University of Pittsburgh

ECE 2195: Special Topics – Computers
Machine Learning

Support Vector Classifiers & Support Vector Machines

Mai Abdelhakim, PhD

Assistant Professor of ECE

Swanson School of Engineering

University of Pittsburgh

maia@pitt.edu

Outline

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

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

- Very popular & powerful method for classification
  - SVM becomes popular because of its success in handwritten digit recognition - 1.1% test error rate for SVM (comparable to error rates of a carefully constructed neural network)

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

- SVC provides a linear model for classification
  - **Decision boundary** used to separate the classes in the feature space is **linear function of the features**

- Linear decision boundary can be written as:

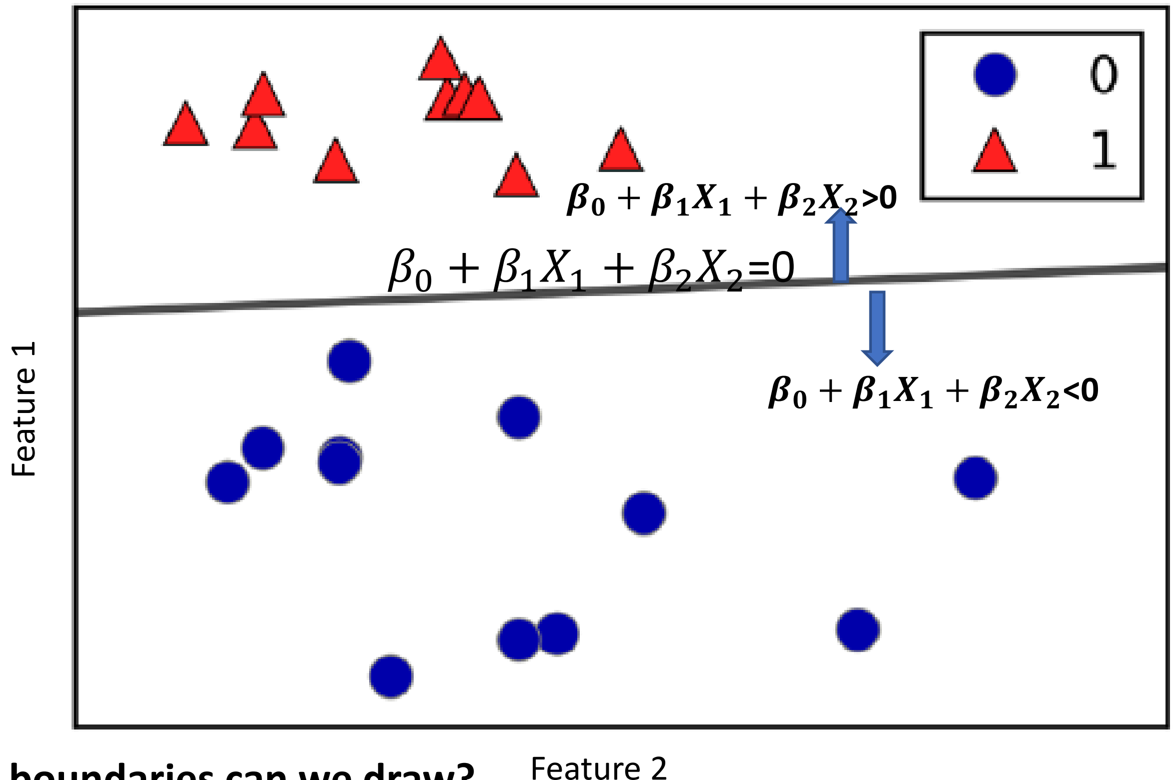
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

  - 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 ($Y_i = +1$) if:** $\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} > 0$
  - **Decide the negative class ($Y_i = -1$) if:** $\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} < 0$

