INFSCI 2915: Machine Learning

Support Vector Machines

Mai Abdelhakim

School of Computing and Information

610 IS Building

Spring 2018

Outline

- Maximum Margin Classifier
- Support Vector Classifier (SVC)
- Support Vector Machines (SVM)

Support Vector Classifiers (SVC) / Support Vector Machines (SVM)

- Very popular & powerful

- **Objective**: **Find a hyperplane (decision boundary) that best separates the classes**

- Linear support vector classifier is one of the linear models for classification
  - This means that the **decision boundary** used to separate the classes in the feature space is **linear function of the features**

- Linear decision boundary can be written as:

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

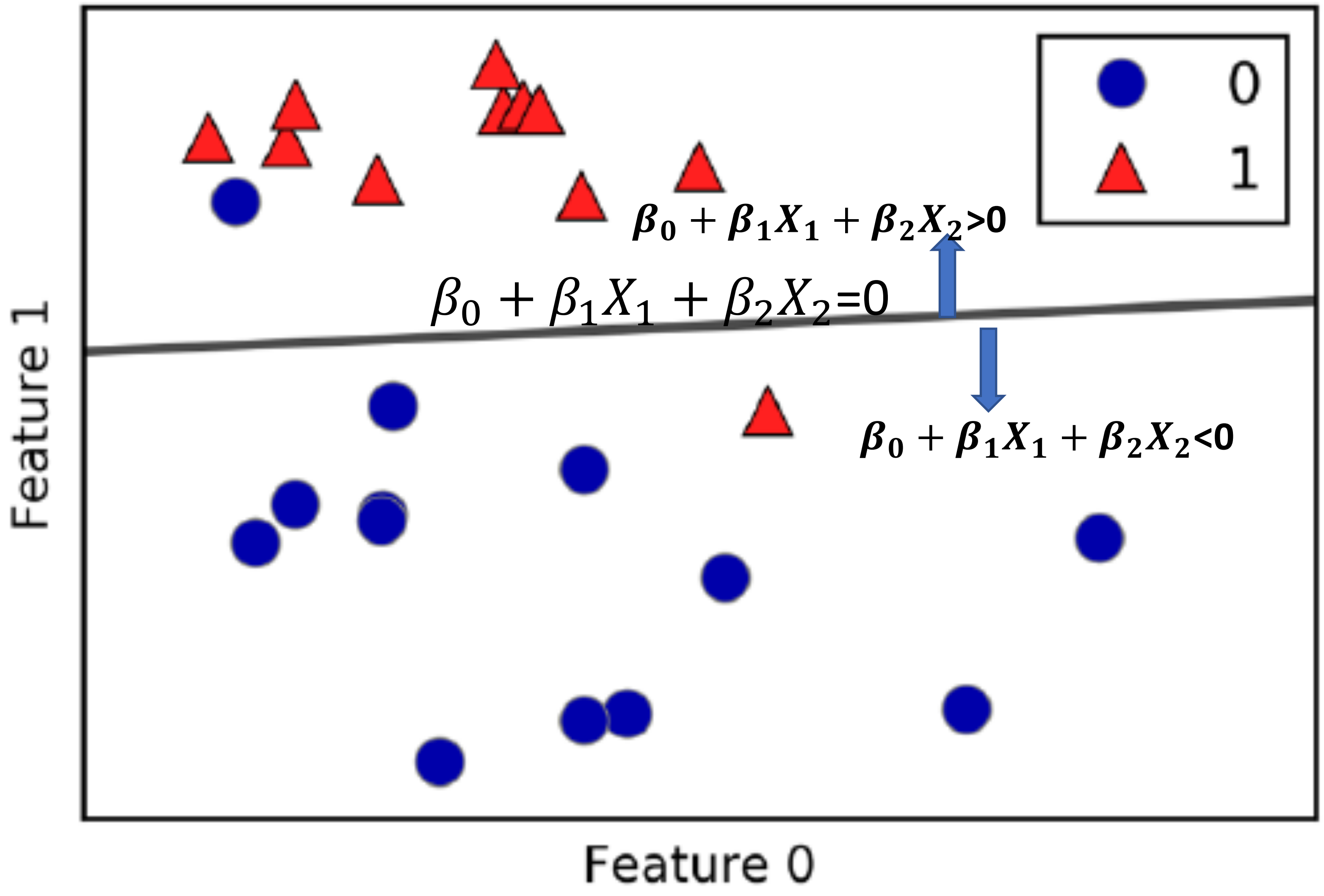
  - When p=2 the boundary is a line, for higher dimension it is a **hyperplane**

- Consider binary classification (two classes): positive class ($label = +1$) and negative class ($label \ = -1$)

  For each training point i

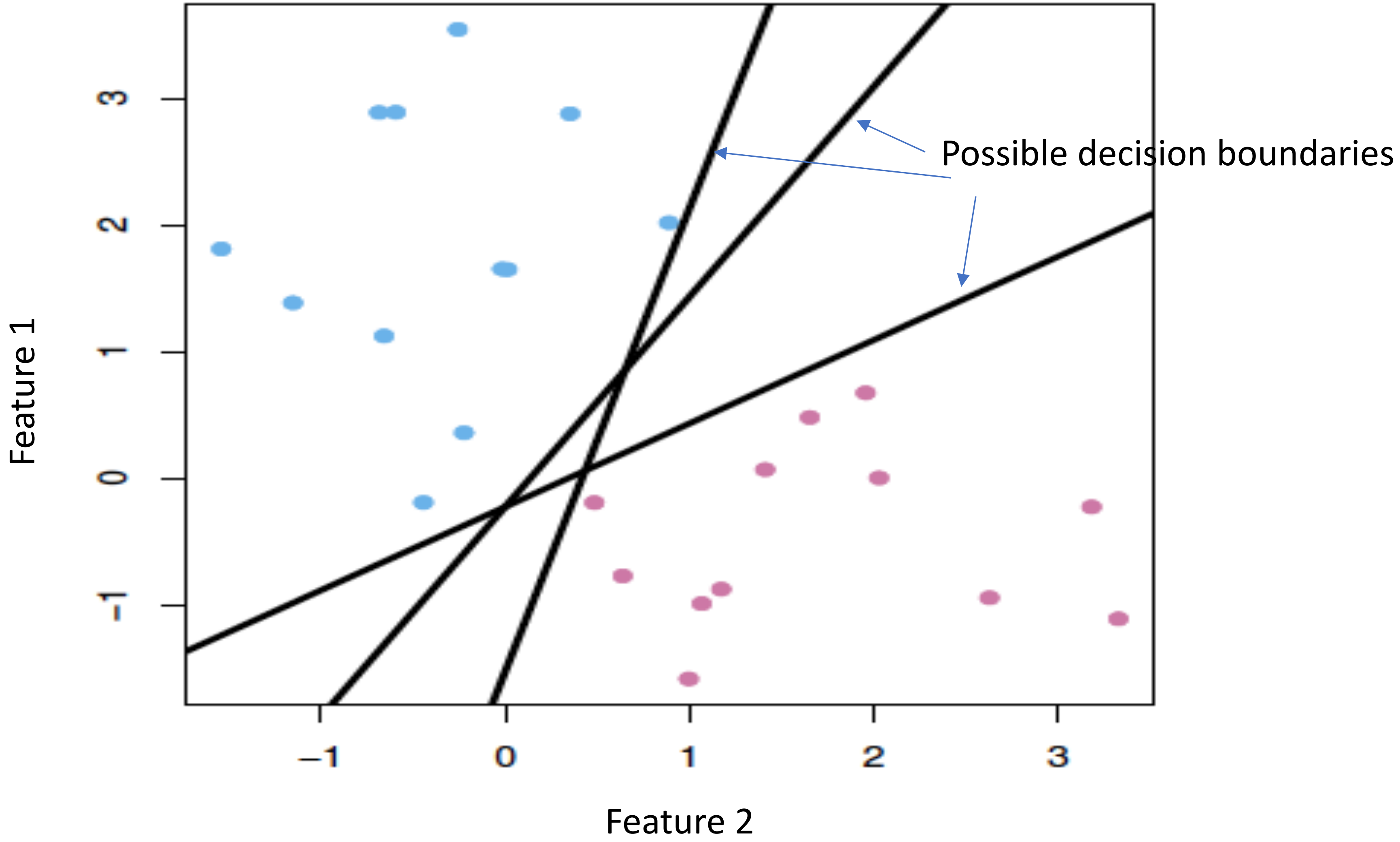
  - **Decide the positive class (** $\boldsymbol{Y}_{\boldsymbol{i}} = +\mathbf{1}$) **if:** $\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip}$>0
  - **Decide the negative class (** $\boldsymbol{Y}_{\boldsymbol{i}} = -\mathbf{1}$) **if:** $\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip}$<0

Example of Data Set with 2 Features



Infinite Possible Decision Boundaries

- But there could be many possible decision boundaries. Which one to choose?

