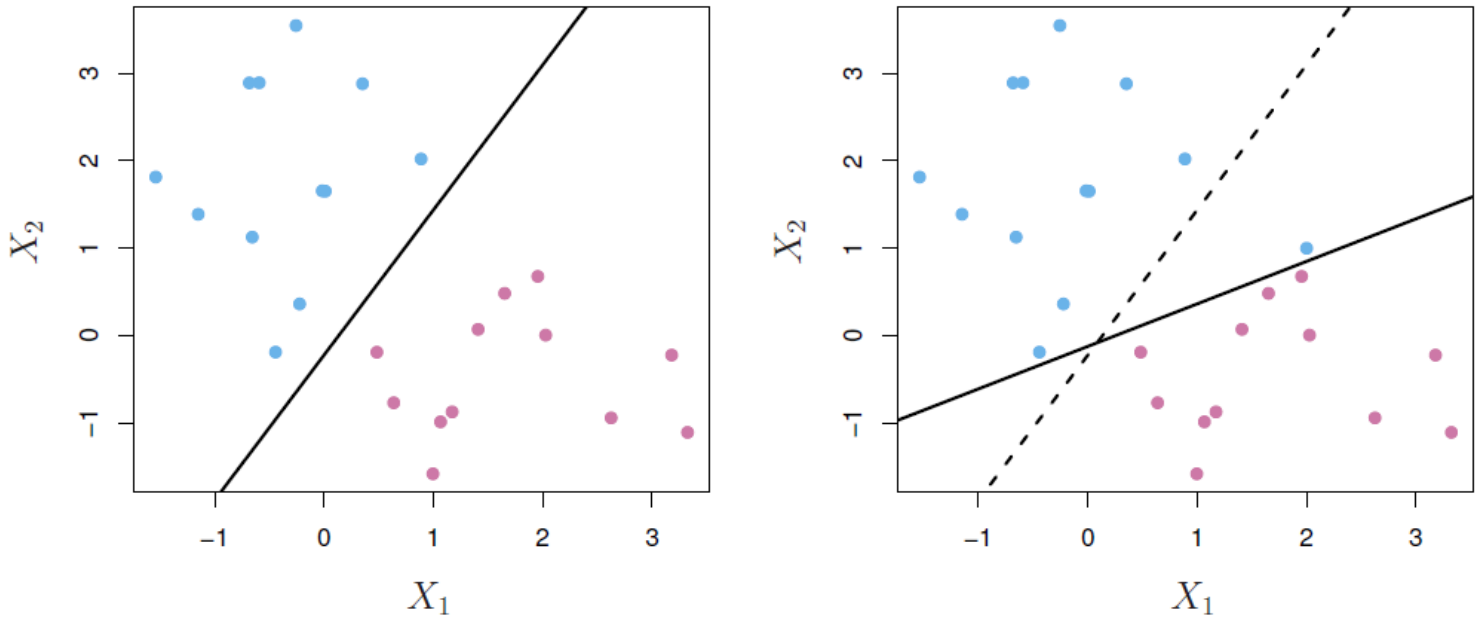
Non-Separable Data

- The data on the right are not separable by a linear boundary.
- This is often the case, unless N < p.
- What to do now?
- Even if a separating hyperplane exists, we might prefer to use something else instead. Why?

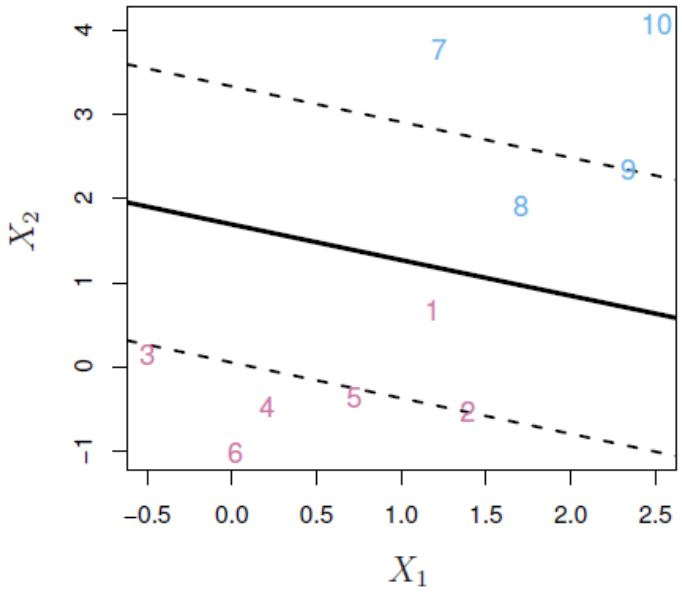


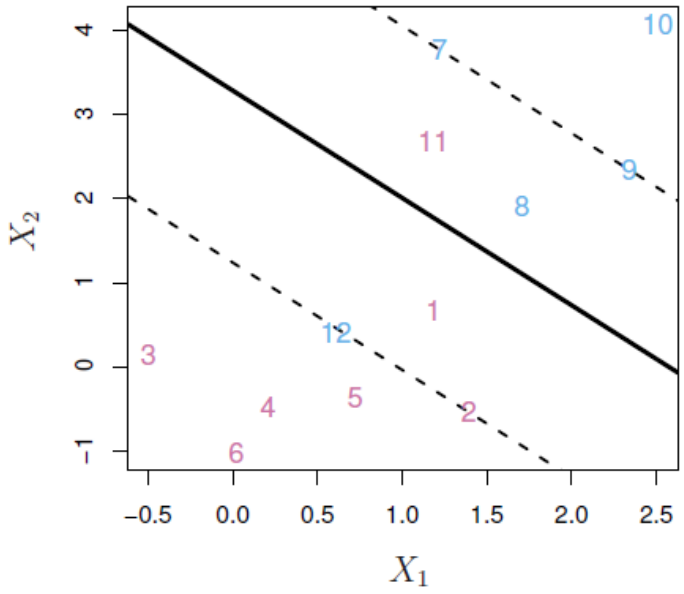
Noisy Data

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



Support Vector Classifier





$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

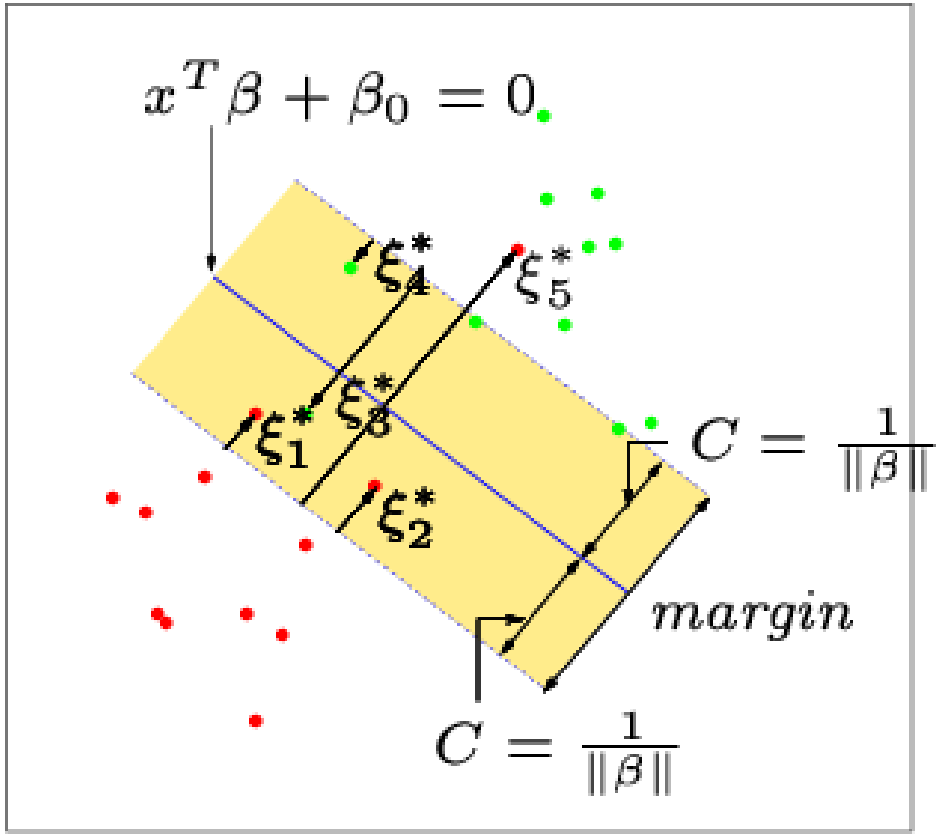
$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

Non-Separating Example

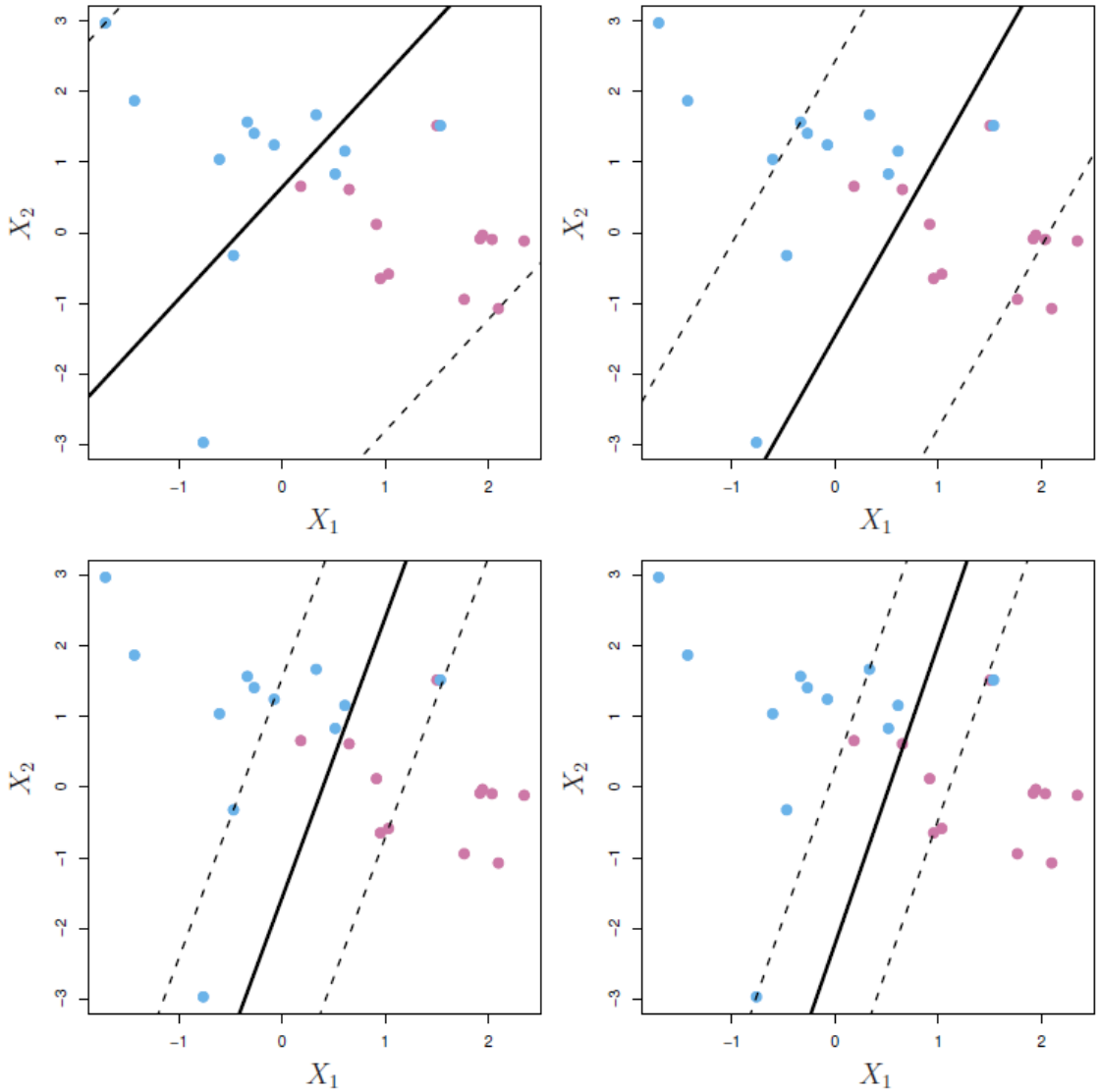
- Let ξ^*_i represent the amount that the i^{th} point is on the wrong side of the margin (the dashed line).
- Then we want to maximize C subject to

$$\frac{1}{C}\sum_{i=1}^{n}\xi_i^* \leq \text{Constant}$$

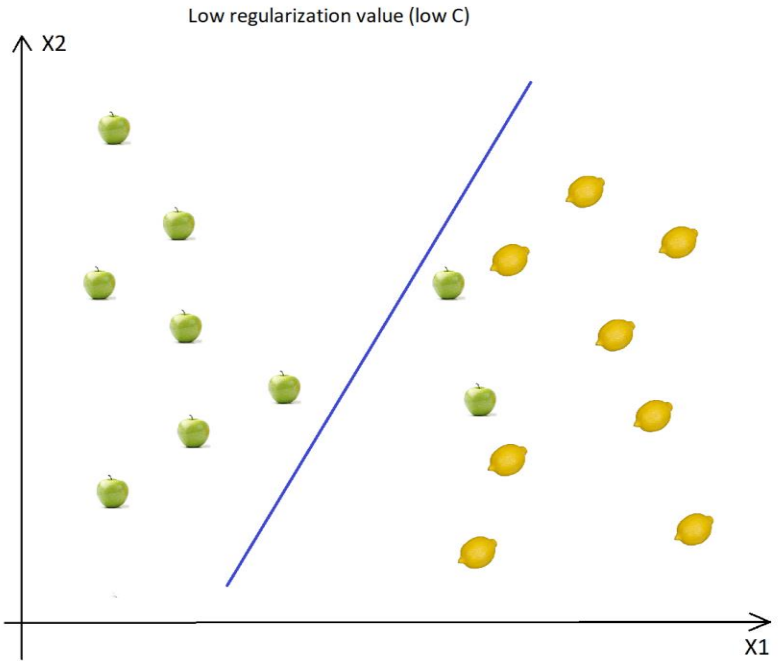
- The constant is a tuning parameter that we choose.