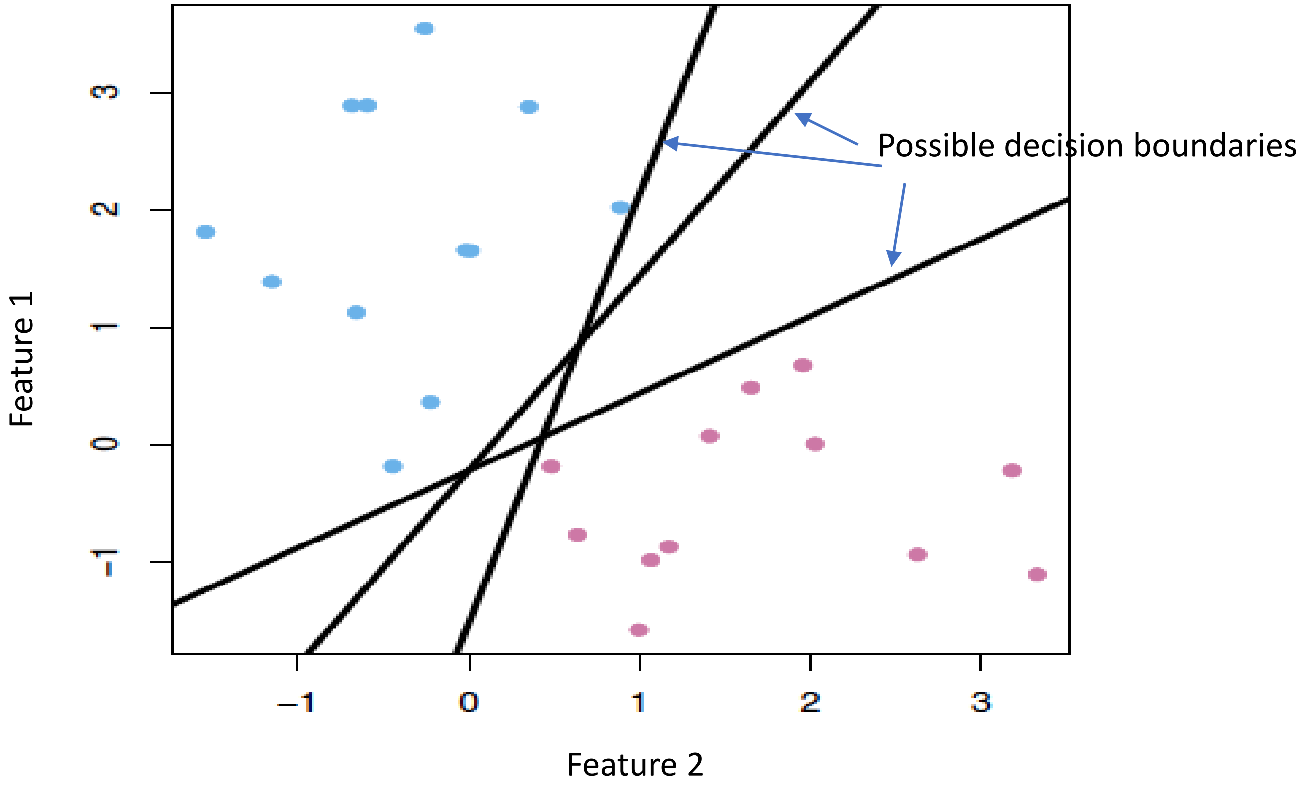
Example of Data Set with 2 Features



How may decision boundaries can we draw?