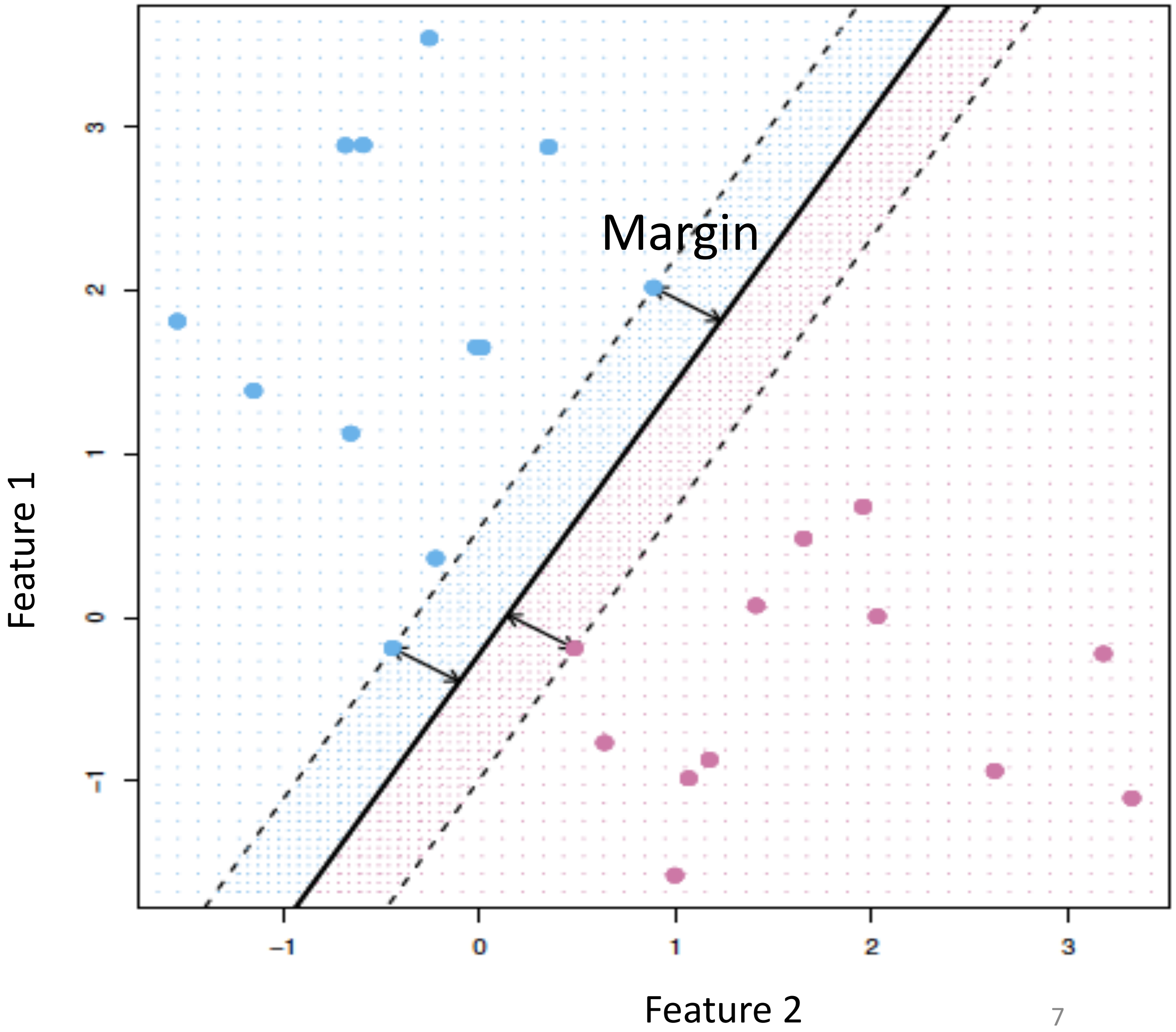


Maximal Margin Classifier

- The distance from the decision boundary (hyperplane) gives us more confidence about the class assignment

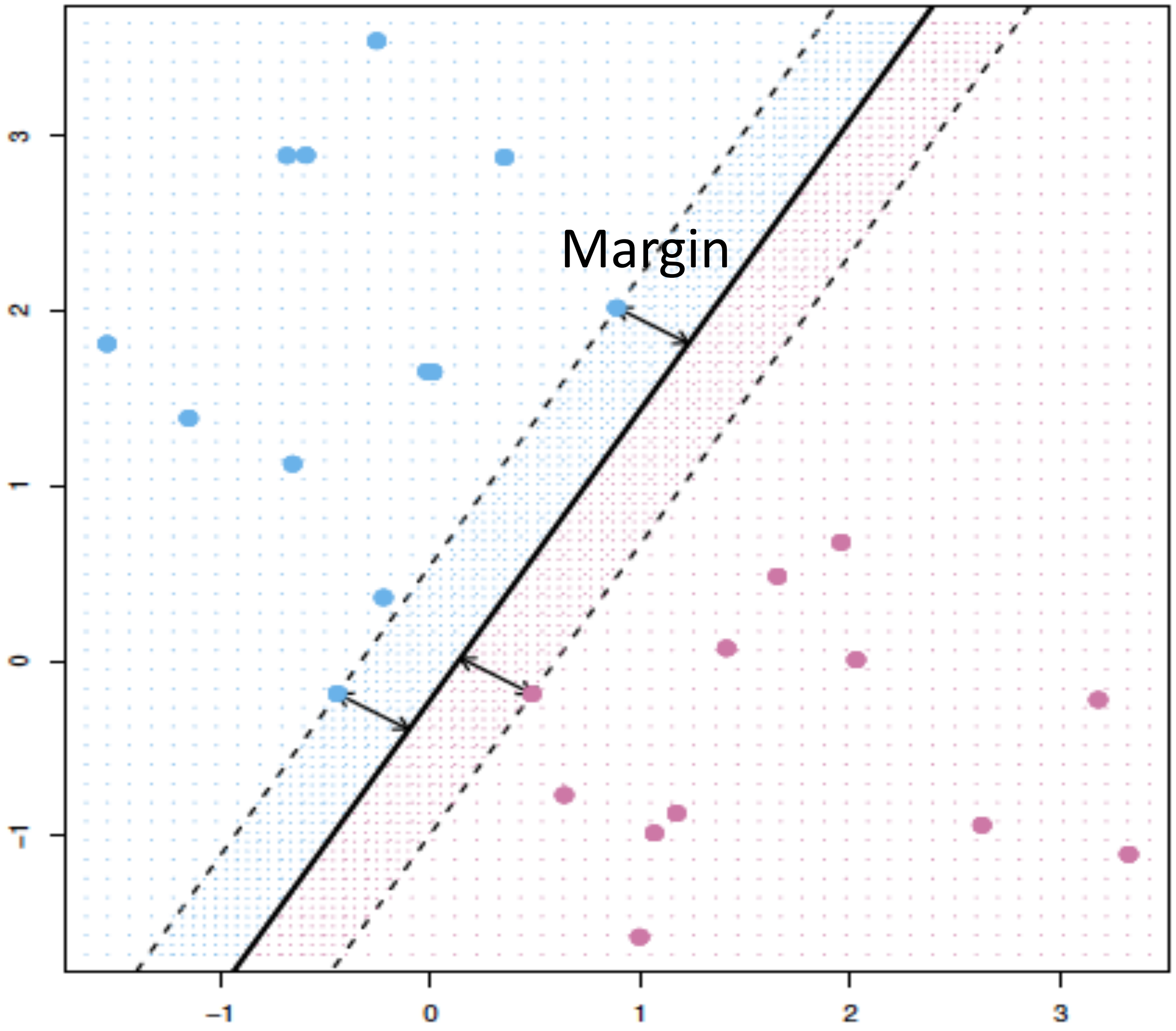
- From all possible decision boundaries, **find the one the maximizes the margin (gap) between the two classes**

- Choose boundary is the farthest from the training observations



- **Margin**: is the perpendicular **distance** of the **closest training points** to the **decision boundary**

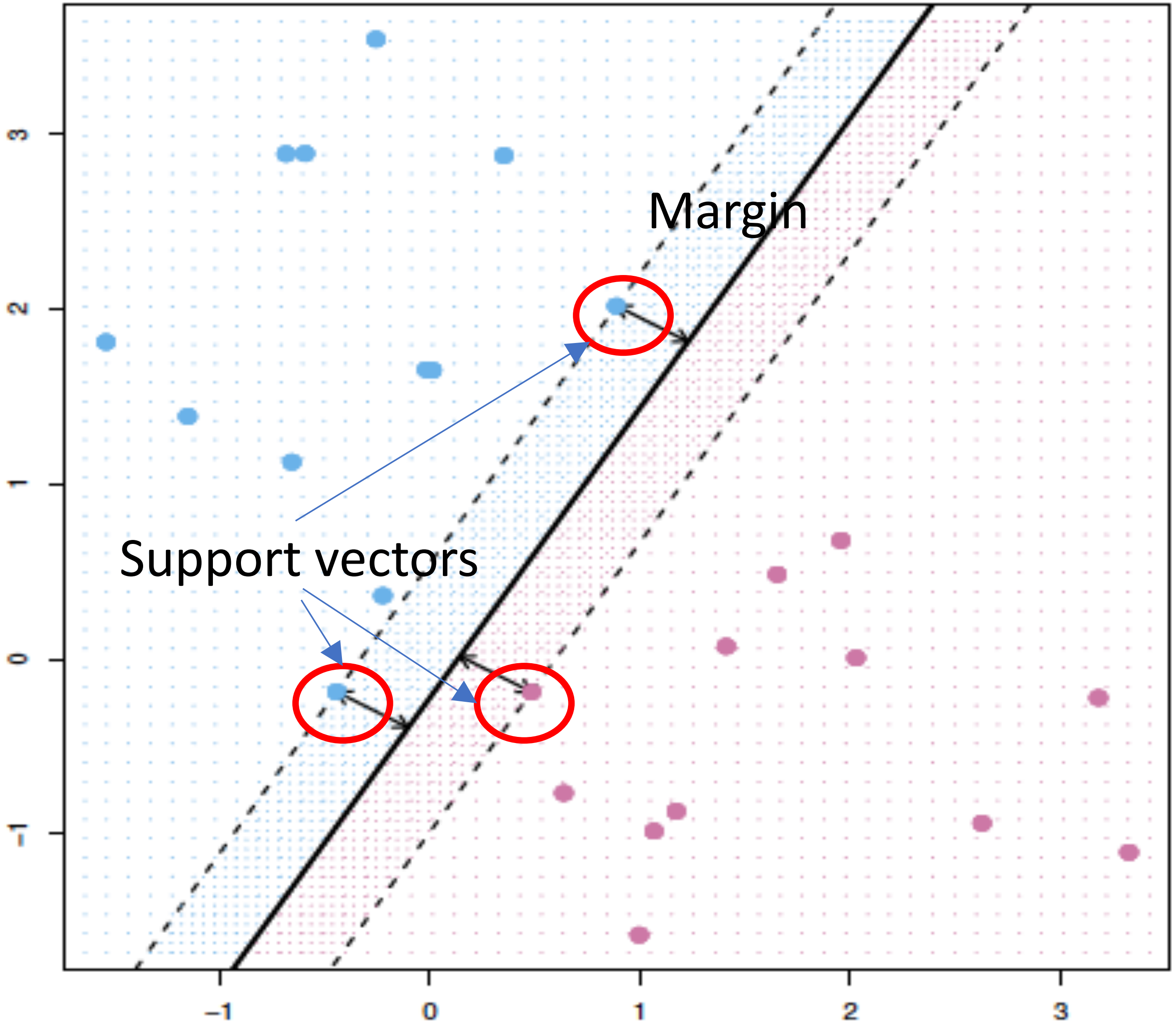
- **Maximal margin classifier maximizes this margin**
    - **Largest minimum distance to the nearest training point**



Support Vectors

- Training observations that indicate the width of the margins are called **support vectors**

- When classes are **perfectly separable**, **support vectors** are training observations that are **equidistant** from the maximal margin **decision boundary**



Maximum Margin Classifier

- **Finds the hyperplane (linear decision boundary) that maximizes the margin (gap) between two classes in the feature space**
- Unique property: The maximum margin classifier **depends on the support vectors** and not on other training observations
- The hope is that **largest margin on training data** will also work well on **test data**
- **Assumes** that the classes can be **perfectly separable**

How to Construct the Maximal Margin Classifier

- Recall that for training point i
  - Decide $y_i = +1$, if $\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$>0
  - Decide $y_i = -1$, if: $\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$<0

- Therefore, we have $y_i\ (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})$ > 0 always positive
  - then it can act as if it is absolute value ($|\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}|$)

- The **distance between point** (x_{i1}, x_{i2}, x_{i3}) and a **hyperplane** defined by:

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$=0 is:

$$\frac{|\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}|}{\sqrt{\beta_1^2 + \beta_2^2 + \beta_3^2}}$$

Note: |.| is the absolute value
(negative sign ignored)

Problem Formulation of Maximal Margin

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

Find coefficients that maximize the margin M

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

These two constraints ensures that each observation is on the correct side of the plane and at least M distance away

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

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

N = n = number of training points

Can be transformed into a convex optimization problem then solved with convex optimization tools

Same Problem Formulation: Vector Notations

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

$$\text{subject to } y_i(x_i^T \beta + \beta_0) \geq M, \; i = 1, \ldots, N,$$

The term $||\beta|| = \sqrt{\sum_{j=1}^{p} \beta_j^2}$ is the norm of β (vector of coefficients excluding β_0)

Comments on the Formulation

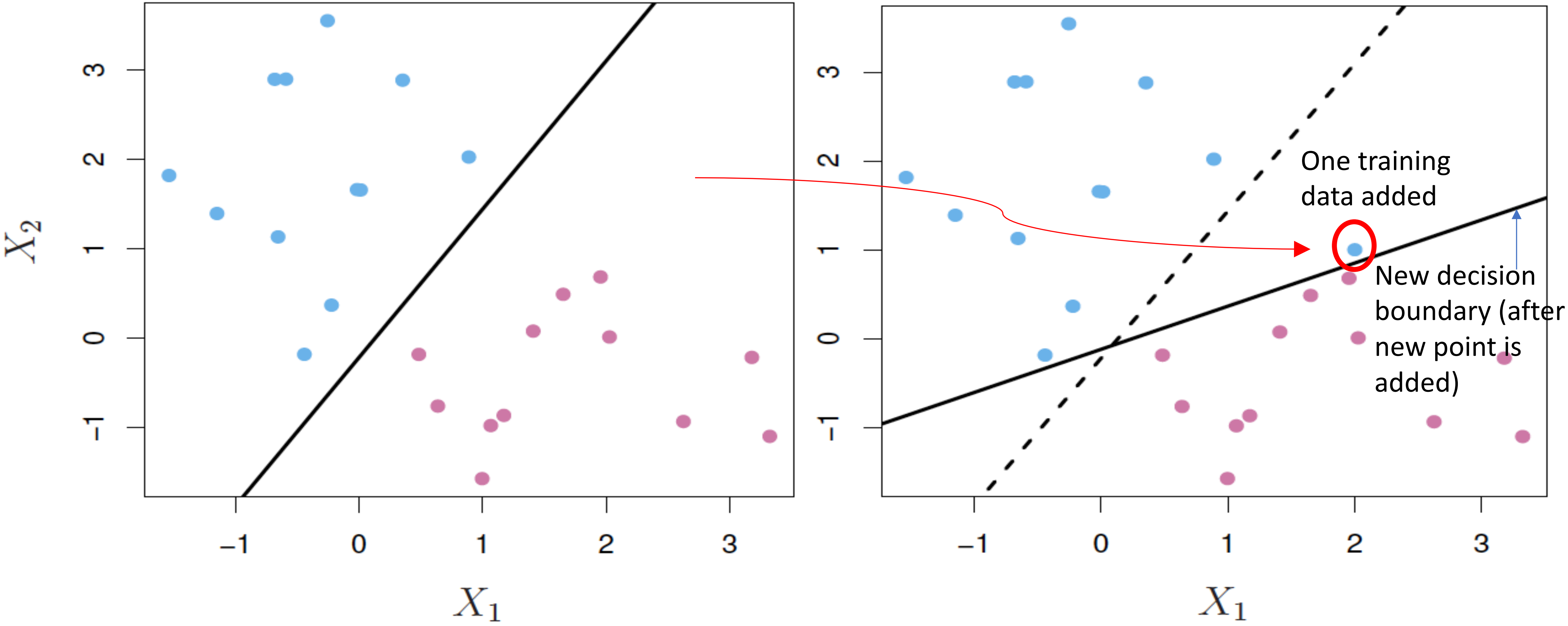
- Other formulation of the optimization problem doesn't include the constraint $\sum_{j=1}^{p} \beta_j^2 = 1$, in this case we have to scale by the norm

- The optimization problem can also be formulated as:

$$\min_{\beta, \beta_0} ||\beta||$$
$$\text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1, \; i = 1, \dots, N,$$

Limitation: Very Sensitive to Training Data

Decision boundary changed dramatically when a training point is added

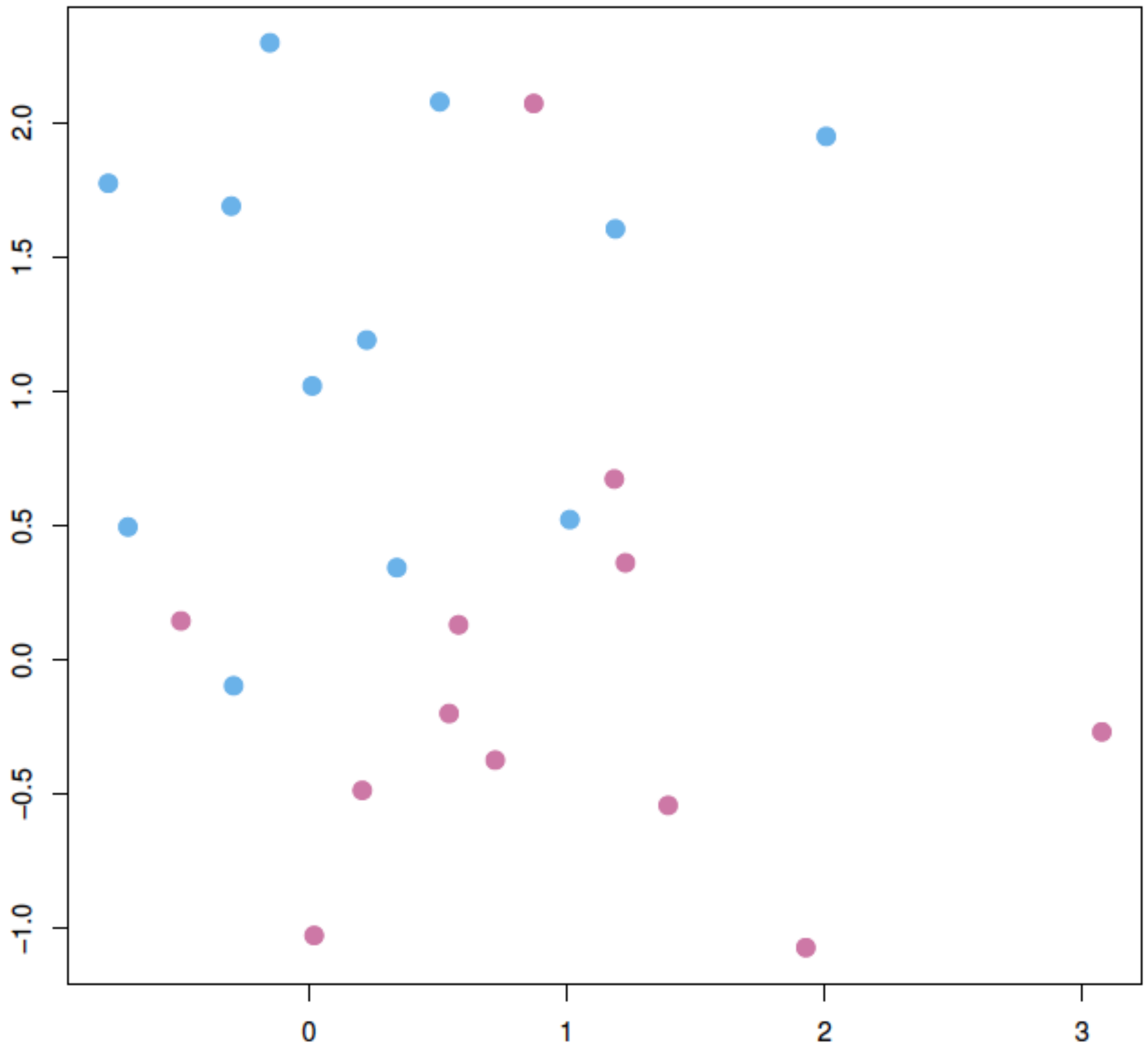


Drawbacks of Maximal Margin Classifier

1. The **sensitivity** to the training observation
2. Classes may not **be perfectly separable by a hyperplane**
   - If classes are not separable? There won’t be any solution for the previously formulated optimization problem

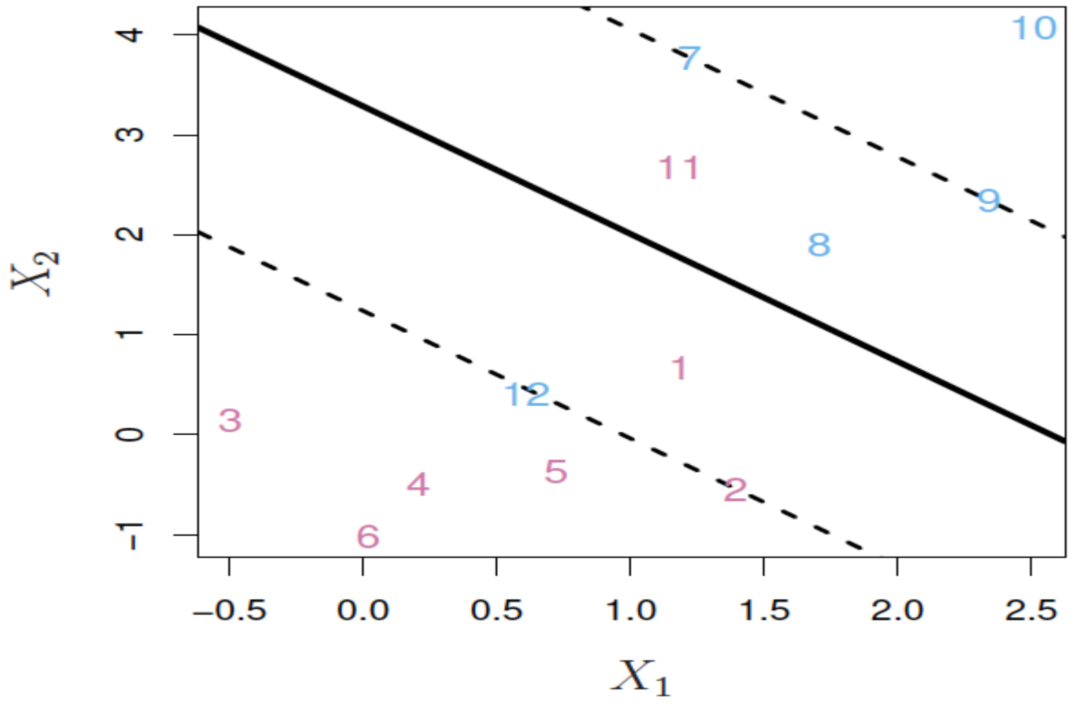
Non Separable Case

- **The support vector classifier is the generalization of maximal margin classifier to the non-separable case**