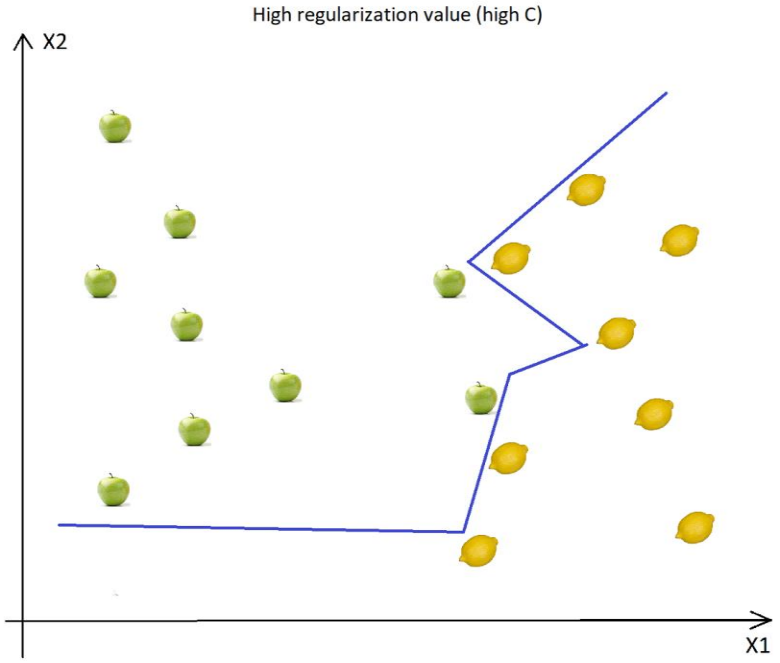


Constant in the Constraint



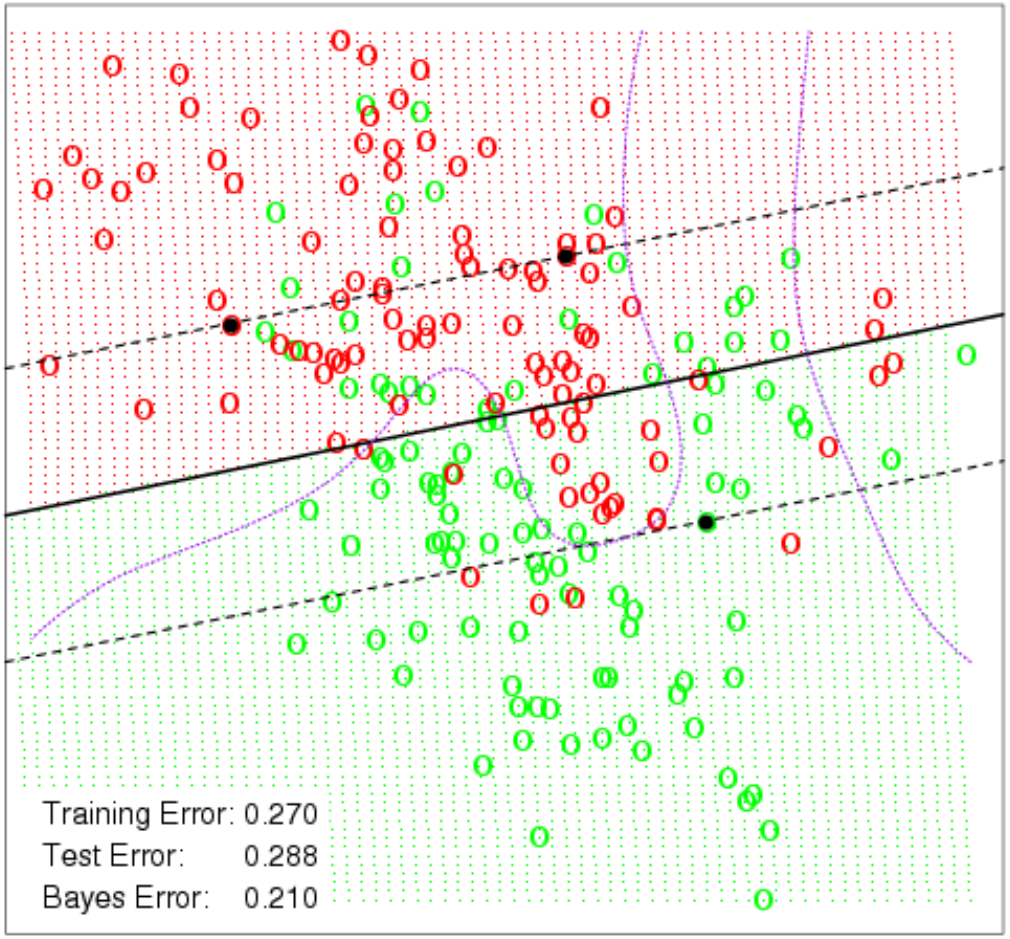
Regularization Parameter





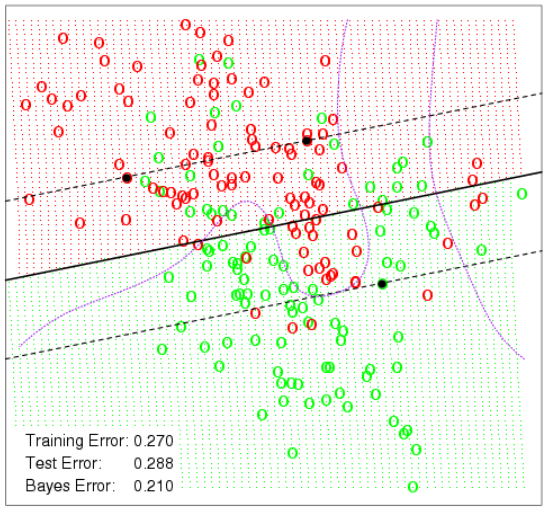
A Simulation Example With A Small Constant

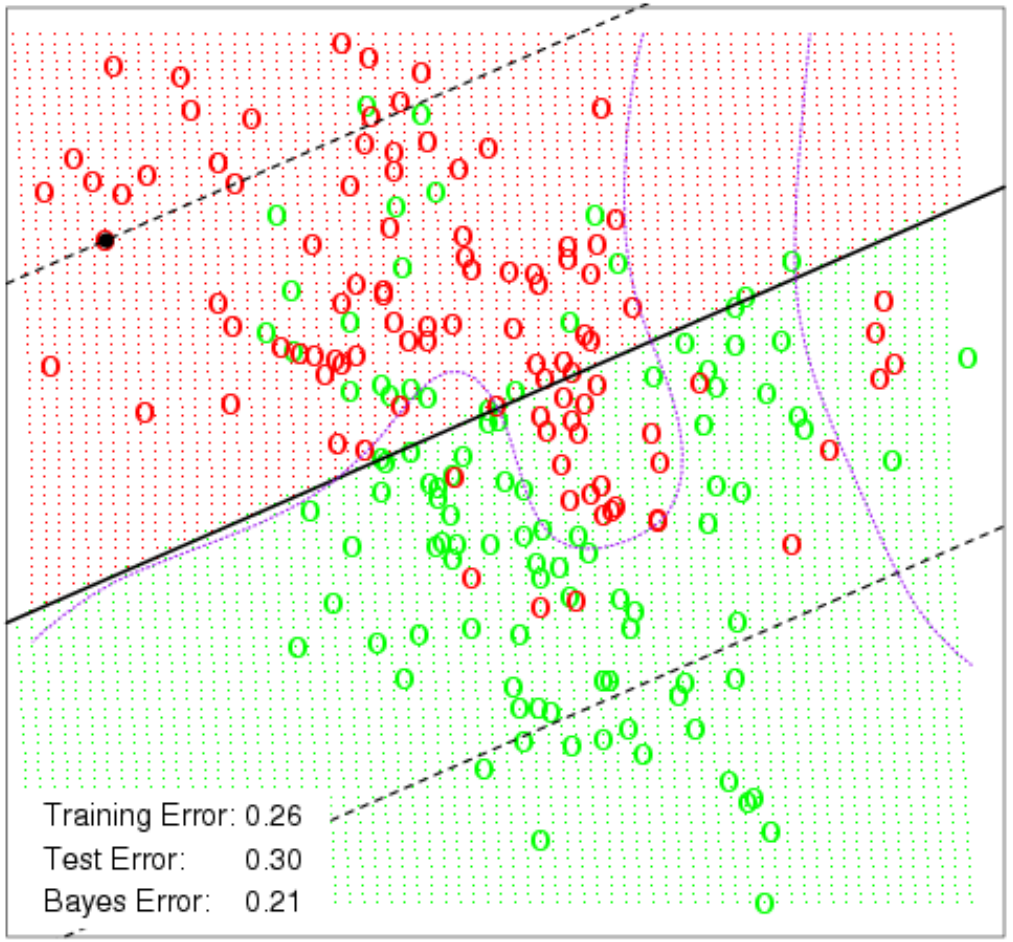
- This is the simulation example with two classes.
- The distance between the dashed lines represents the margin or 2C.
- The purple lines represent the Bayes decision boundaries



The Same Example With A Larger Constant

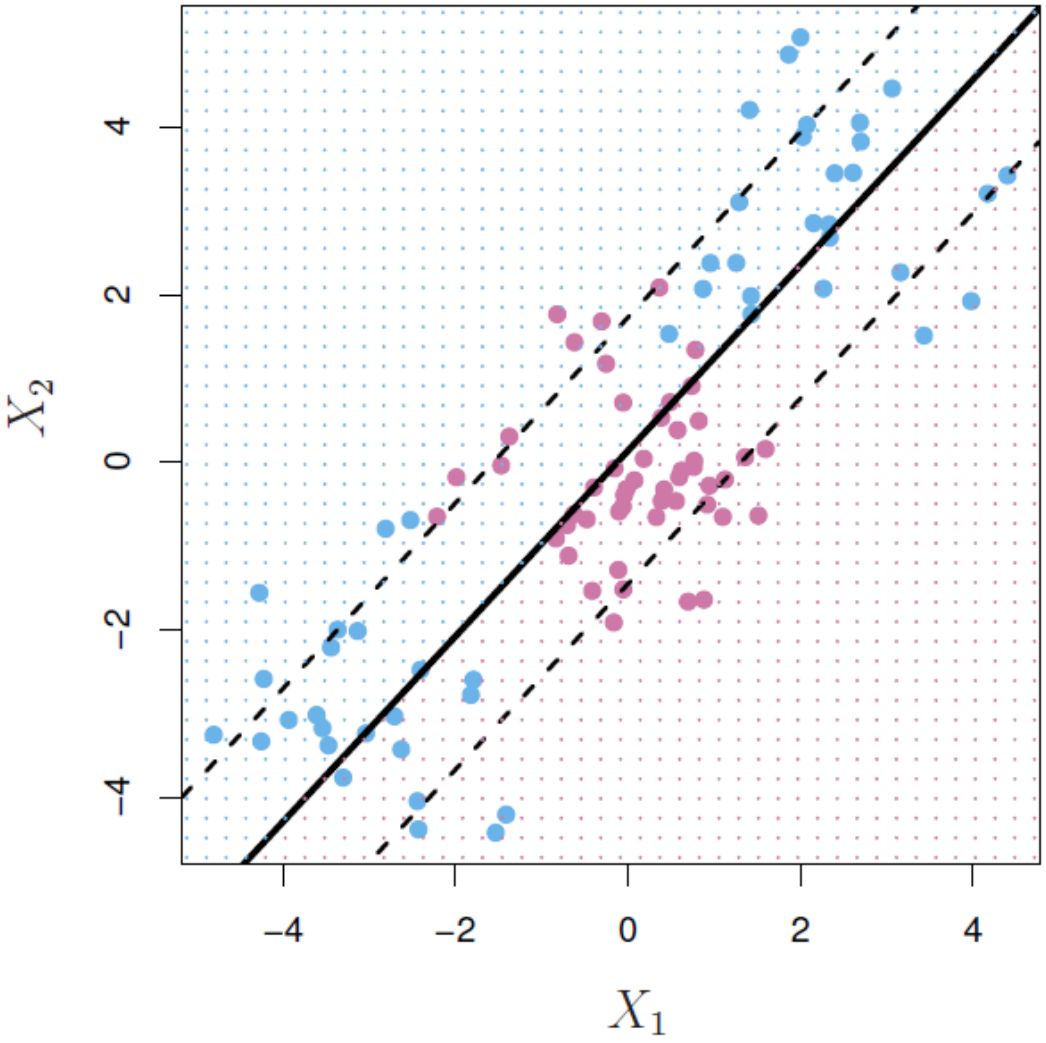
- Using a larger constant allows for a greater margin and creates a slightly different classifier.
- Notice, however, that the decision boundary must always be linear.