Infinite Possible Decision Boundaries

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

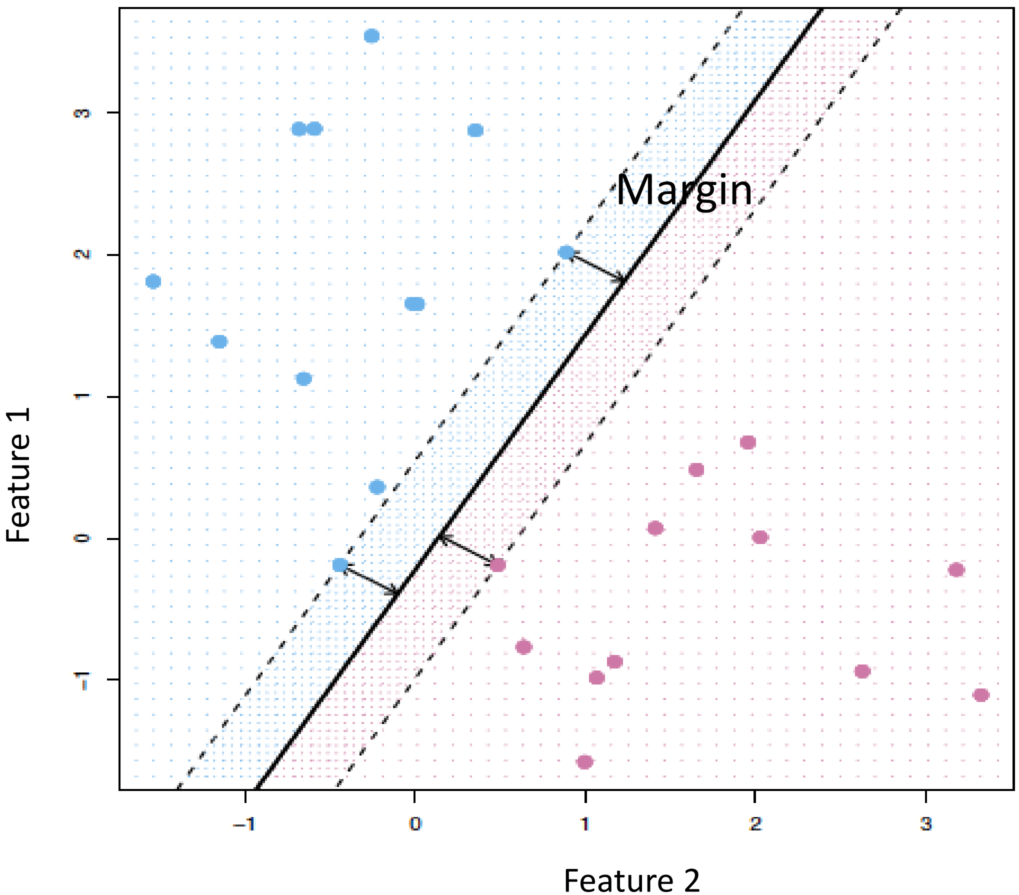


Maximal Margin Classifier

- 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 gap (margin) between the two classes**

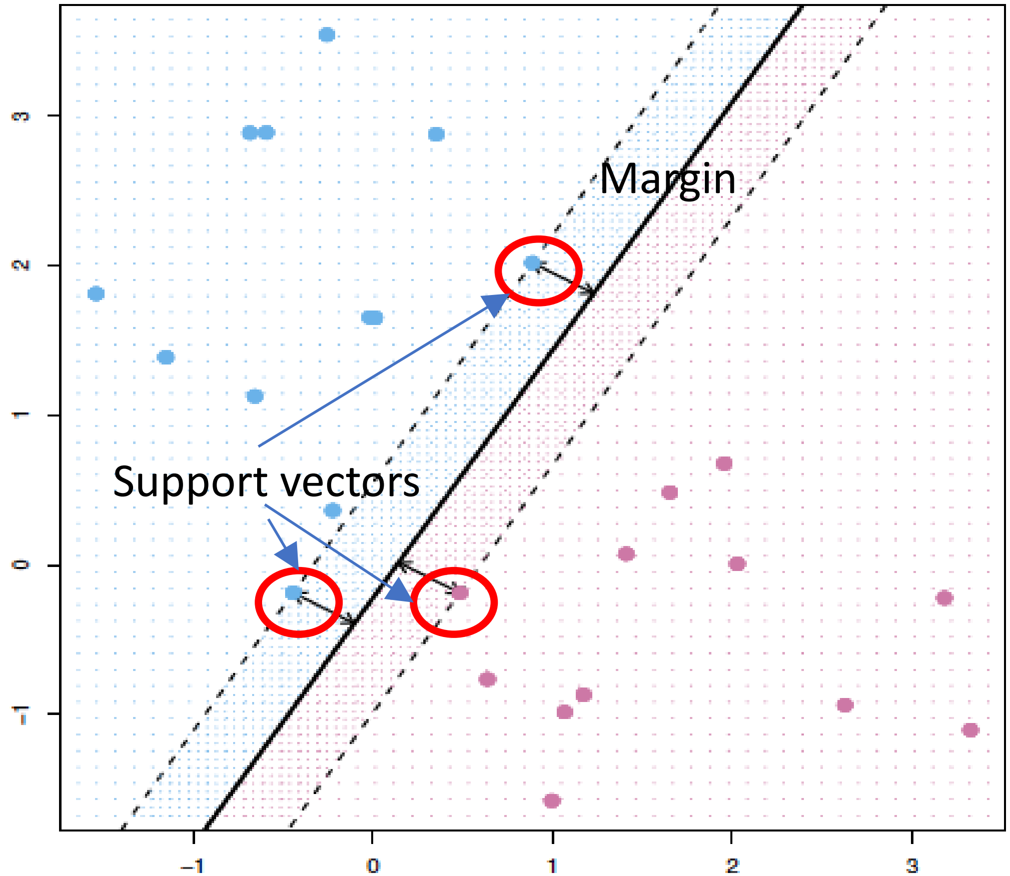
- Choose boundary is the farthest from the training observations



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 -Properties

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

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 $y_i \left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) > 0$ always positive
  - 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:

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

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

Construct the Maximal Margin Classifier

- 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:

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

- Lets say refer to the coefficients without the bias as **$\boldsymbol{\beta}$ = w**, and the bias term $\boldsymbol{\beta}_o = b$
  - Let $\beta_0 = b,\ \beta_1 = w_1, \beta_2 = w_2,\ \beta_3 = w_3$