Support Vector Classifier (SVC)

- In many cases classes will not be perfectly separable
- Support vector classifier allows some training observations to be in the incorrect side of the hyperplane (decision boundary)
- It defines **a soft margin**, which allows observation points to be **within this margin**



1,2,3,4,5,6,11 => Class 1 (red)
Other point => class 2 (blue)

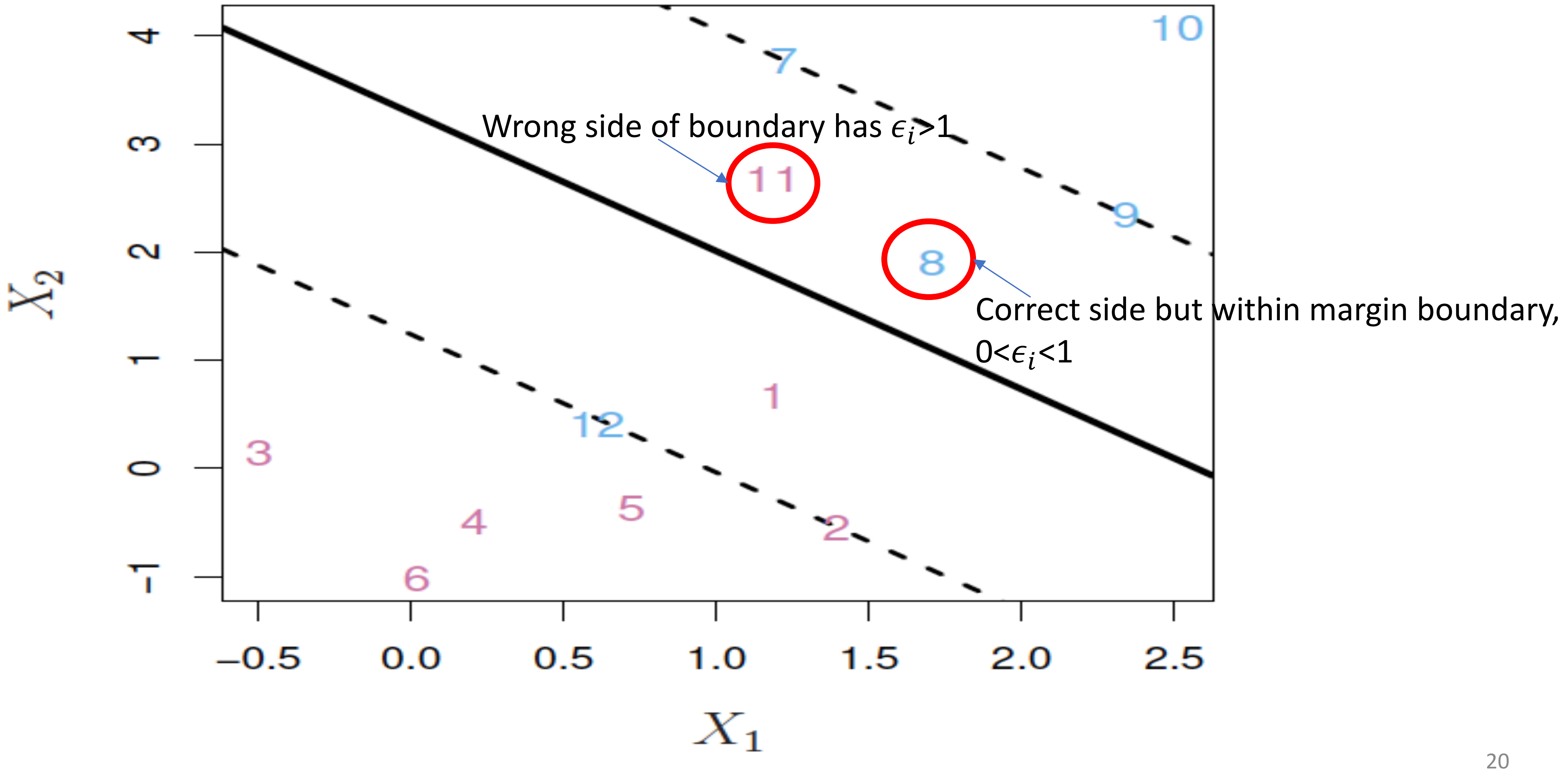
Slack Variables

- Define slack variables: ϵ_i, which depends on the location of the ith training point
- Change condition to

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

  - ϵ_i=0, training point is on the margin (M)
  - $0 < \epsilon_i < 1$, then ith point is within the margin (correct side of the boundary) – violates the margin
  - $\epsilon_i > 1$, then ith point is on the wrong side of the boundary (hyperplane)
    - In this case, $y_i\ (\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)$<0

Slack Variables



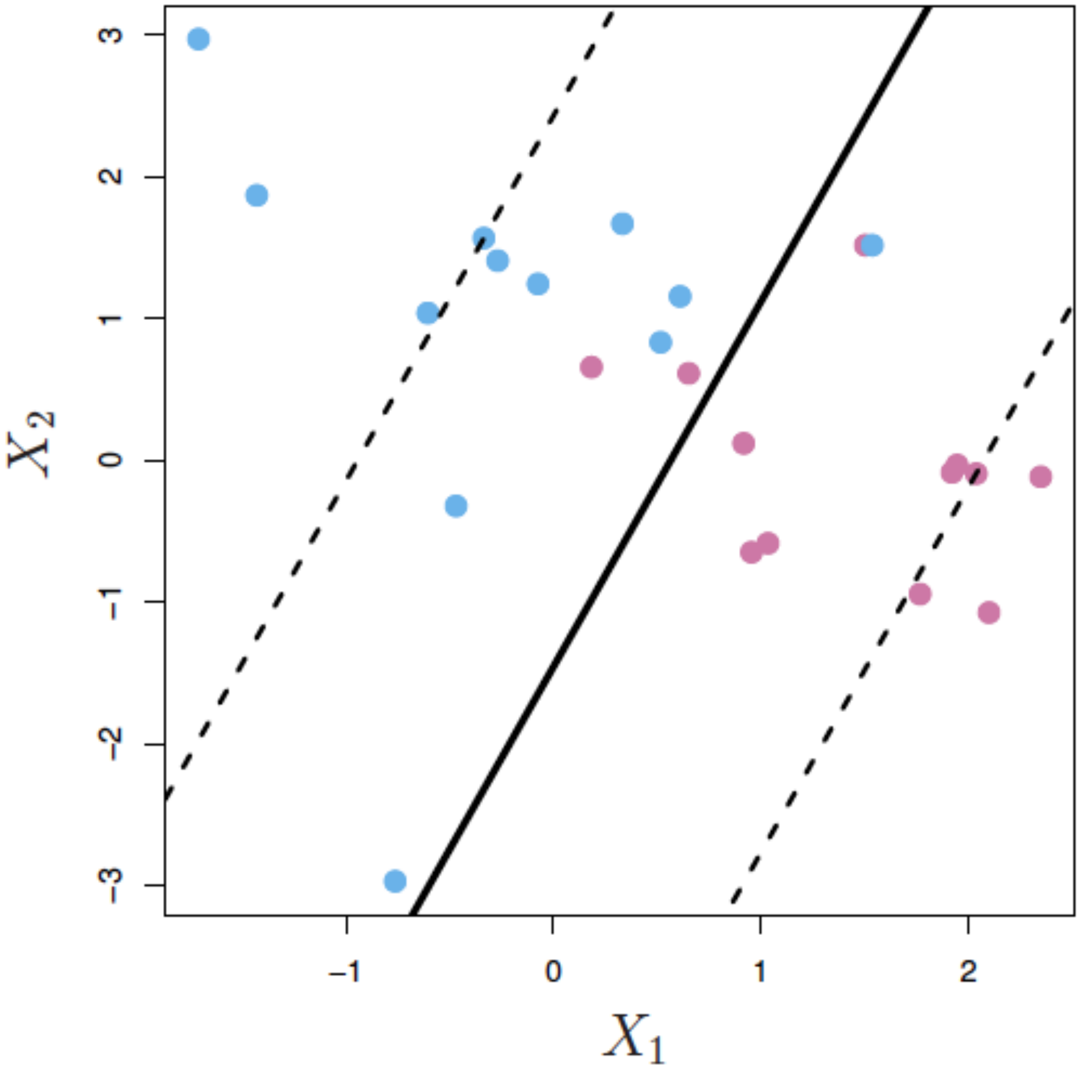
Slack Variables and Regularization

- Put a constraint Constraint: $\sum_{i=1}^{n} \epsilon_i \leq C,$
    - That limits the number the number and severity of violation to margin and boundary

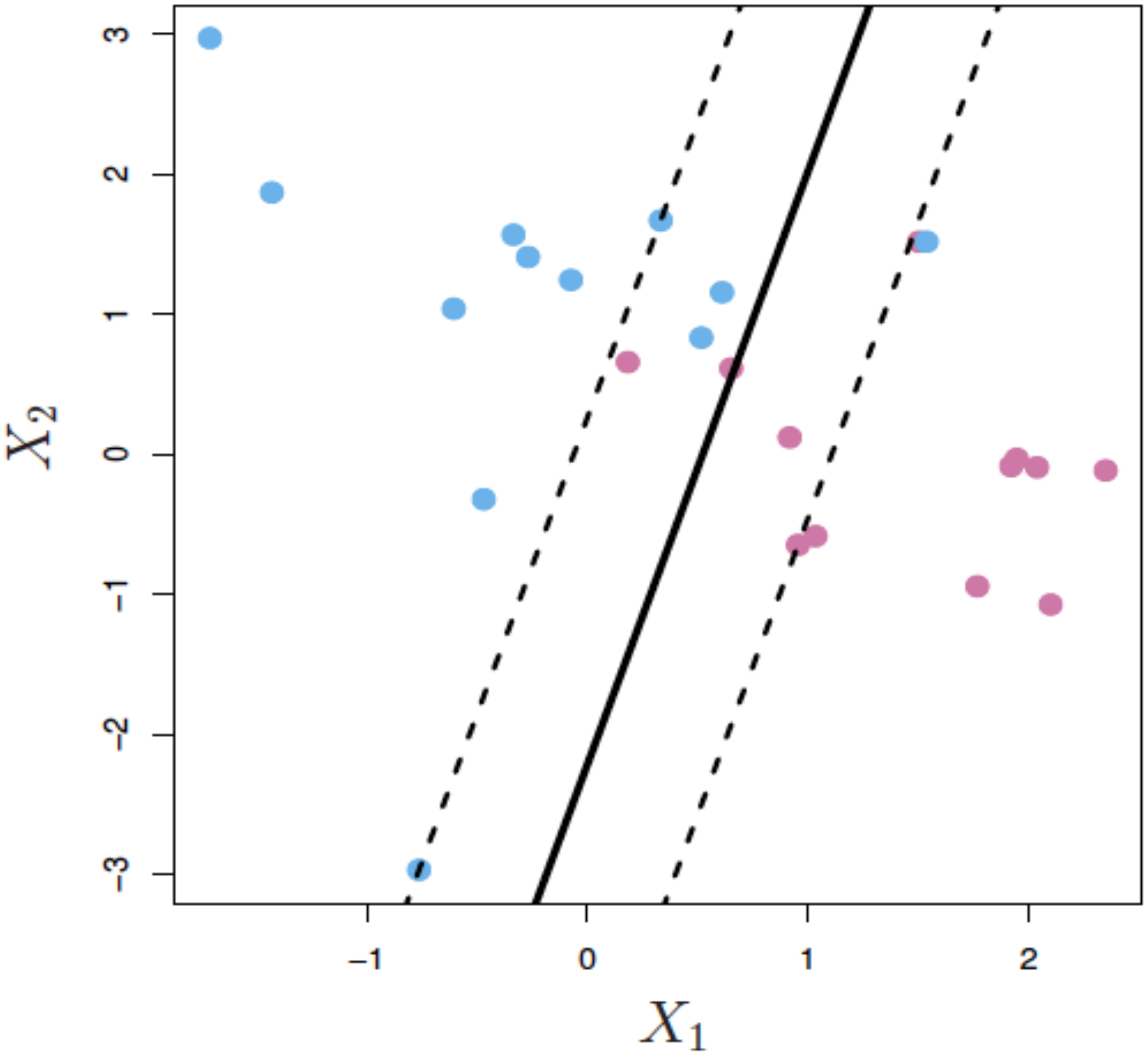
- The constant C is a tuning parameter that we choose (similar to regularization)
    - Large C, more tolerance to violation ➔ wide margin ➔ large bias, small variance
    - Small C, less tolerance to violation ➔ narrow margin ➔ small bias, large variance