Non-Linear Boundary

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



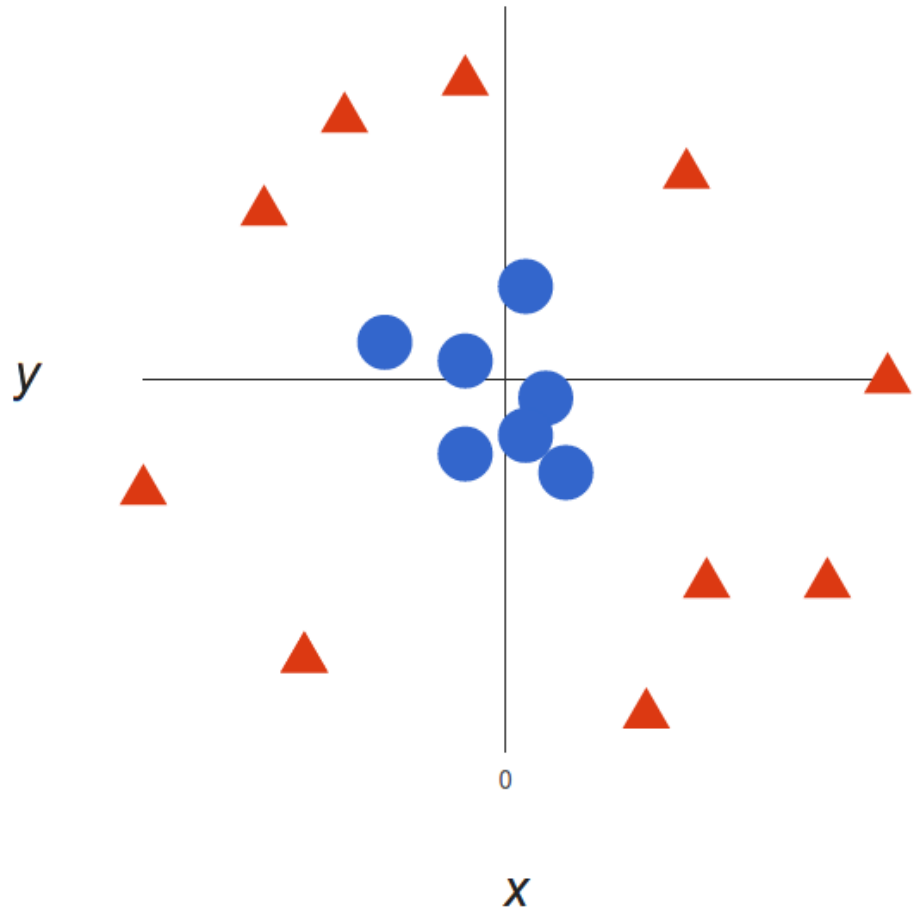
Non-Linear Classifier

- The support vector classifier is fairly easy to think about. However, because it only allows for a linear decision boundary it may not be all that powerful.
- Enlarge the space of features by including transformations;

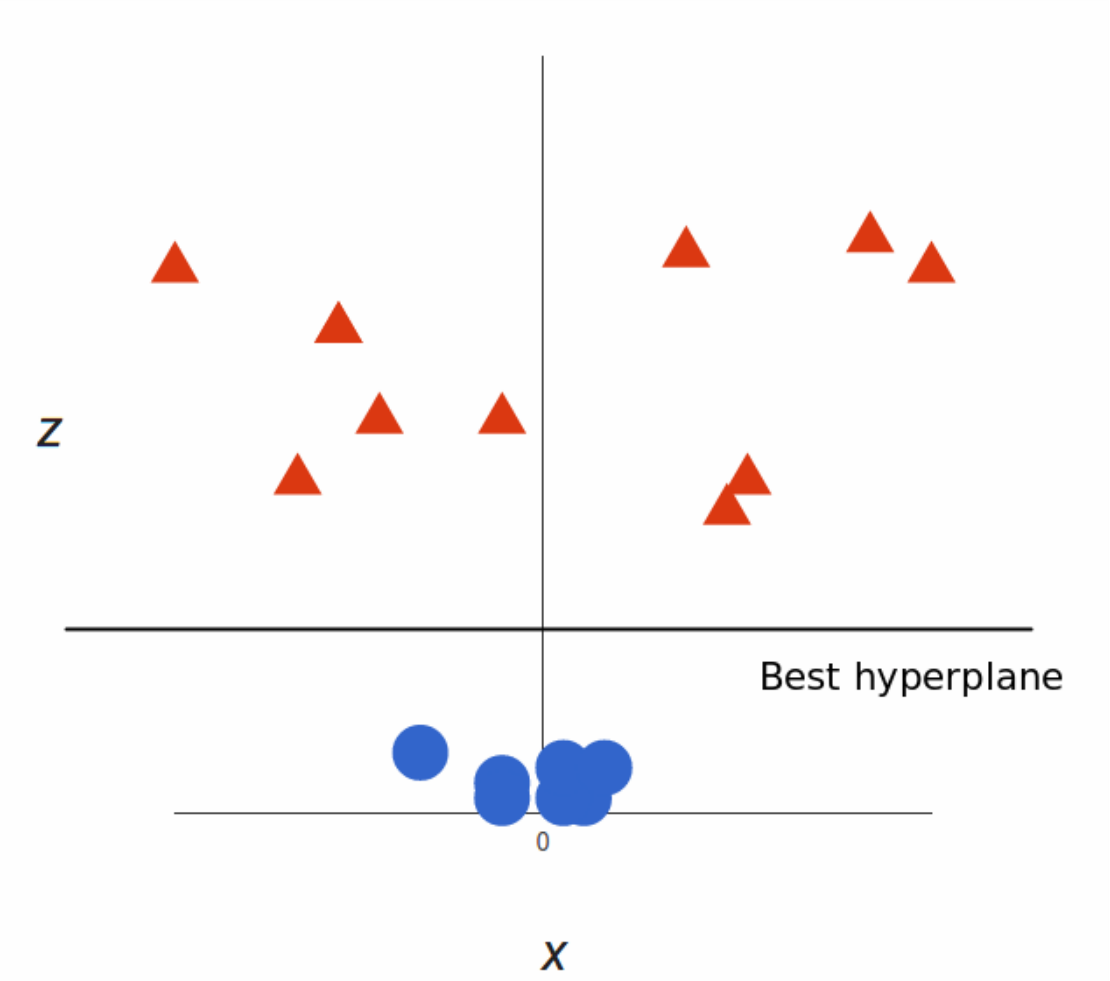
$$X_1^2, X_1^3, X_1X_2, X_1X_2^2, \ldots$$

- Hence go from a p-dimensional space to a M > p dimensional space.
- Fit a support-vector classier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

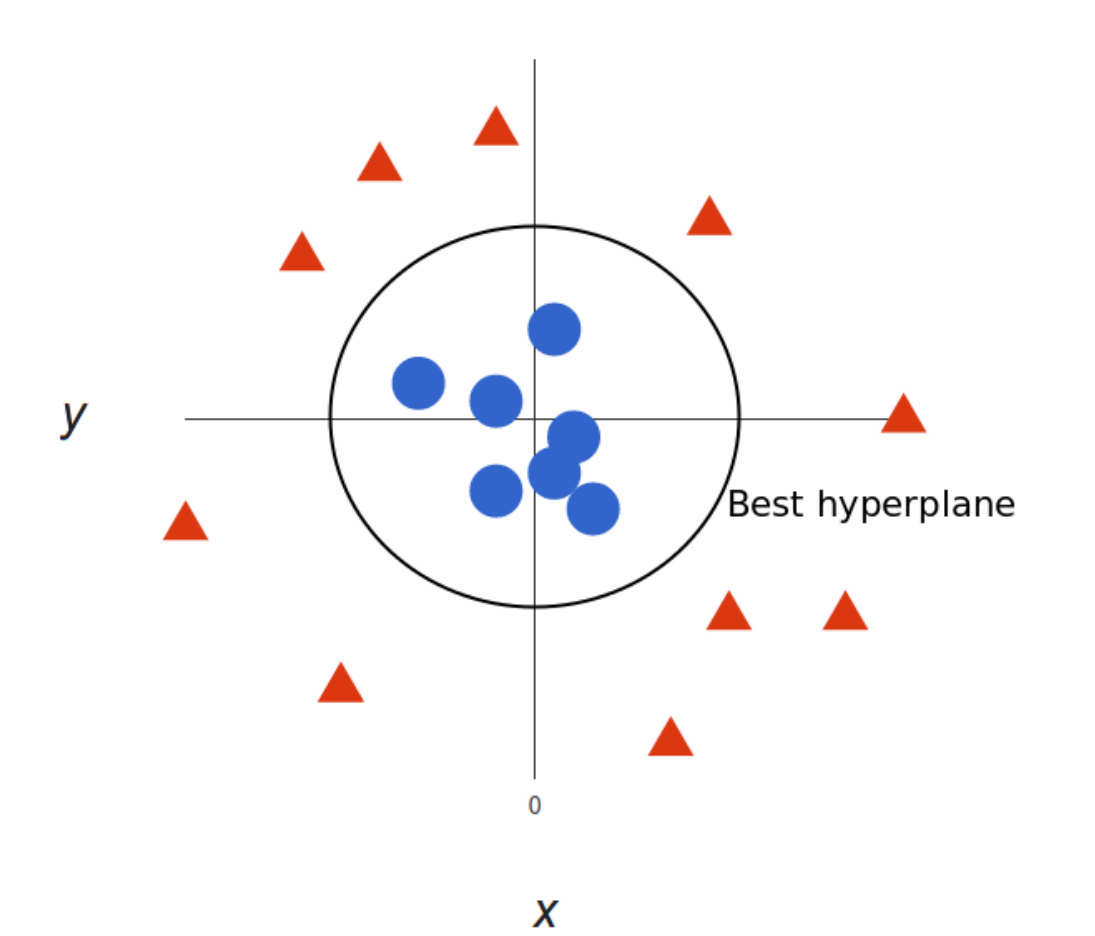
Visualization



Visualization



Visualization



Non-Linear Classifier

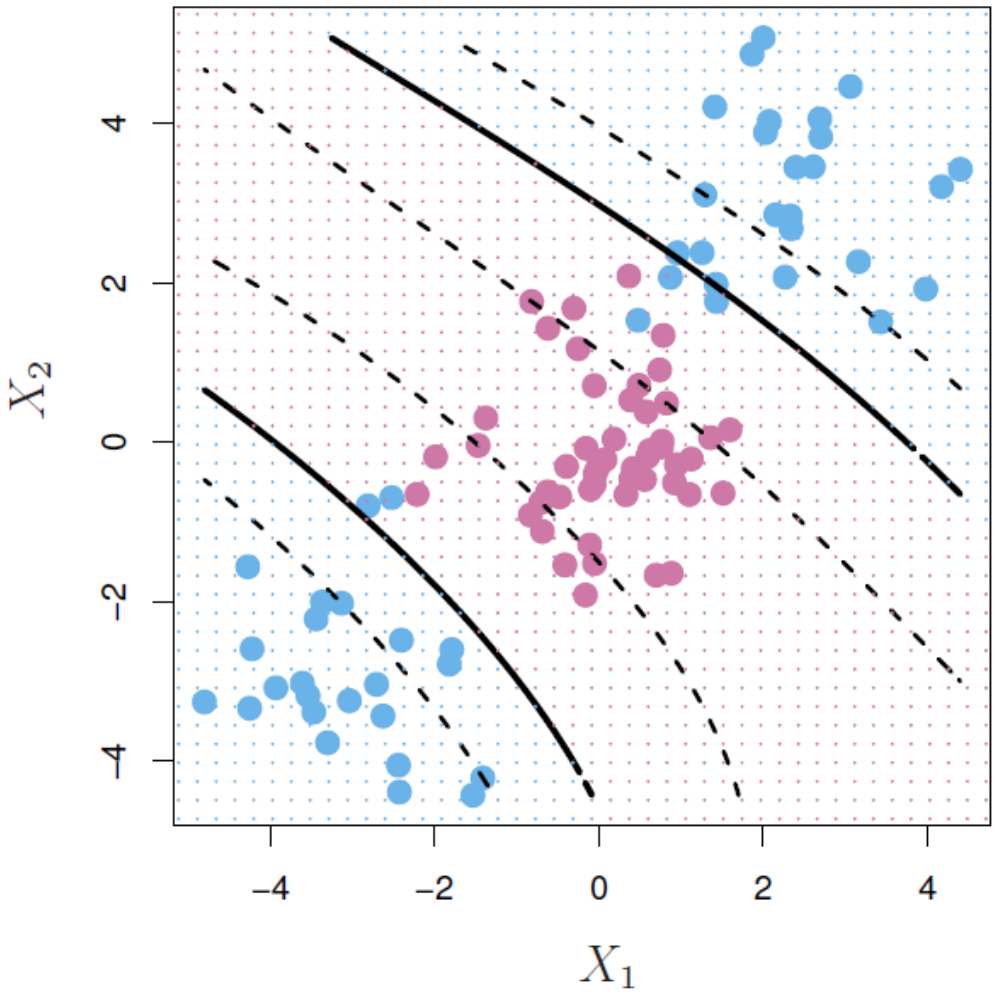
- Suppose we use other predictors instead of just (X_1, X_2). Then the decision boundary would be of the form (for example)

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

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

Non-Linear Classifier

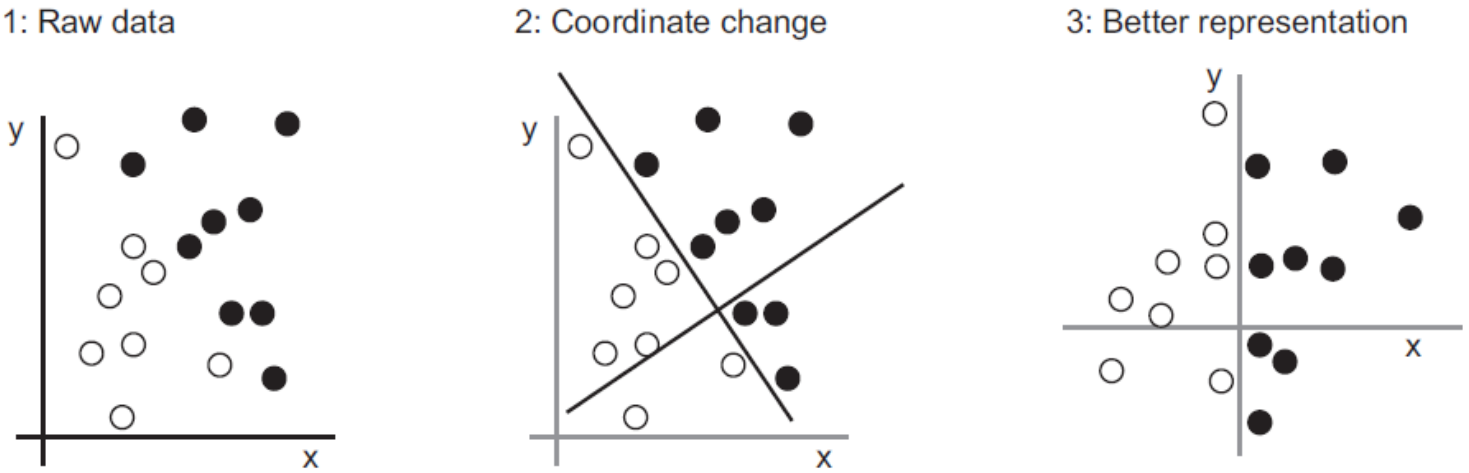
- Here we use a basis expansion of cubic polynomials
- From 2 variables to 9
- The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space



$$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_1^2+\beta_4X_2^2+\beta_5X_1X_2+\beta_6X_1^3+\beta_7X_2^3+\beta_8X_1X_2^2+\beta_9X_1^2X_2=0$$

Kernels

- Everything boils down on how you represent the incoming data.
- An ML model transforms its input data into meaningful outputs
- Key is to transform data: useful representations of the input data at hand



Kernels

- SVM ~ finding good decision boundaries
  - Map the data to a new high-dimensional representation
  - Find a good decision boundary possibly non-linear
- https://www.youtube.com/watch?v=3liCbRZPrZA
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers - through the use of kernels.

Kernels

- Kernel Trick: to find good decision hyperplanes in the new representation space, we don’t need to explicitly compute the coordinates of your points in the new space
- Instead we need the distance between them
- Kernel function efficient way to compute these distances.
- The idea is: the inner product

Inner Products and Support Vectors

- Inner Product between 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

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

Inner Products and Support Vectors

- To estimate the parameters α and β, all we need are the inner products (x_i, x_i') between all pairs of training observations.
- It turns out that most of the α can be zero

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

- S is the support set of indices i such that $\alpha_i > 0$.

Kernels and Support Vectors Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special kernel functions can do this for us. E.g.

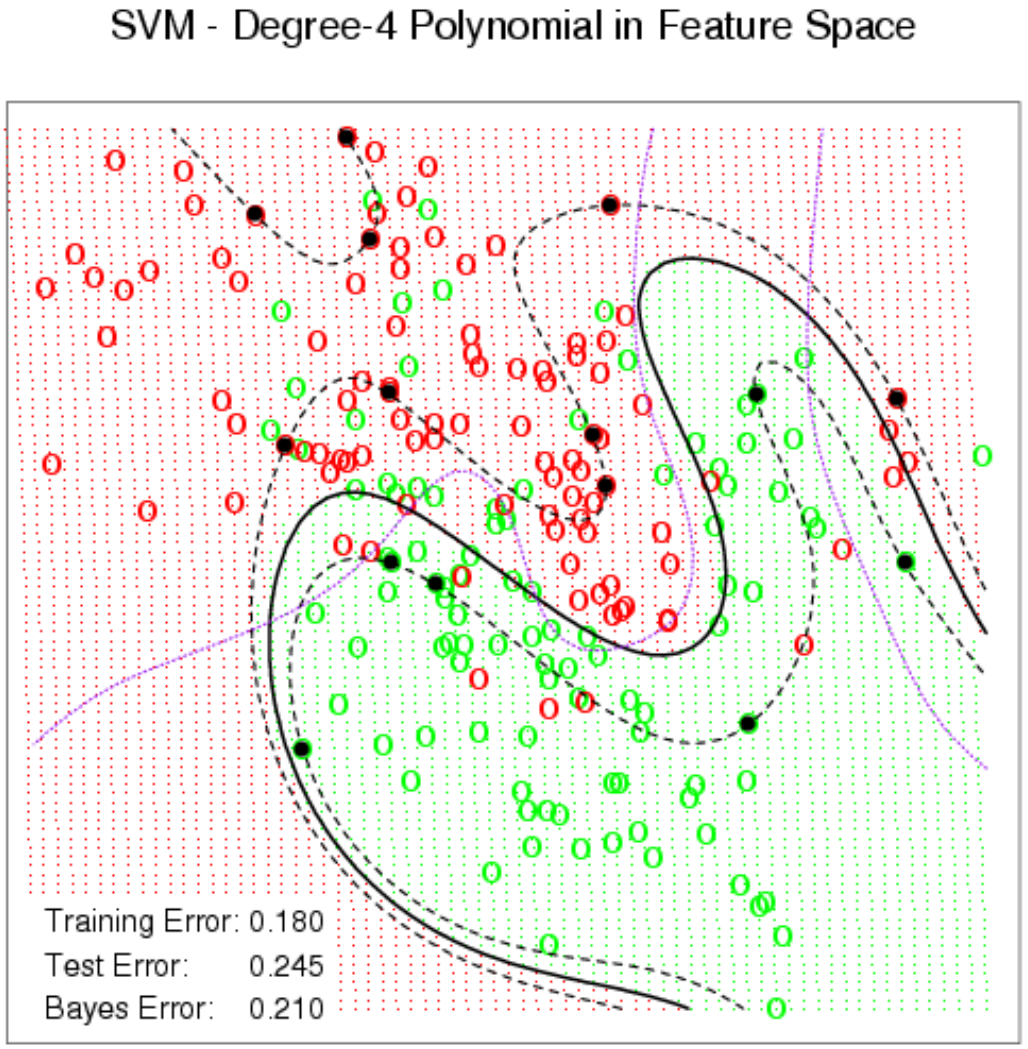
$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

- computes the inner-products needed for d dimensional polynomials
- The solution has the form

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

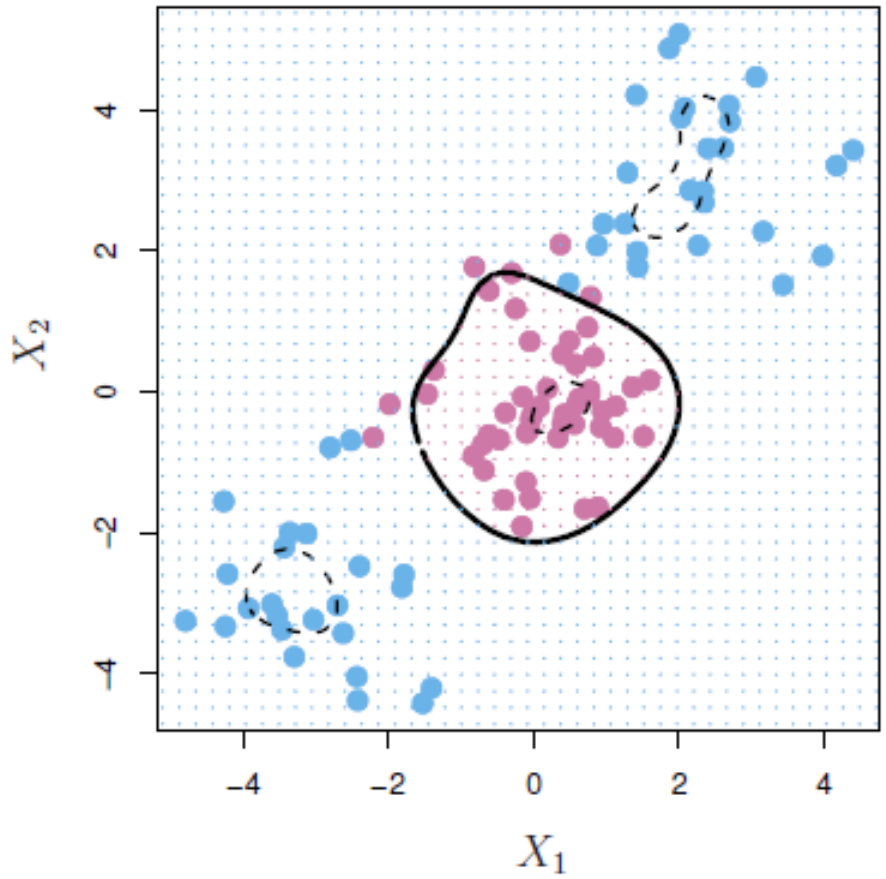
Polynomial Kernel On Sim Data

- Using a polynomial kernel we now allow SVM to produce a non-linear decision boundary.
- Notice that the test error rate is a lot lower.