C is Regularization Parameter

Large C



Small C



Formulation of Optimization Problem for SVC

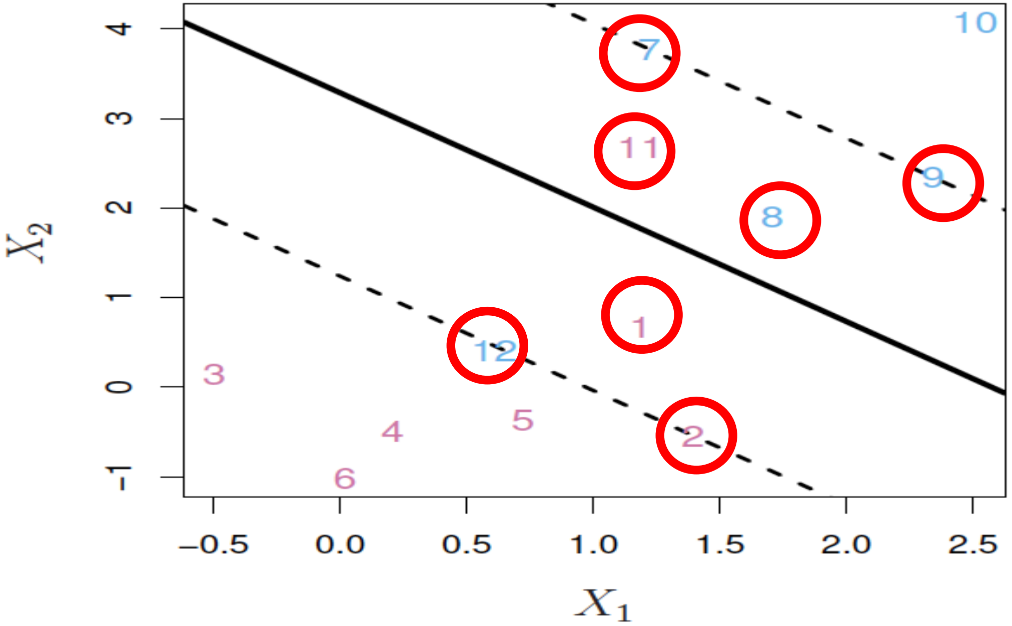
$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \ \sum_{j=1} \beta_j^2 = 1,$$

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

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

Support Vectors

- Training data points that lie on the margin or close to the decision boundary (within the margin) are called support vectors
- Support vectors are hardest to classify as they are closest to the boundary
- The SVC's decision is determined by the support vectors
    - **Therefore decision is based on subset of training samples**



Solution of the SVC

- Solving the optimization problem, it turns out that **SVC decision boundary** can be expressed as as:

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

(derivation is beyond the scope)

number of training samples is n, and x_i is the feature vector of observation i

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

- The solution will have: $\begin{cases} \alpha_i = 0 & if\ xi\ is\ not\ a\ support\ vector \\ \alpha_i \neq 0 & if\ x_i\, is\ a\ support\ vector \end{cases}$
- Hence, instead of summing over all n points, we can sum over $\mathcal{S}$ which is collection of indices of the support vectors:

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

Overall Procedure

- Overall, the SVC classifier is

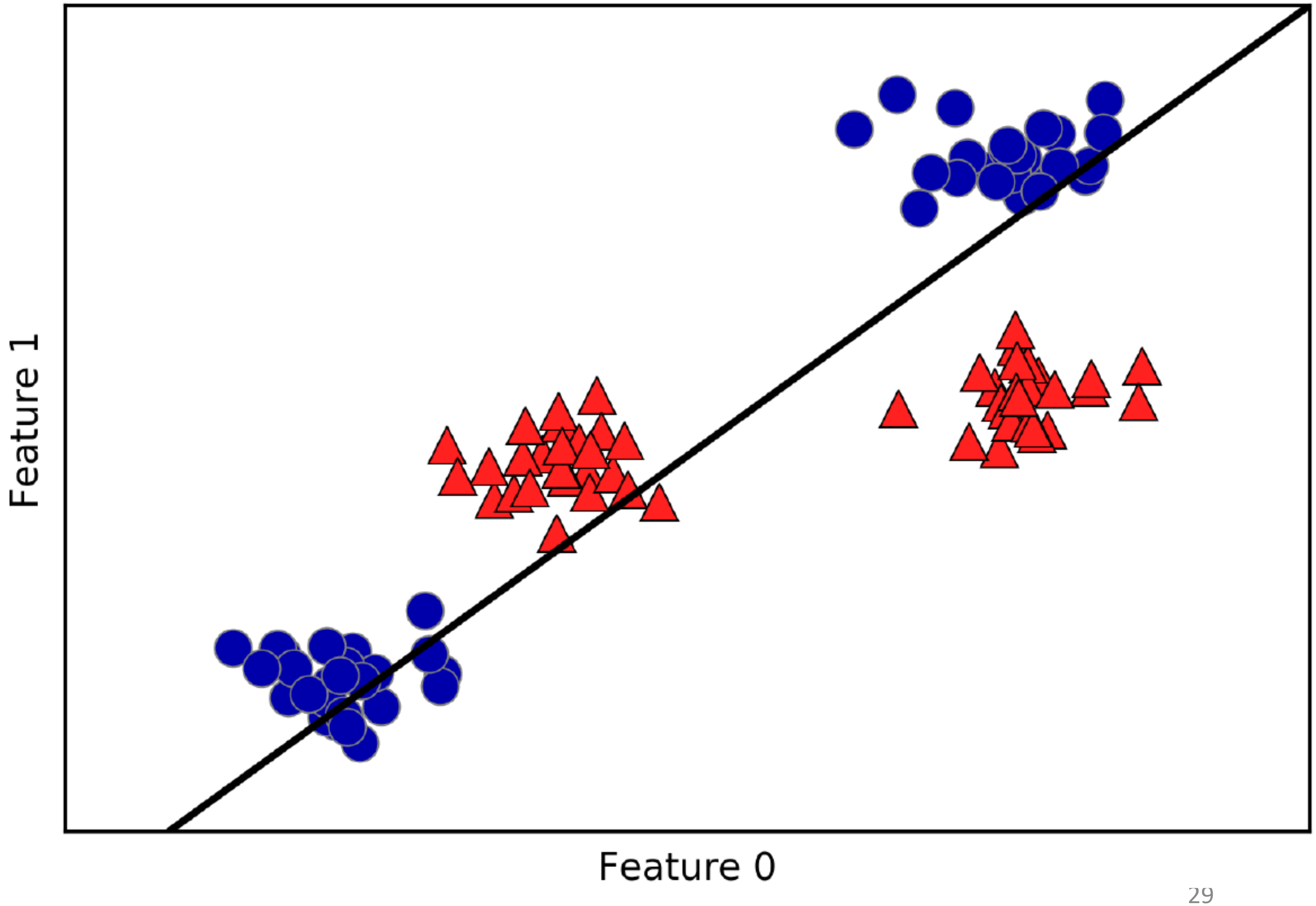
$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \quad = \quad f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

- We get the parameters that solve the optimization problem

- Then classify:
  - Positive class : if $f(X) > 0$
  - Negative class : if $f(X) < 0$

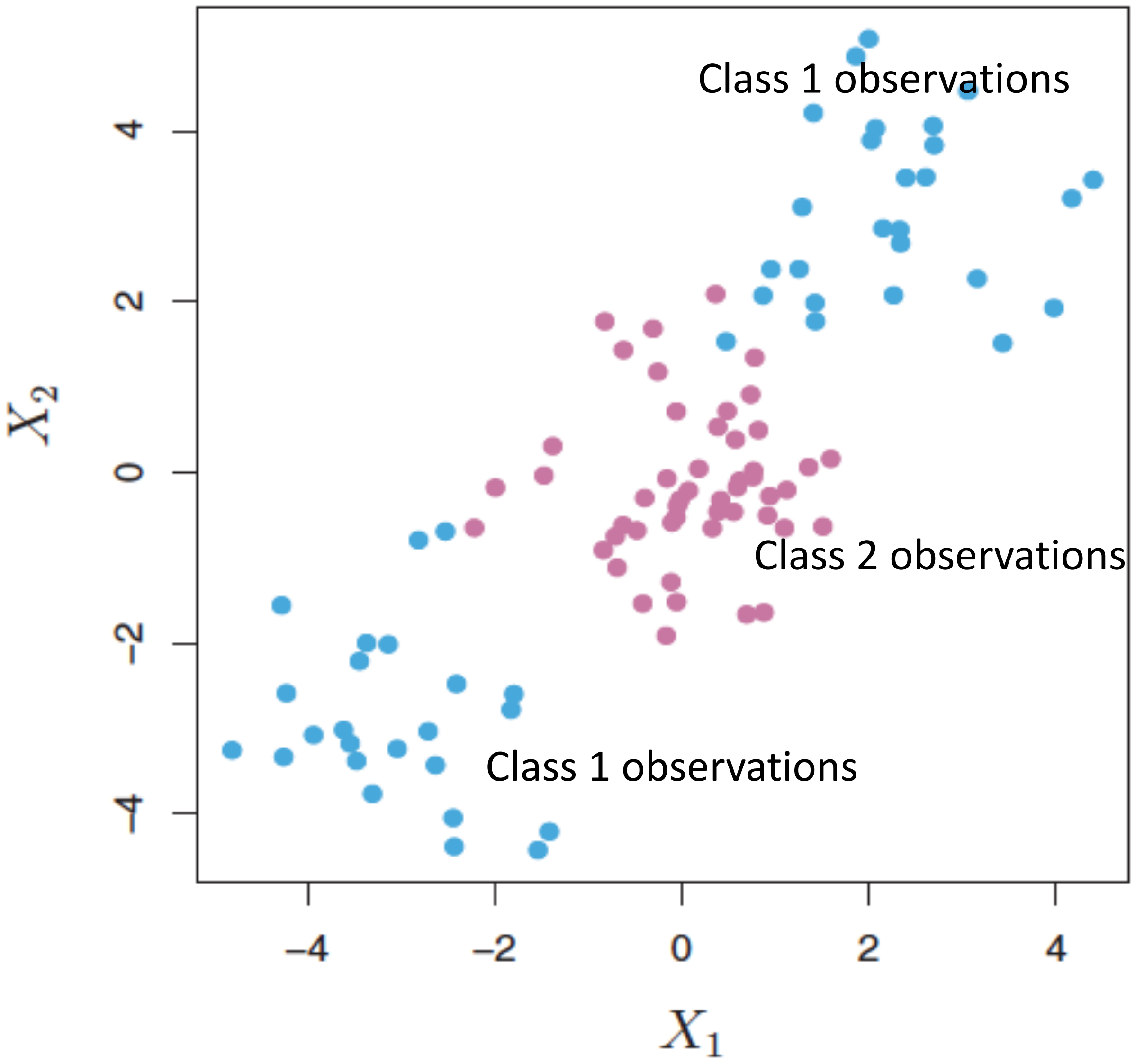
Linear Decision Boundaries Will Not Always Work

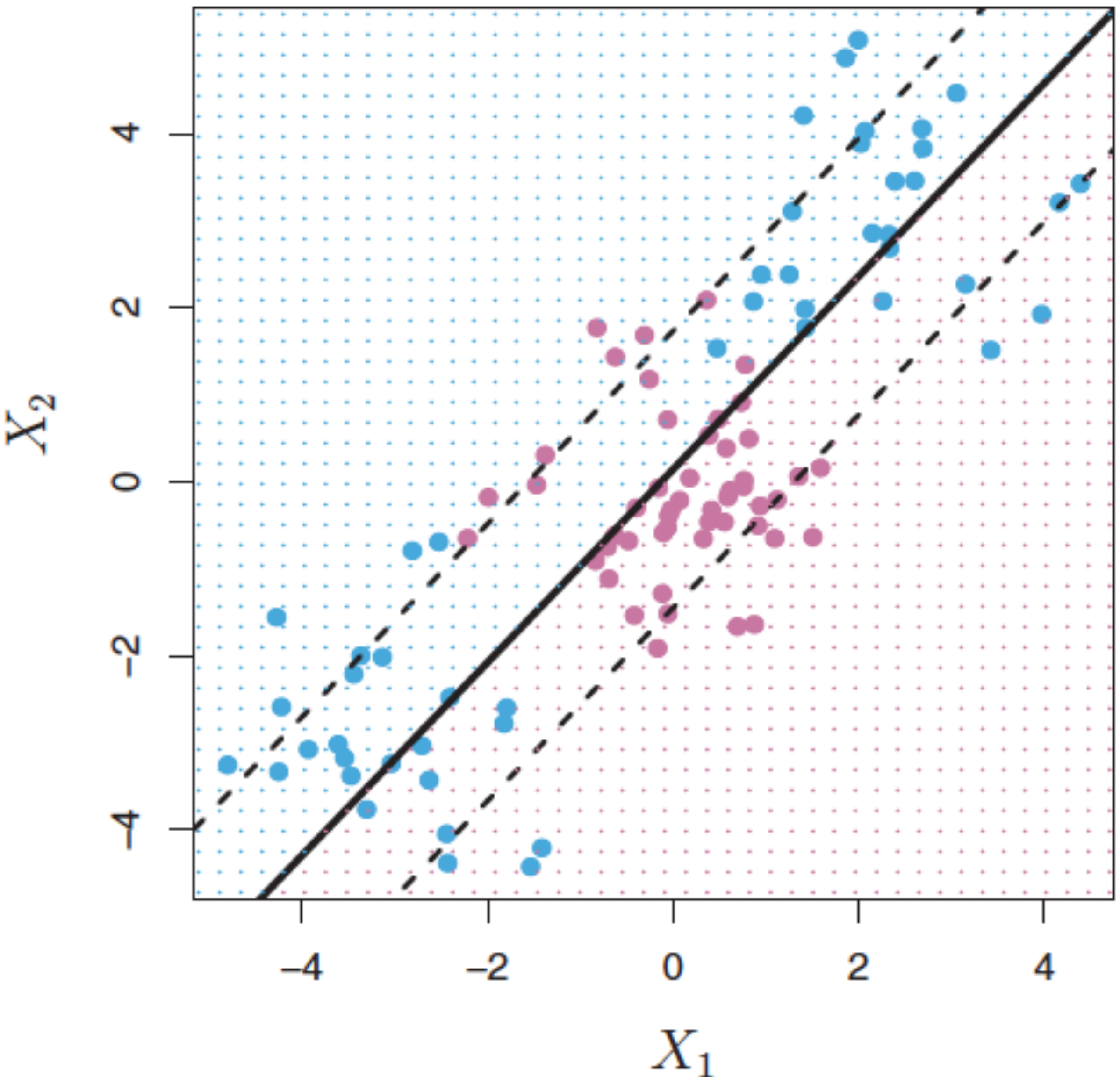
- In some cases, linear boundaries will not work
- The figure shows one example of this case:
  - Class 1 (red triangle) and class 2 (blue circle), a simple linear boundary will not separate them properly



Example:

Linear SVC performs very poorly in this example as it tries to find linear decision boundary





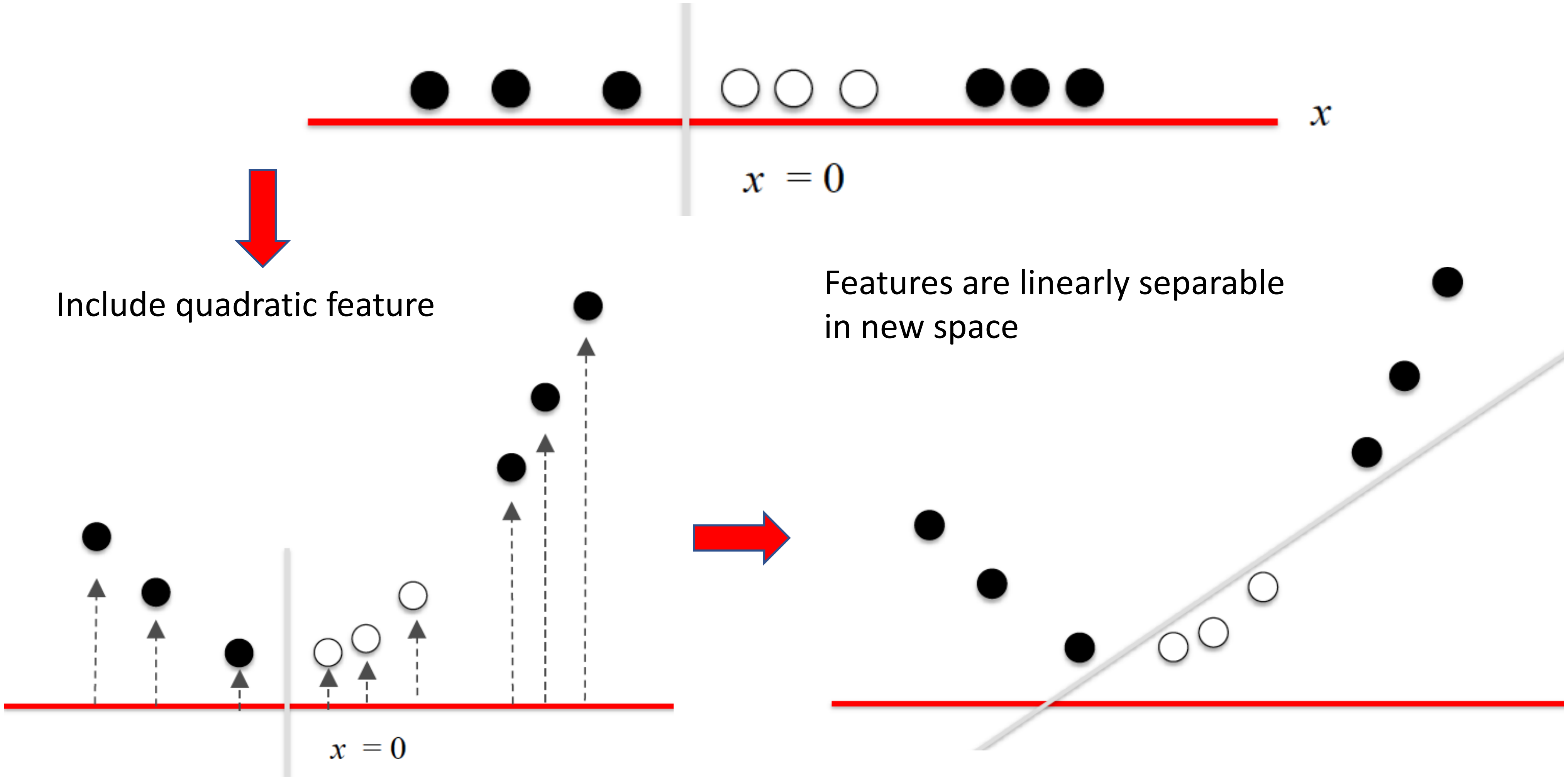
One Possible Solution: Feature Expansion

- We can get non-linear decision boundaries by including polynomial terms and interaction terms (similar to what we did in polynomial regression)
  - Add features like $X_1^2,\ X_1^3,\ X_1X_2,\ X_1X_2^2, \ldots$
  - This will result in non-linear decision boundary in the original space (X_1, X_2)

- Now decision boundary has the form:

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

Example 1 D



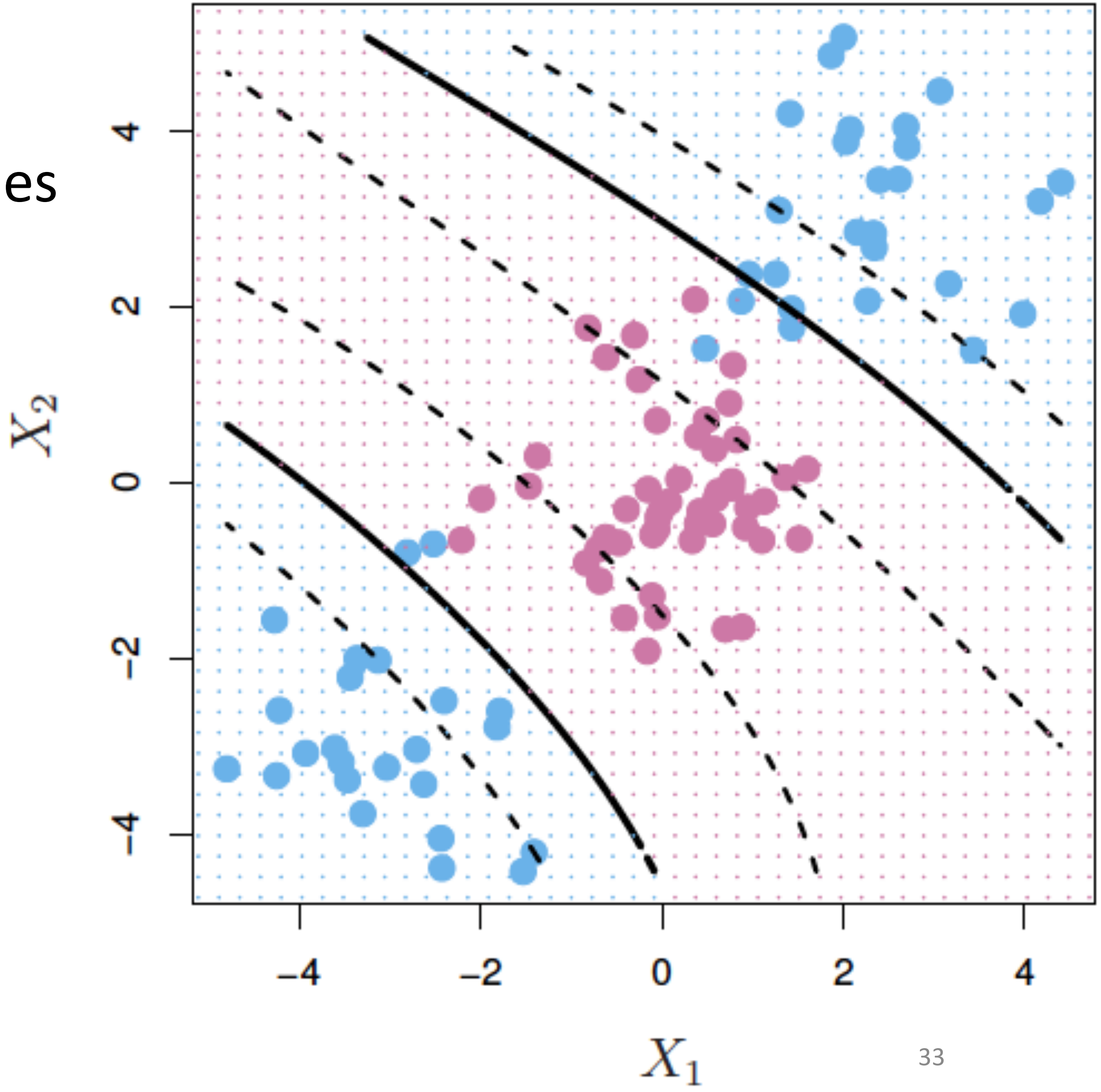
Modified from Kevyn Collins-Thompson

Example

- Original feature space with two variables
- Feature expansion includes 9 variables
- Decision boundary takes the form:

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

- We can get non-linear boundaries as shown



More Efficient Solution: Kernels

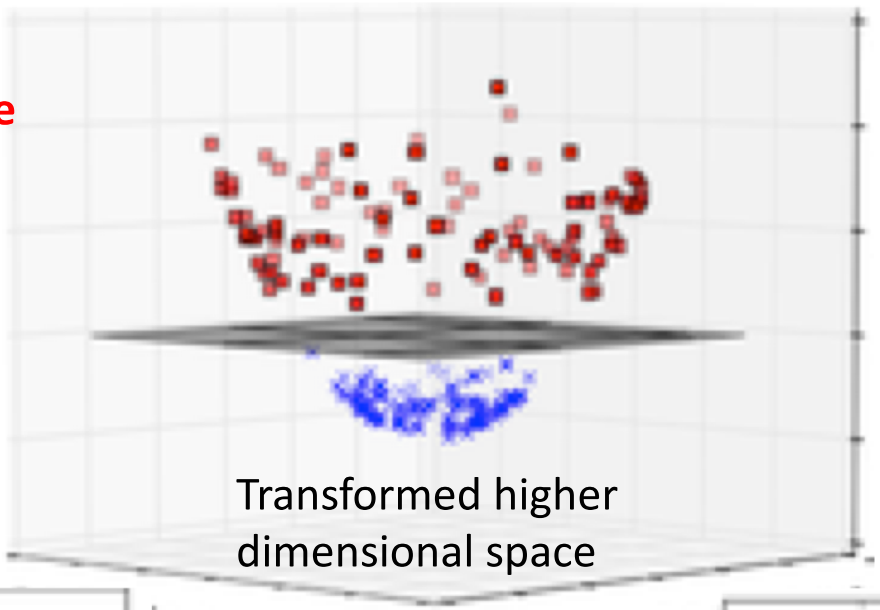
- Feature expansion may need large number of features, and may get complex and computationally inefficient

- **More computationally efficient** way to get non-linear decision boundary is to use **<u>Kernels</u>**

- **Idea:** find a **higher dimensional space** where classes can be separated with a **linear** boundary, but this boundary in **non-linear in the original space**

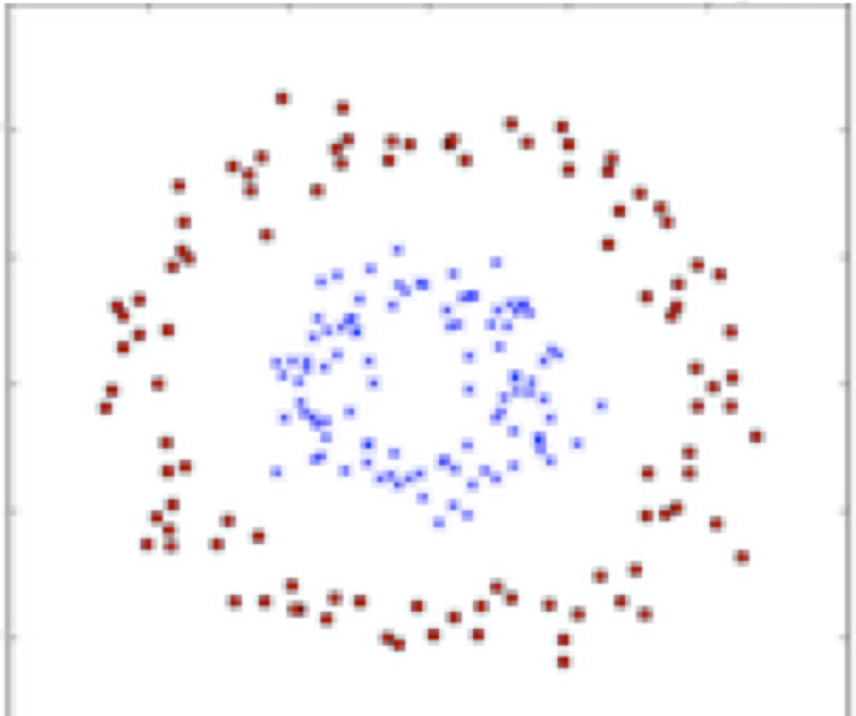
- **Support vector machine (SVM) is extension to the support vector classifier that uses Kernels to learn non-linear decision boundaries**
  - SVM <u>enlarges the feature space using Kernels</u>
  - The new feature space is allowed to get very large **without explicitly defining new features**

Higher dimensional space where classes are linearly separable

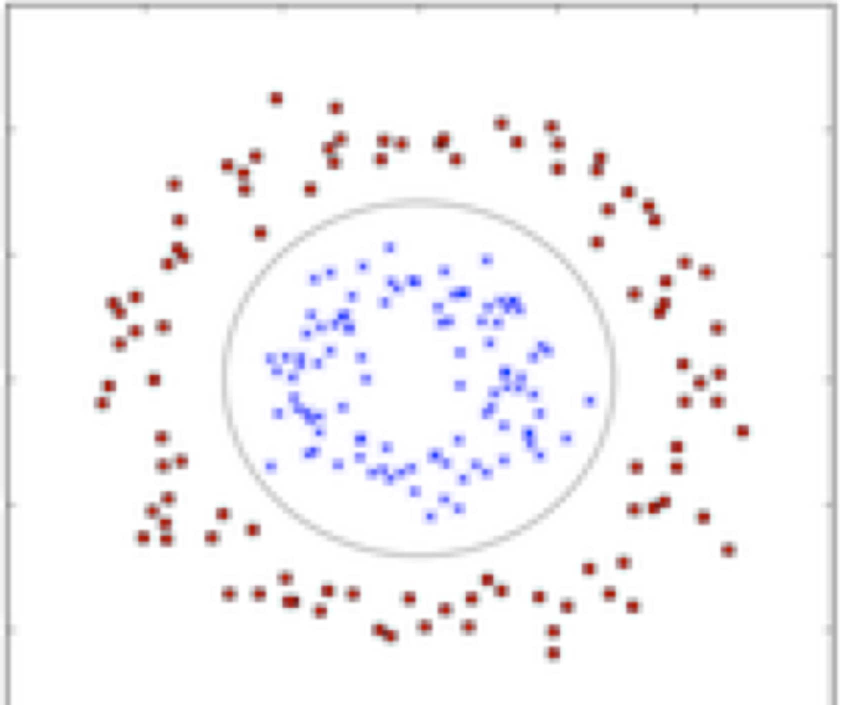


Transformed higher dimensional space

The linear boundary in higher dimension is non linear in the original feature space



Original feature space



Original feature space

[Demo](https://youtu.be/3liCbRZPrZA): https://youtu.be/3liCbRZPrZA

Support Vector Machine

- Recall that in the SVC there is an inner product term (between supports vectors and the new observation)

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

- Support vector machine replaces the inner product in the solution with non-linear Kernel function
    - Use Kernel functions to measure similarity instead of inner product)

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

- **Support vector machine (<u>SVM</u>) is a support vector classifier (<u>SVC</u>) combined with <u>non-linear Kernel function</u>**

Kernels

- **Linear Kernel:** **Gives same solution as linear SVC**

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

- **Polynomial Kernel:** with degree $d > 1$

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

  - Polynomial kernel with degree 2 (d=2) is equivalent to adding features that includes squared of features and all interaction terms .. However, Kernel requires much less computations