Radial Kernel

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

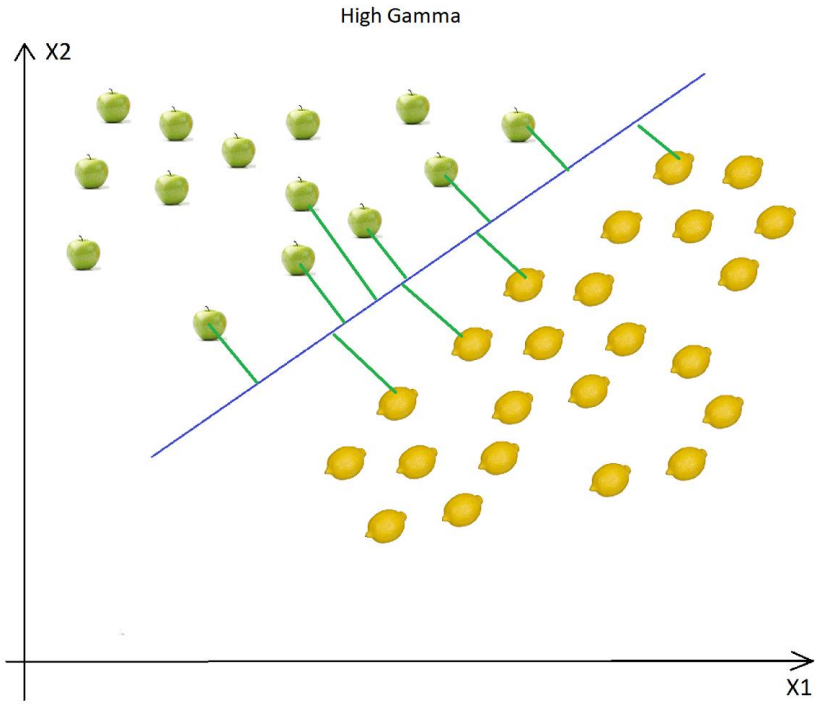


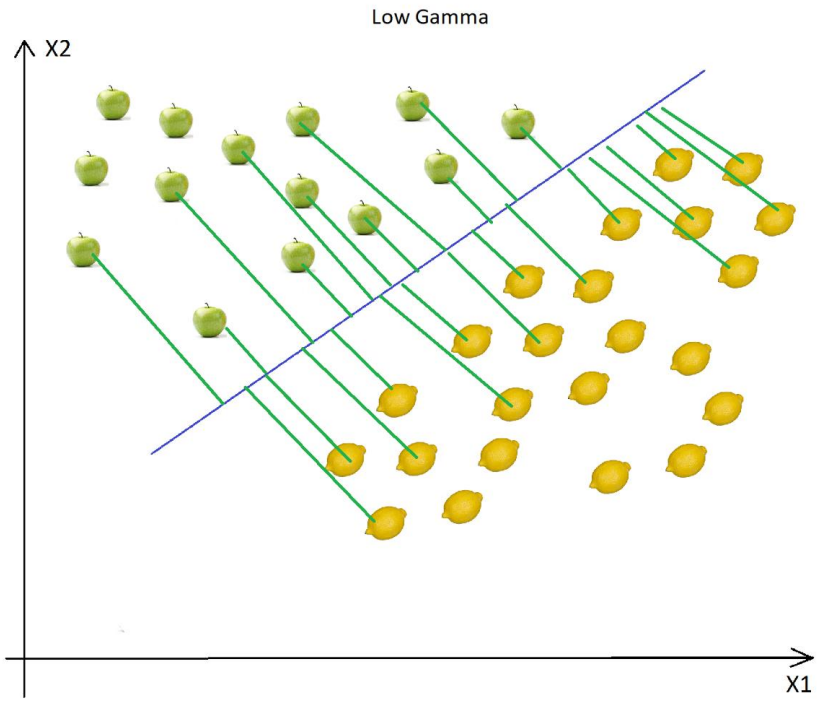
$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space; very high dimensional.

Controls variance by squashing down most dimensions severely

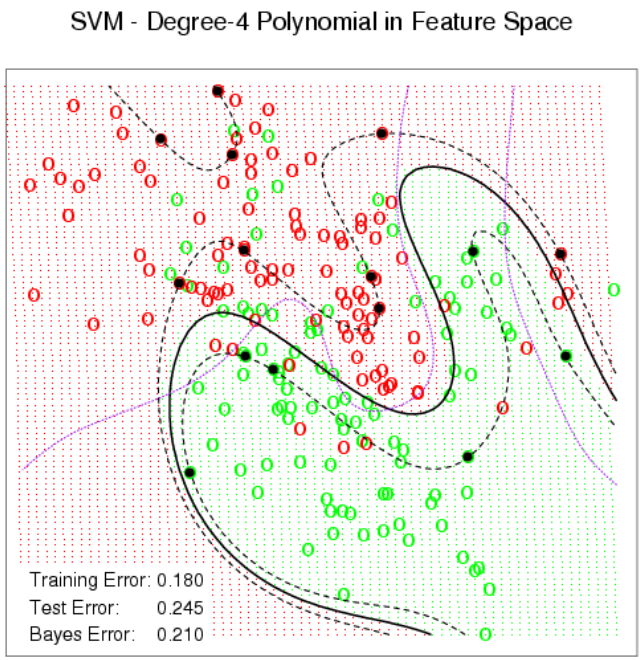
Gamma

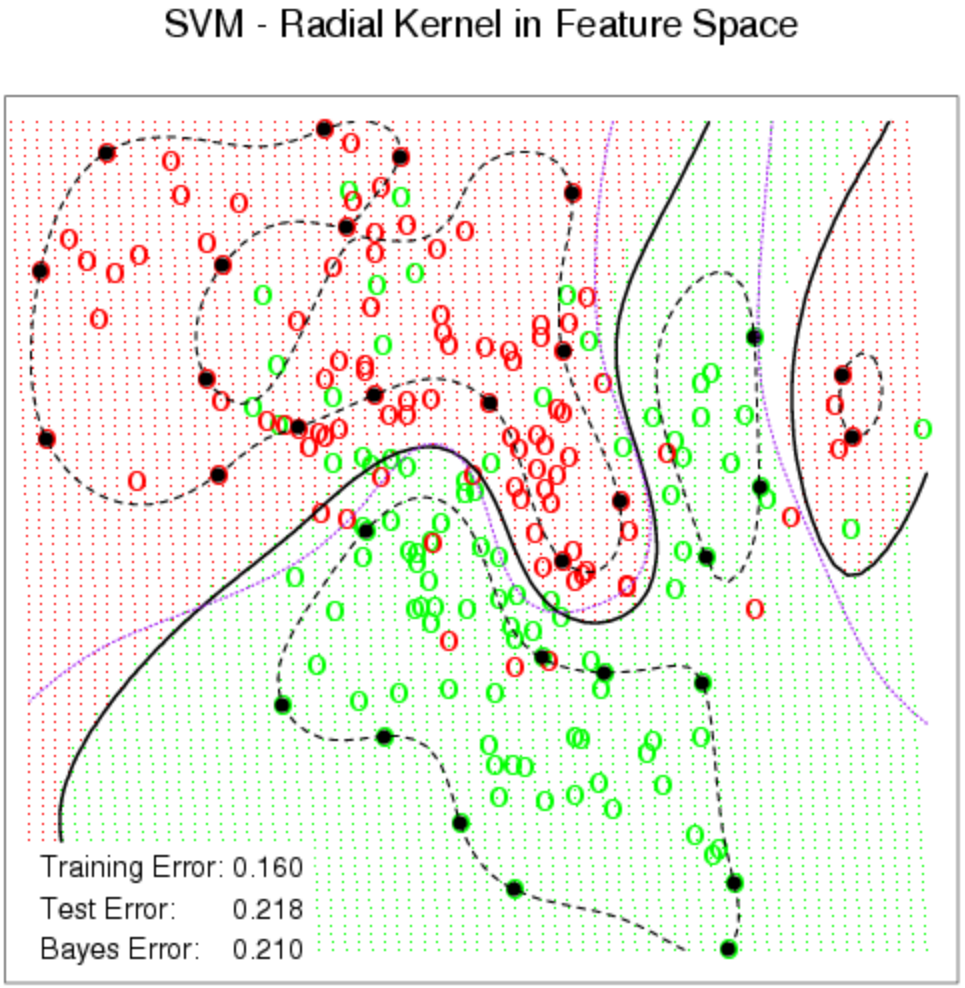




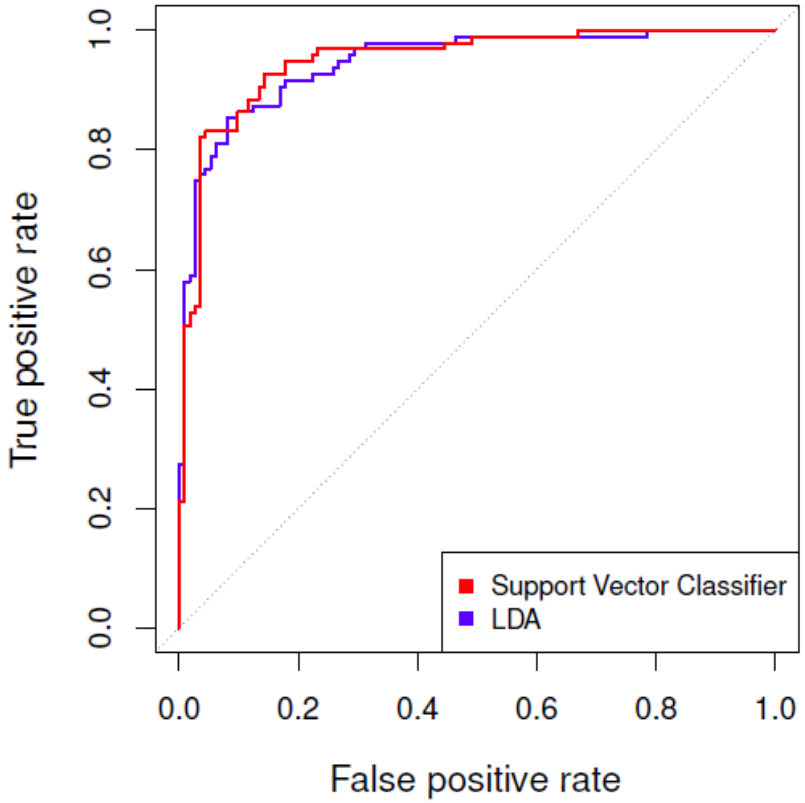
Radial Basis Kernel

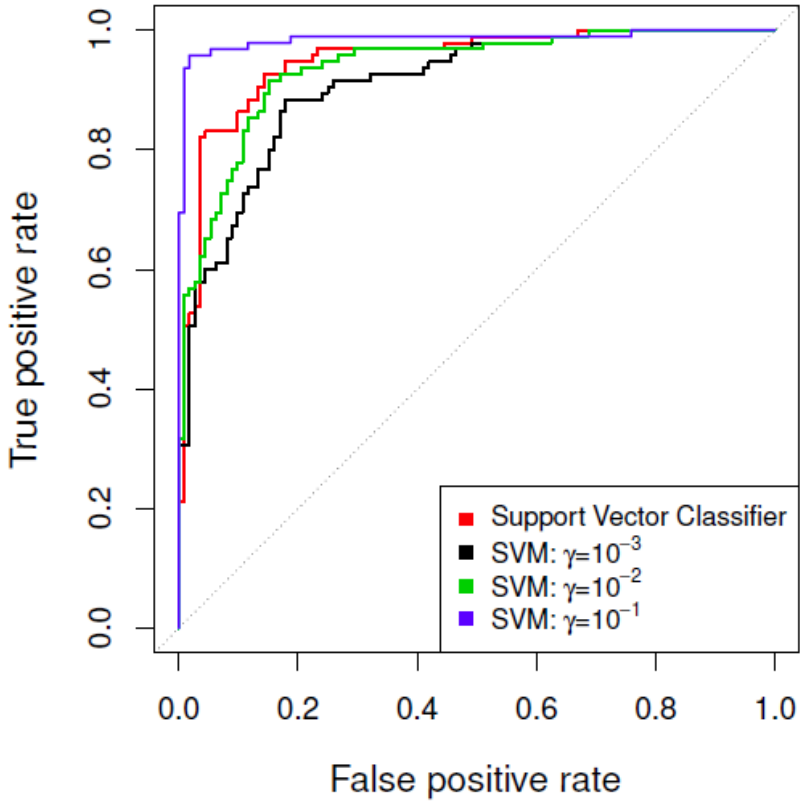
- Using a Radial Basis Kernel you get an even lower error rate.



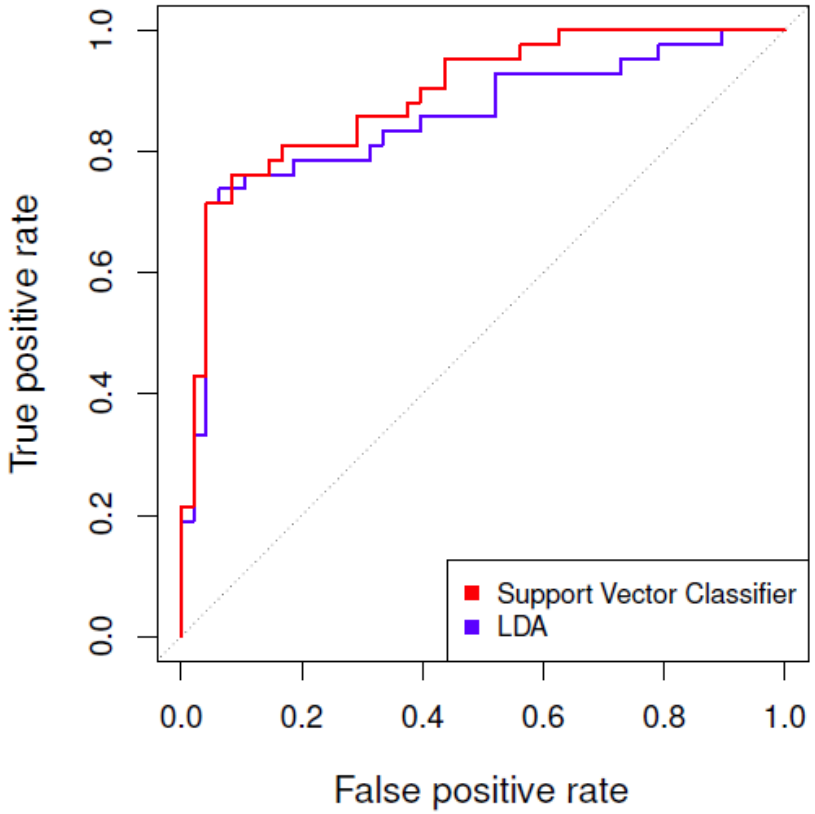


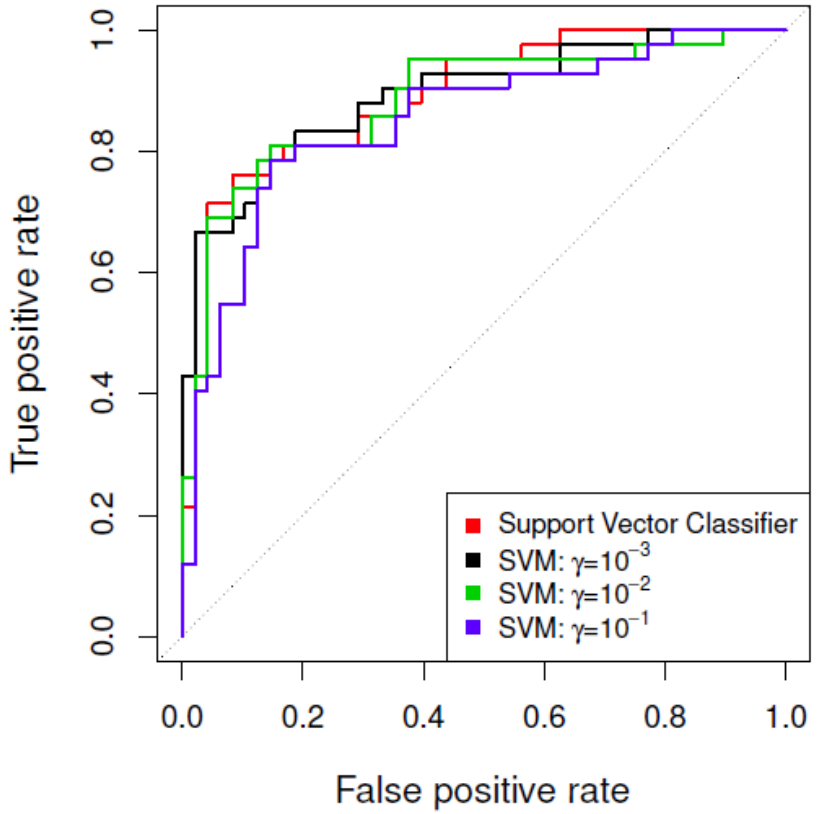
Comparison


True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Support Vector Classifier
LDA

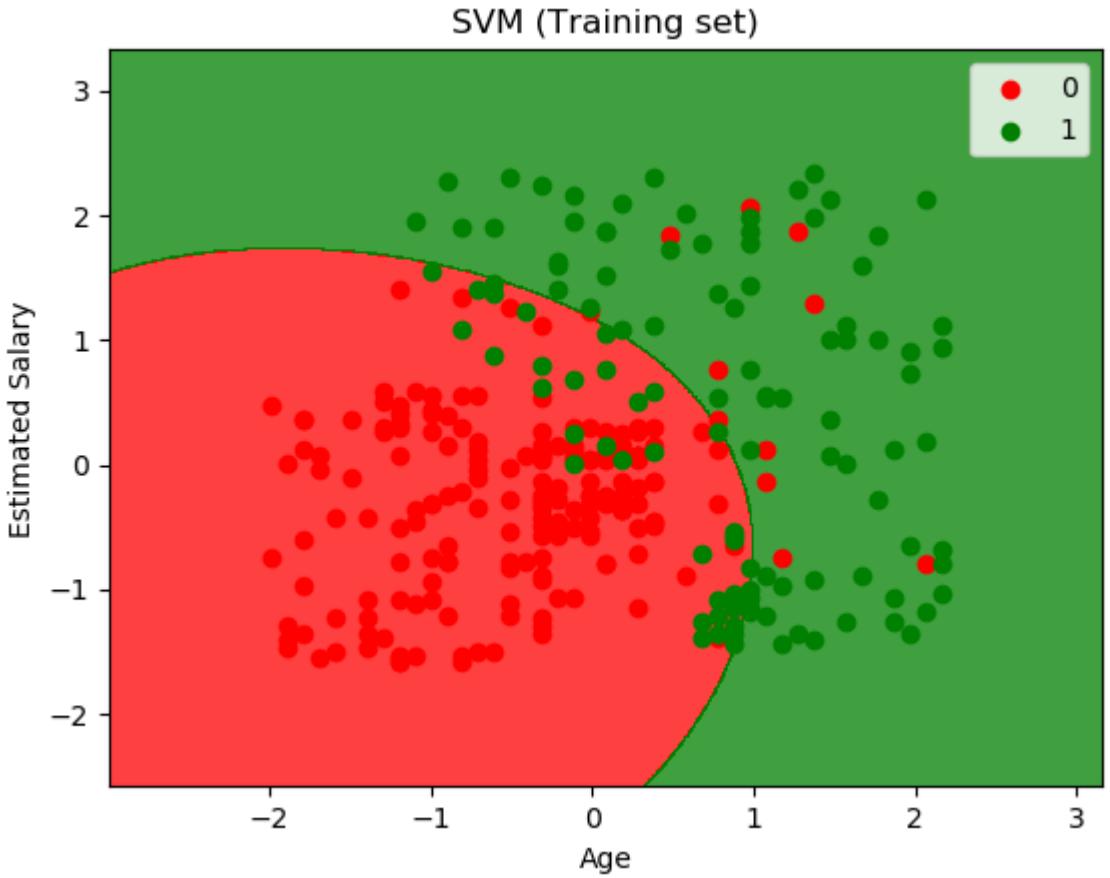

True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Support Vector Classifier
SVM: γ=10^−3
SVM: γ=10^−2
SVM: γ=10^−1

Comparison

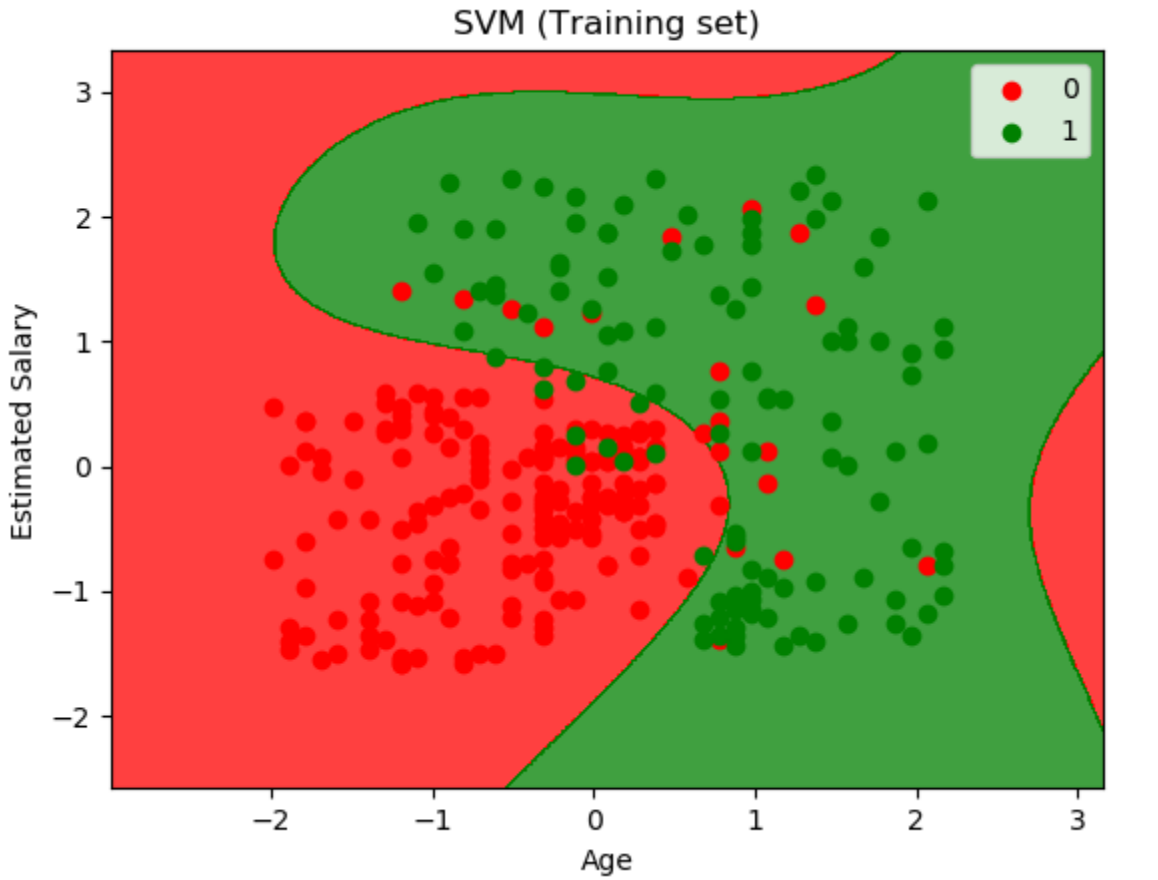

True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Support Vector Classifier
LDA


True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Support Vector Classifier
SVM: γ=10^-3
SVM: γ=10^-2
SVM: γ=10^-1