Kernels

[Demo](https://cs.stanford.edu/people/karpathy/svmjs/demo/): https://cs.stanford.edu/people/karpathy/svmjs/demo/

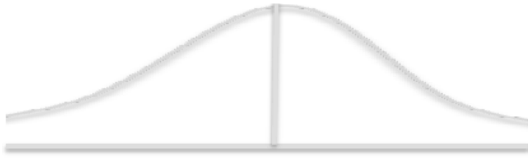
- **Radial Basis Function (RBF) Kernel:** Very popular! Use a Gaussian-like similarity measure

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

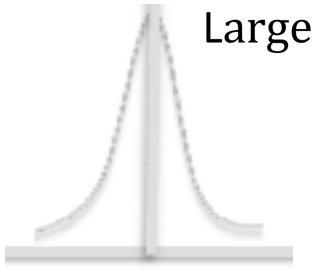
γ is a constant

  - Computes the squared Euclidean distance between the observation point and training point
  - $1/\gamma$ is a constant, regarded as variance… **large γ (small variance) may overfit** as it becomes more local

Small γ



Large γ



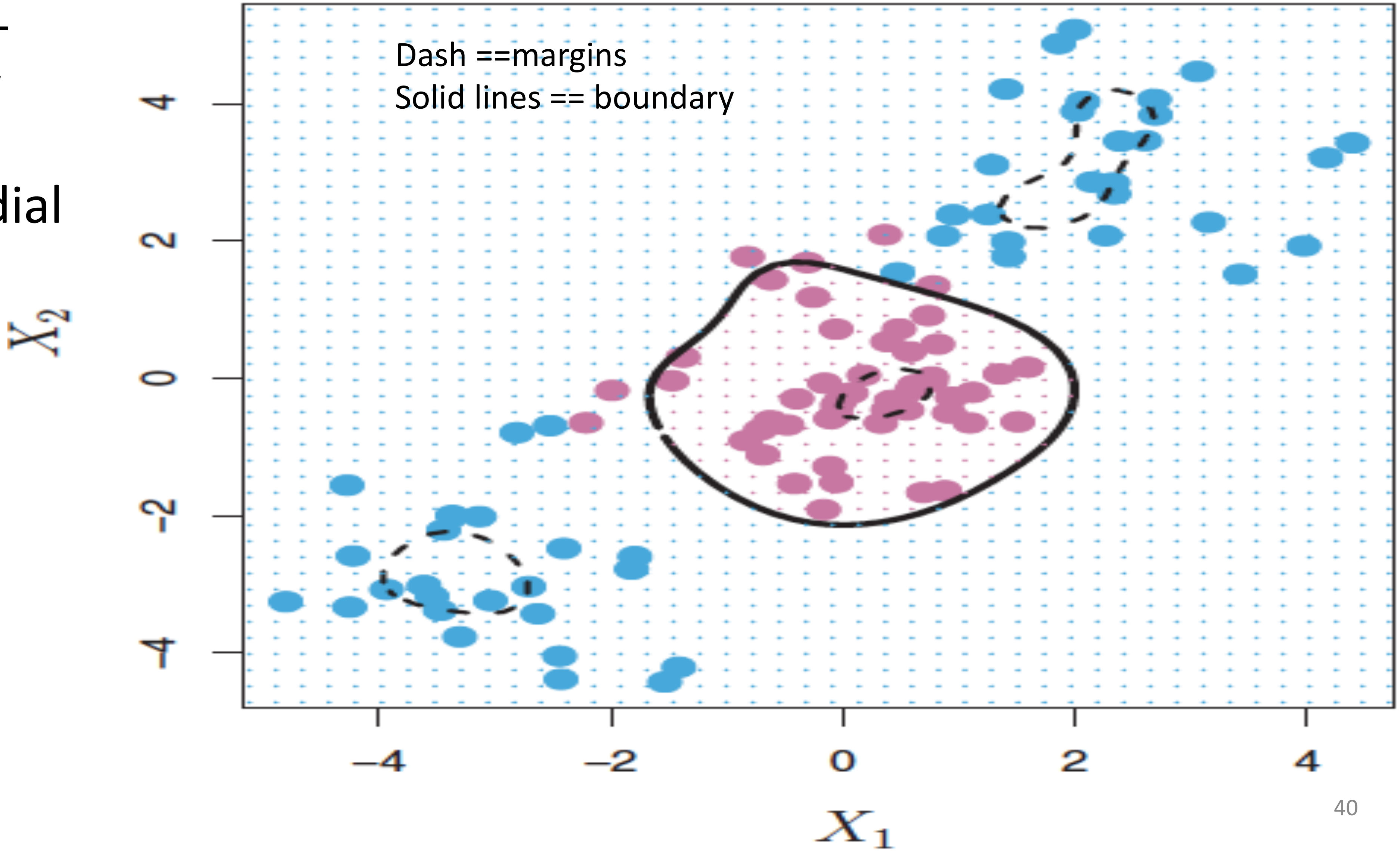
Kernels

- **Sigmoid function:** output ranges from -1 to 1

$$tanh(\gamma \ < x, xi > +c)$$

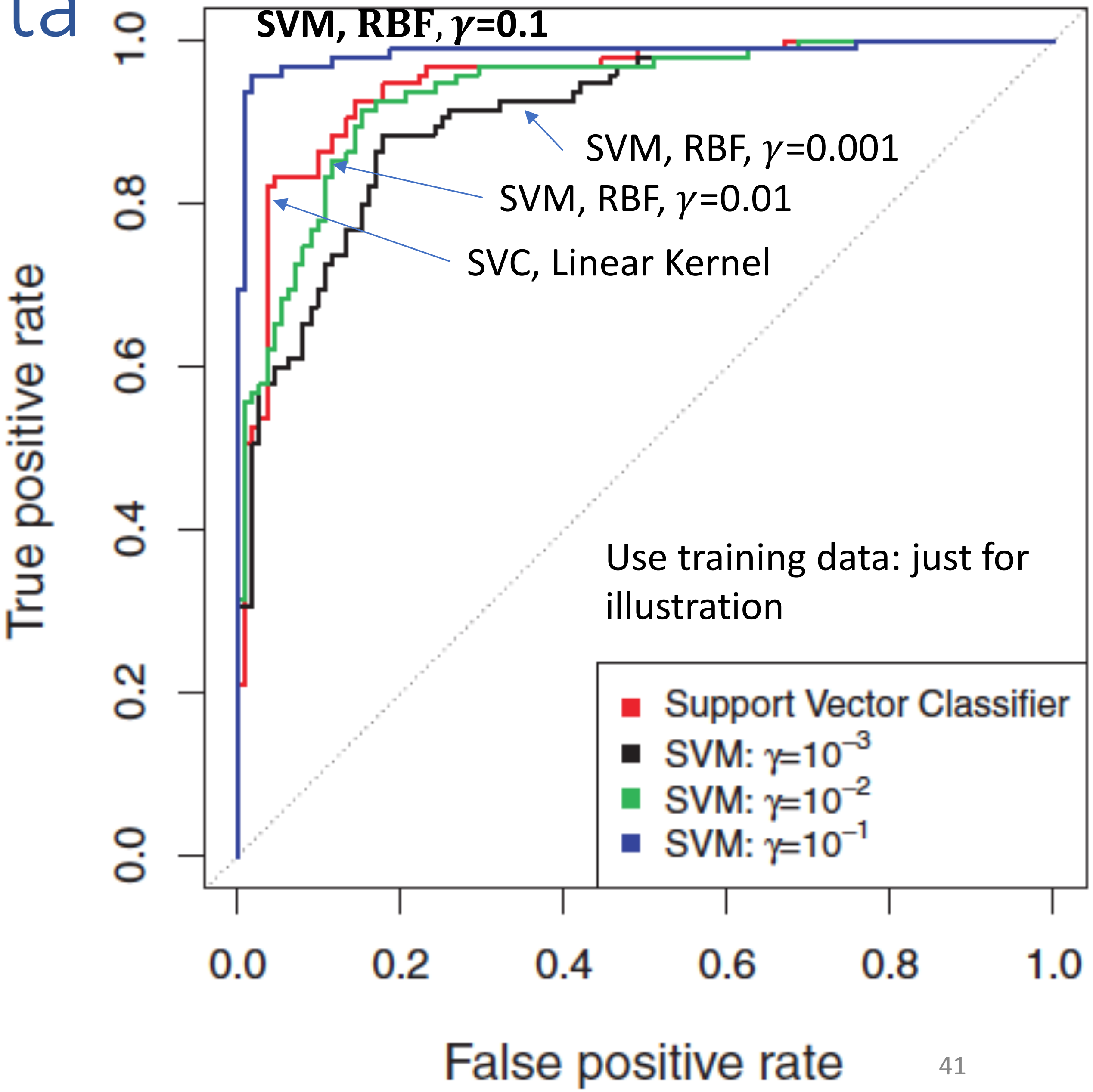
Non-linear Boundaries by Radial Kernel (SVM)

- Example of non-linear boundary that can be captured by Radial Kernel



Example Heart Disease Data

- Predict whether an individual has heart disease using heart disease dataset
  - 13 features: examples age, cholesterol measurement,..
- Classifier learns $f(X)$, then chooses
  - Positive class : if $f(X)$ > threshold t
  - Negative class otherwise
- We can draw ROC by changing threshold t



Multiclass Classification (K Classes)

1-2
1-3
2-3

- **One-vs-one approach: All pairs are compared**
  - Construct K(K-1)/2 classifiers each compares a pair of classes
  - Predicted class is the one that wins the most pairwise competitions
    - Final predicted class is the most frequently assigned class in all pairs

- **One-vs-all approach:**
  - Constructs K classifiers ($f_k(X = x^*)$): each compares one of the classes to the rest
  - If $x^*$is the features of the test observation, then assign it to the class where $f_k(X = x^*)$ is largest

Python Function

- Import and define model:

  From **sklearn.svm** import **SVC**
  svmModel**=SVC(kernel='rbf', gamma=0.1, C=100).fit**

  radial basis function (rbf)

  Gamma is used in the Kernel function

  Regularization/penalty parameter for slack variables ,

- Then use .fit and . score like before
- Kernel can be: 'linear', 'poly', 'rbf', 'sigmoid',.. (default is rbf)
- Details found here: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

- The multiclass support is handled according to a **one-vs-one scheme**.

Summary

- Support vector classifiers (SVC) learn a **linear boundary** that **maximizes a <u>soft</u> margin** between two classes
- The SVC's decision is based on subset of training samples called **support vectors**
- Support vector machines (SVM) are SVC that use **Kernels** to learn **non-linear decision boundaries**.
  - They make SVM learn in a higher dimensional space without explicitly defining new features
    - Kernels calculations can be made efficiently