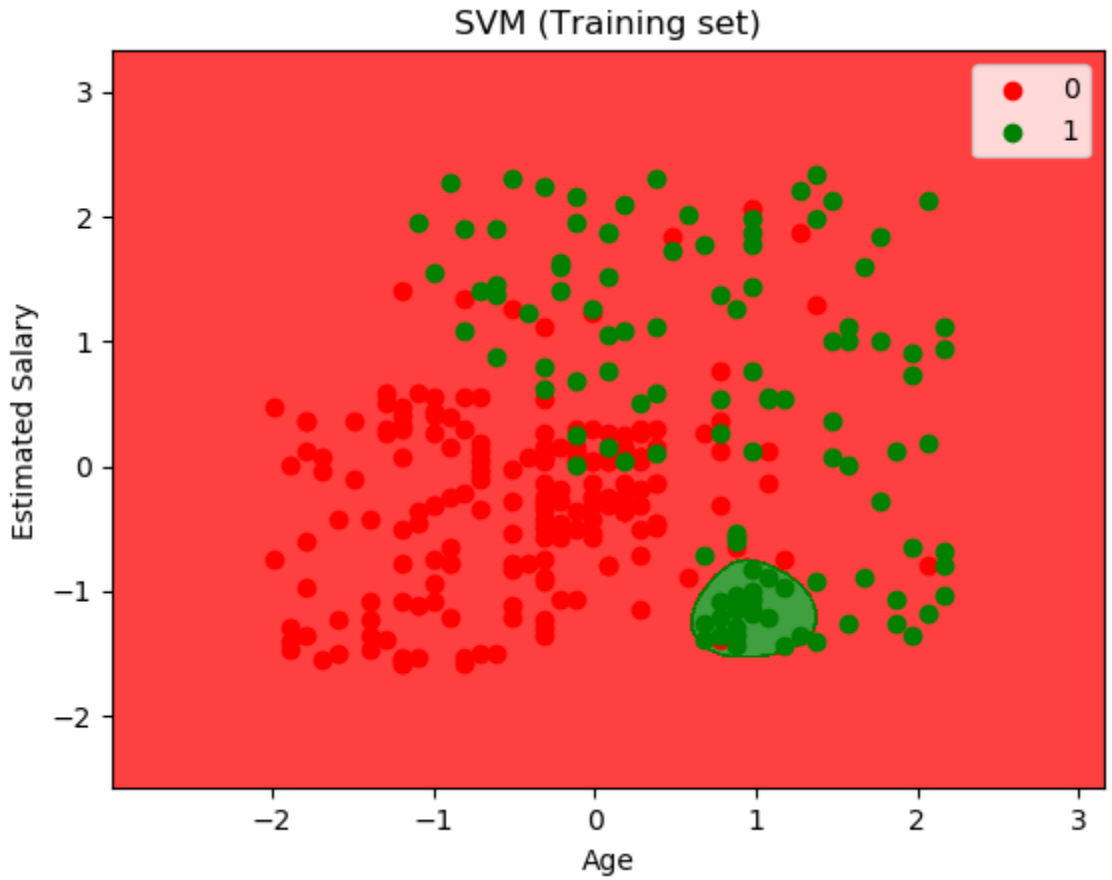
Hyperparameter Tuning



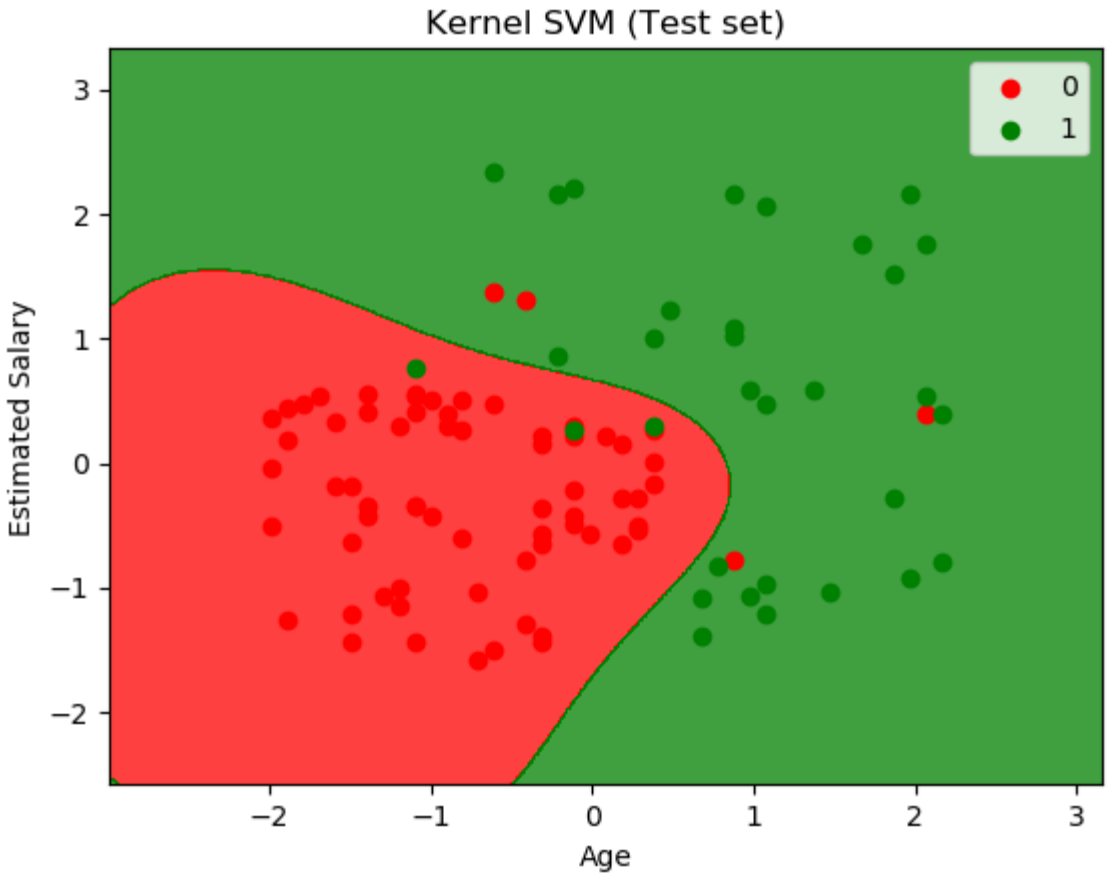
Hyperparameter Tuning



Hyperparameter Tuning



Hyperparameter Tuning



Multi-Nominal Classifier

- 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 e.g. each class versus the rest. Classify x to the class for which the margin is largest.

- OVO: One versus One. Fit all pairwise classifiers e.g. each class versus each other. Classify x to the class that wins the most pairwise competitions

Multi-Nominal Classifier

- Linear Kernel

$$k(x_i, x_j) = x_i * x_j$$

- Polynomial Kernel

$$k(x_i, x_j) = (1 + x_i * x_j)^d$$

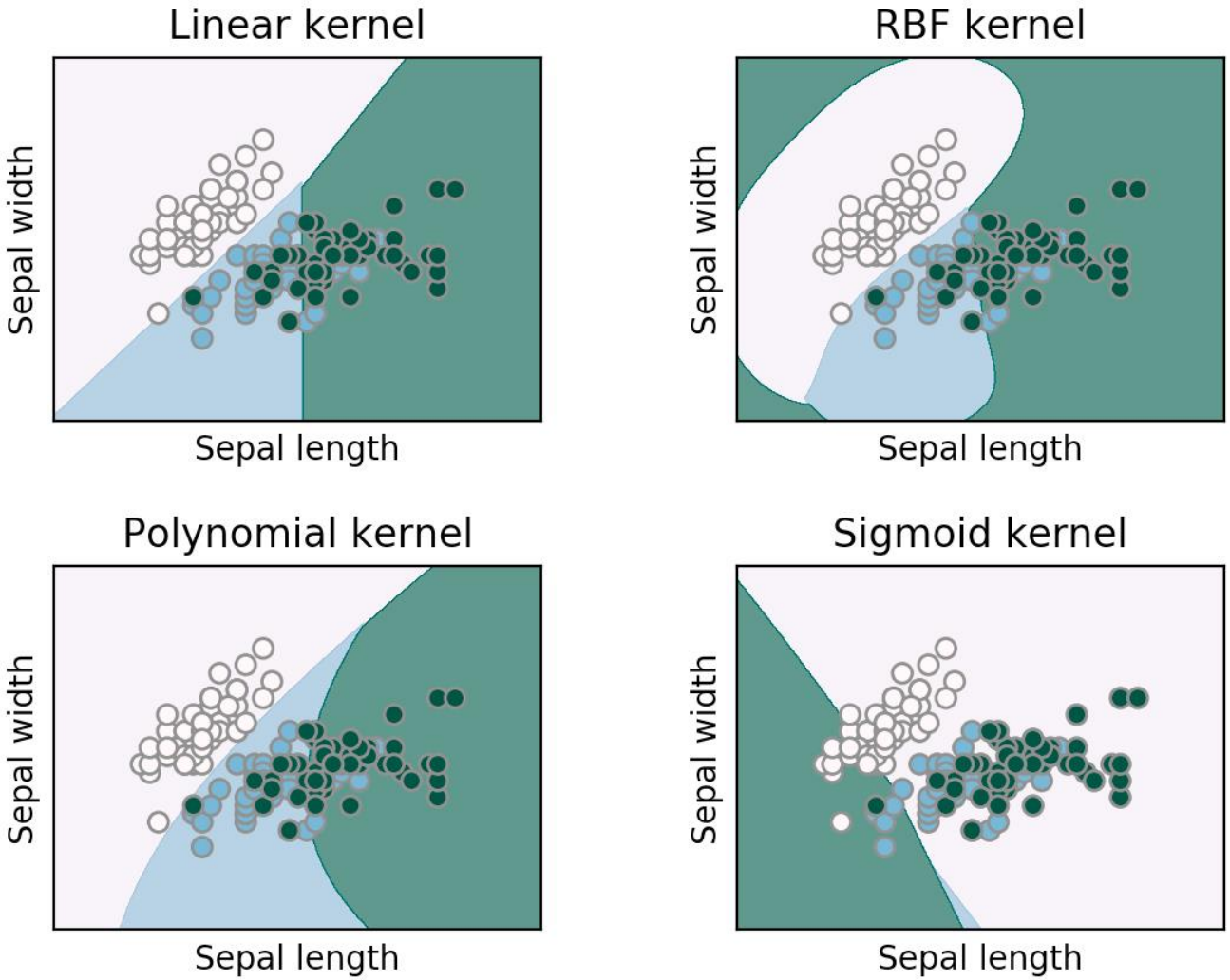
- Radial Kernel

$$k(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2)$$

- Sigmoid Kernel

$$k(x_i, x_j) = \tanh(\alpha x^T y + c)$$

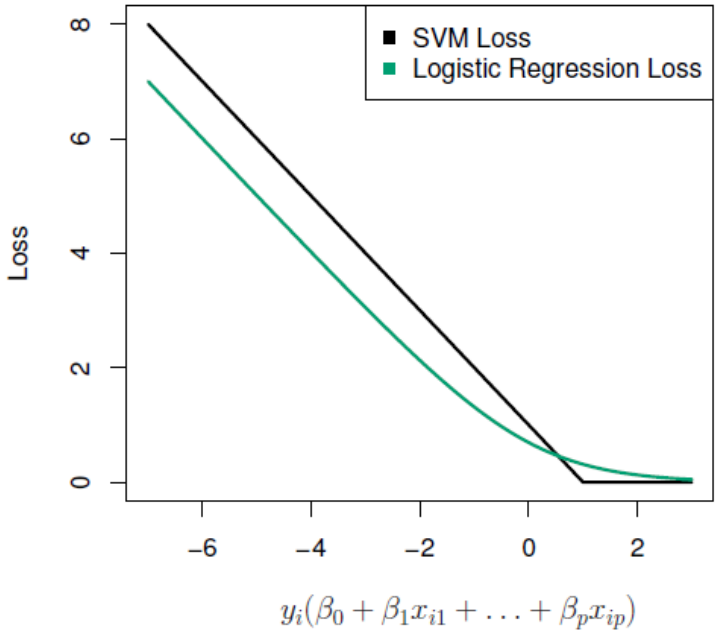
Multi-Nominal Classifier



SVM vs Logistic Regression

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

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



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).

SVM vs 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.

SVM ~ Pros

- SVN can be very efficient, because it uses only a subset of the training data, only the support vectors
- Works very well on smaller data sets, on non-linear data sets and high dimensional spaces
- Is very effective in cases where number of dimensions is greater than the number of samples
- It can have high accuracy, sometimes can perform even better than neural networks
- Not very sensitive to overfitting

SVM ~ Cons

- Training time is high when we have large data sets
- When the data set has more noise (i.e. target classes are overlapping) SVM doesn’t perform well

Support Vector Regression

- Support Vector Machines are developed for use mainly in classification problems.
- However, they might of use of in regression as well
- Such models are known as Support Vector Regression.

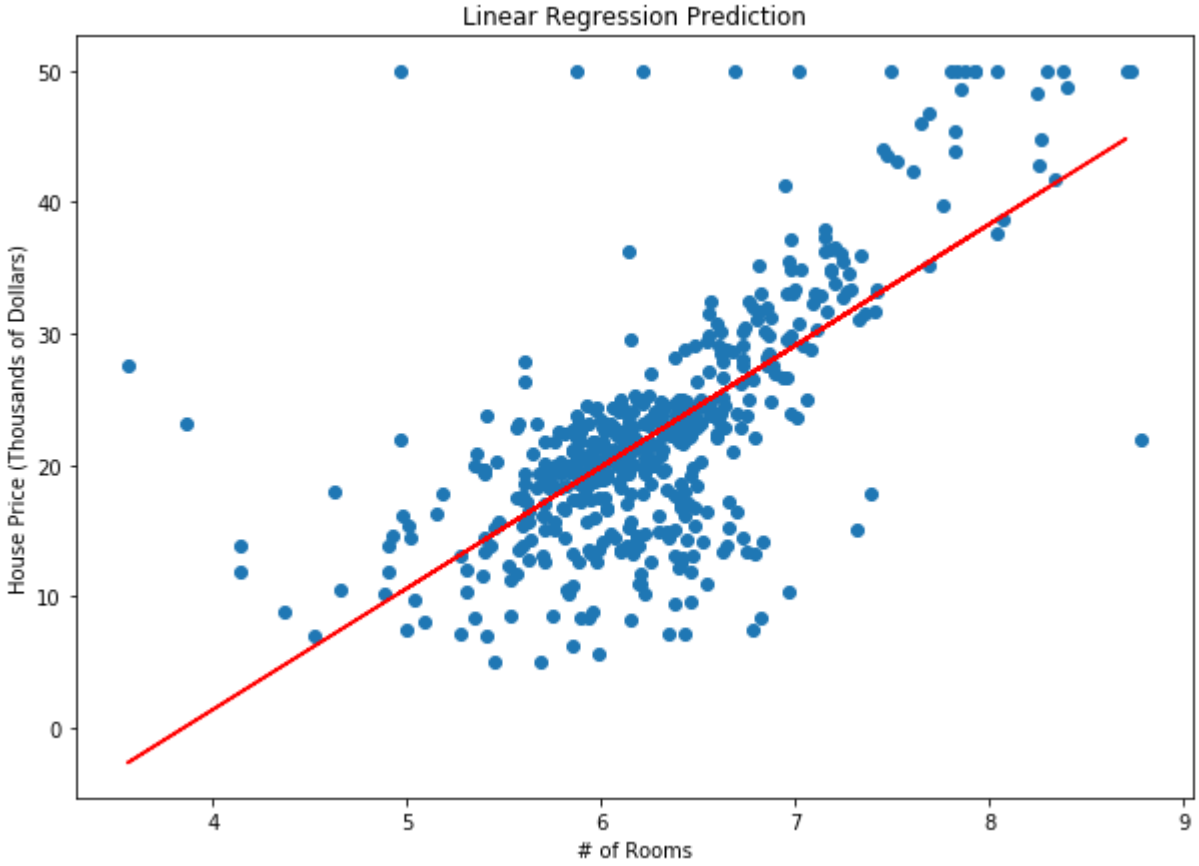
Linear vs Support Vector Regression

- In standard linear regression, our objective is to minimize the sum of squared errors (remember OLS). Suppose with one predictor, we have

$$MIN \sum_{i=1}^{n} (y_i - w_i x_i)^2$$

Linear vs Support Vector Regression

- The resulting function will look like



Linear vs Support Vector Regression

- SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data.
- The objective is to minimize the coefficients — more specifically, the *l*2-norm of the coefficient vector — not the squared error.

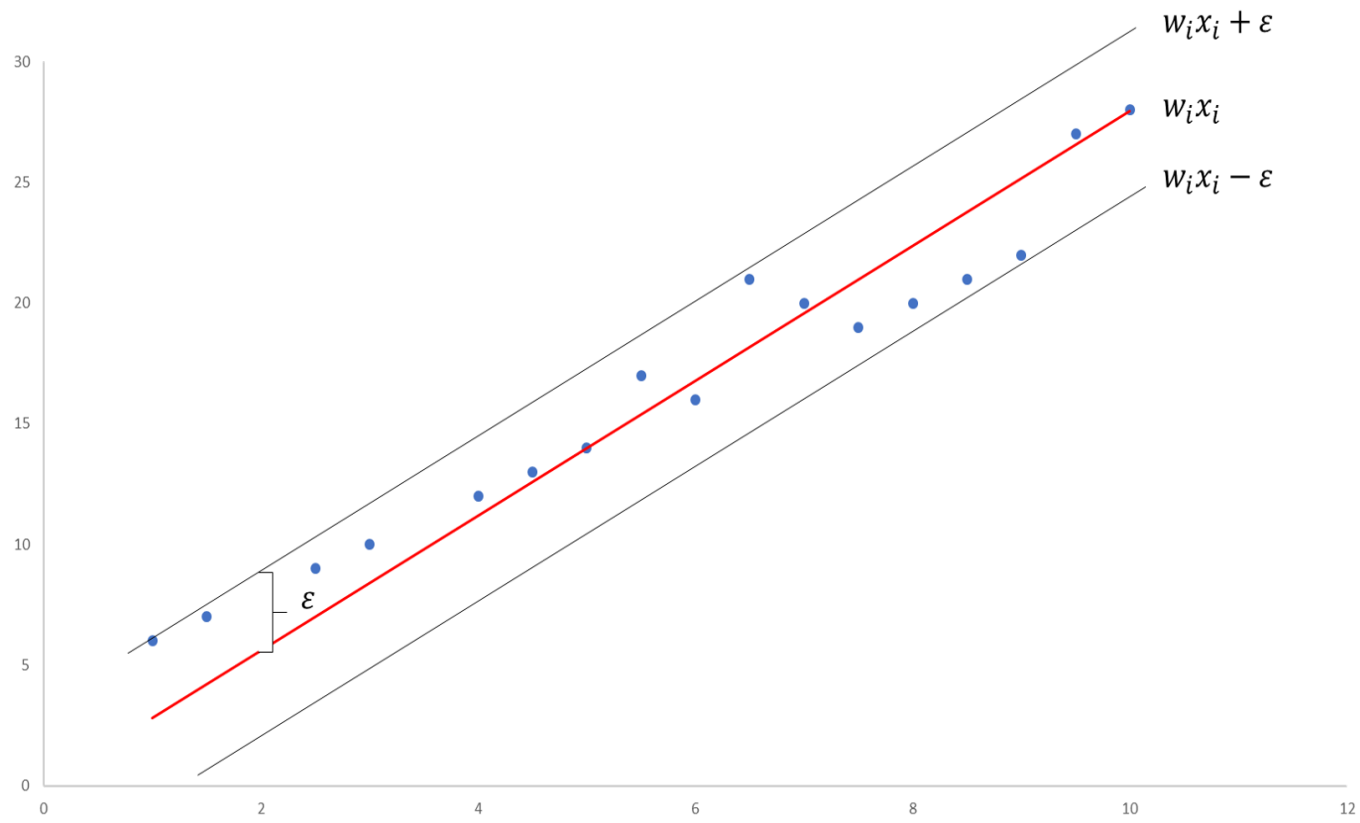
$$MIN\ \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2$$

$$|y_i - w_i x_i| \leq \varepsilon$$

- Why do we do that?

Linear vs Support Vector Regression

- The resulting function will look like



Linear vs Support Vector Regression

- Obviously the model below might be infeasible as we may not found a solution satisfies those constraints
- Even if it may not be infeasible, we may prefer a solution that violates the constraints to certain extend. Why?

$$MIN\ \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2$$

$$|y_i - w_i x_i| \leq \varepsilon$$

- What do we do now?

Linear vs Support Vector Regression

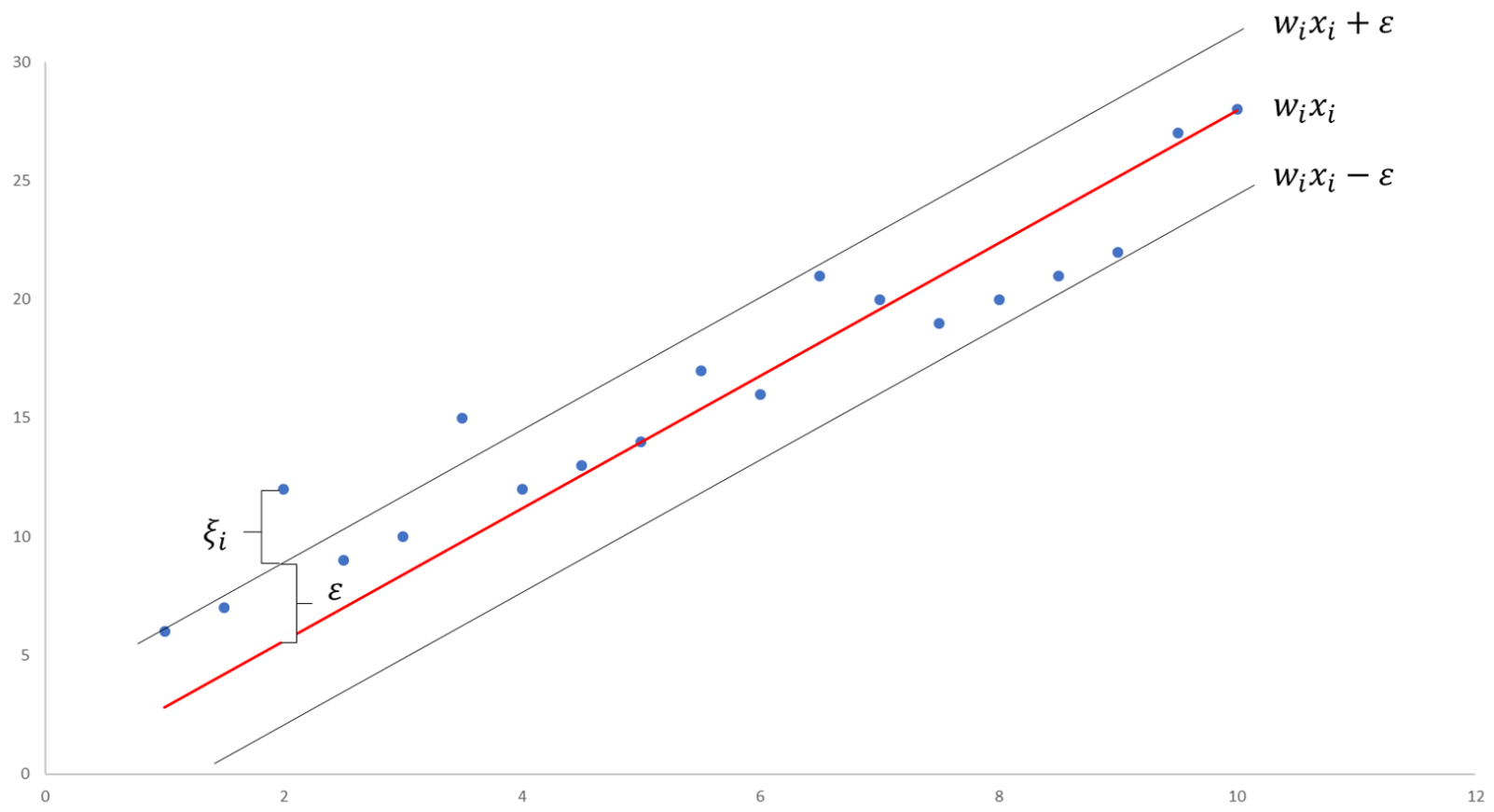
- We relax the error restriction to a certain extend and penalize it with a penalty coefficient in the objective function.

$$MIN\ \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + C\sum_{i=1}^{n}|\xi_i|$$

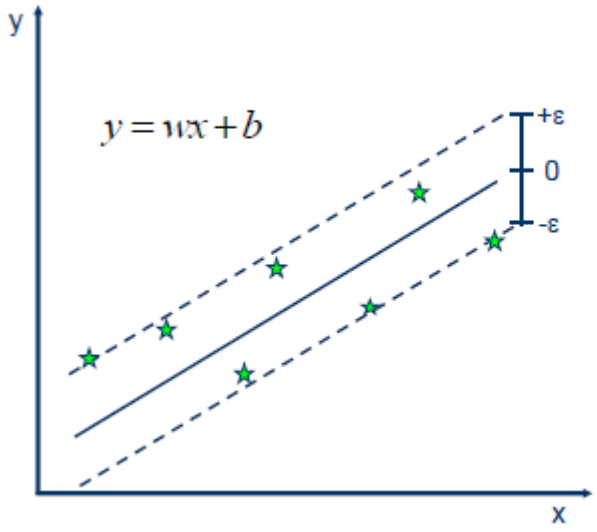
$$|y_i - w_i x_i| \leq \varepsilon + |\xi_i|$$

Linear vs Support Vector Regression

- The resulting function will look like



Support Vector Regression



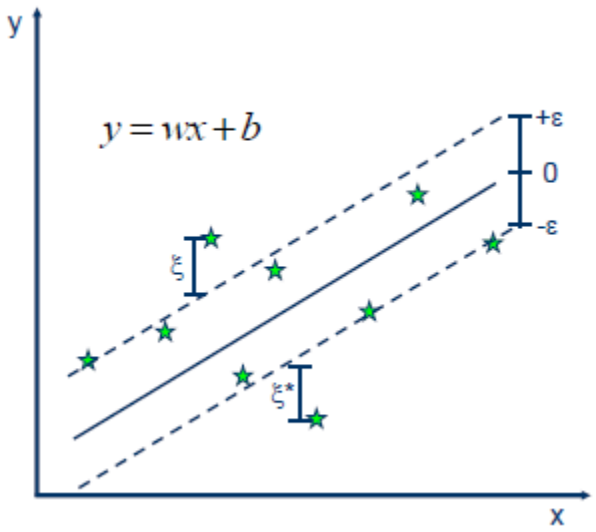
- Solution:

$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \leq$$

$$wx_i + b - y_i \leq$$



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$

$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$

Support Vector Regression