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

- Then the above distance will be

$$\frac{|w^T x_i + b|}{||w||}$$

Define two planes representing tips of the support vectors

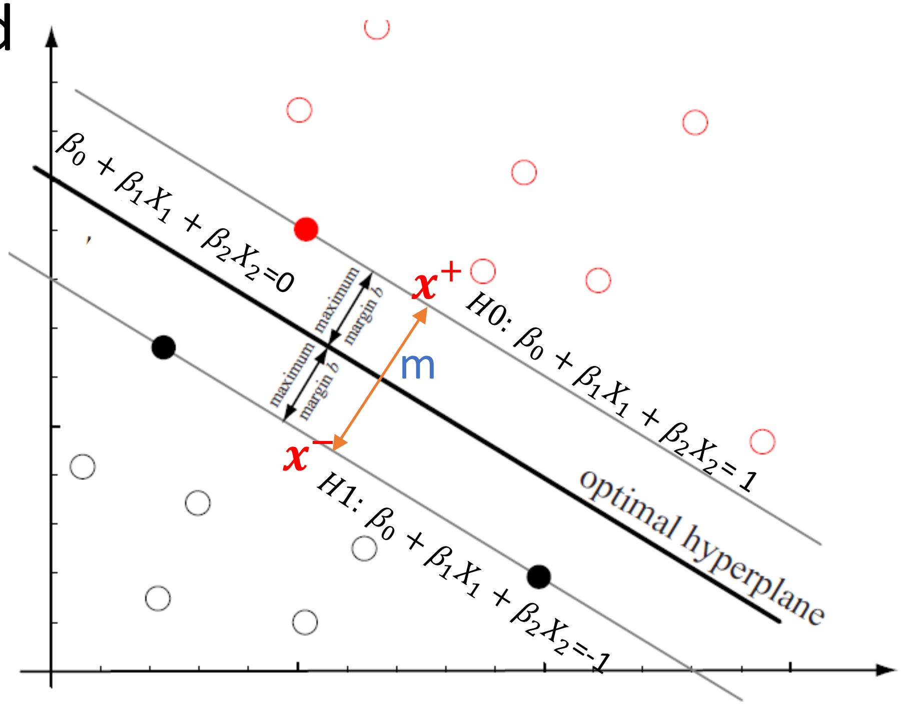
- Margin M should be maximized

$$m=\frac{(x^{+}-x^{-}).w}{||w||} = \frac{2}{||w||}$$

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

  - Distance between $\boldsymbol{x}^{+}$ (on line $w^T x$ =k) & the boundary is k/||**w**||, here k=1

- Maximize the margin, such that all points are classified correctly $y_i\,(w^T x_i + b) \geq 1$



- The optimization problem can also be formulated as

Maximize $\frac{2}{||w||}$

Such that: $y_i\,(w^T x_i + b) \geq 1$

Equivalent to :

Minimize: $\frac{1}{2}.\,||w||^2$

Subject to: $y_i\,(w^T x_i + b) \geq 1$, for all i

Review of Optimization with Constraint

- Assume the objective is

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0$$

- A solution exists if there is $\alpha_i \geq 0$, and solution x_0 satisfy

$$\begin{cases} \underbrace{\frac{\partial}{\partial \mathbf{x}}\left(f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x})\right)}^{\text{Lagrange => its gradient =0}}\Big|_{\mathbf{x}=x_0} = \mathbf{0} \\ g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m \end{cases}$$

- α is the Lagrange multiplier – need one for every constraint
  - $\alpha_i \geq 0$

Back to Out optimization Problem

- Minimize: $\frac{1}{2}.\,||w||^2$
  Subject to: $y_i\,(w^T x_i + b) \geq 1$, for all i

- Equivalent to

Minimize: $\frac{1}{2}.\,||w||^2$
Subject to: $1 - y_i\,(w^T x_i + b) \leq 0$, for all i

$Recall$: $||w||^2 = w^T w$ (doesn’t include the bias)

Solving the optimization (Primal)

- The optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \leq 0$$

- The Lagrangian (optimization function) is

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

- Setting the gradient of the Lagrangian w.r.t w and b to 0

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \boxed{\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i} \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Note:

- Derivative (w.r.t. w) of $w^T w$ is 2 w
- Derivative (w.r.t. w) of $w^T b$ is b

Solving the optimization (Dual)

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- If we subistitute with the w we got

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b)\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b\sum_{i=1}^{n} \alpha_i y_i \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

  - Note that from derivative w.r.t b $\quad \sum_{i=1}^{n} \alpha_i y_i = 0$

- This is function of α_i only

- This is called the **dual problem**
  - **if you know w we know α_i, if you know α_i we know w,**
- We maximize w.r.t α_i

The dual problem

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

- We can find α_i for all i using **quadratic programming (QP)**
  - Specifically sequential minimal optimization (SMO)
    - Each iteration of SMO picks a pair of (α_i, α_j) and solve the QP with these two variables; repeat until convergence

- Then, w can be recovered by

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Dot product between features are measure of similarity

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

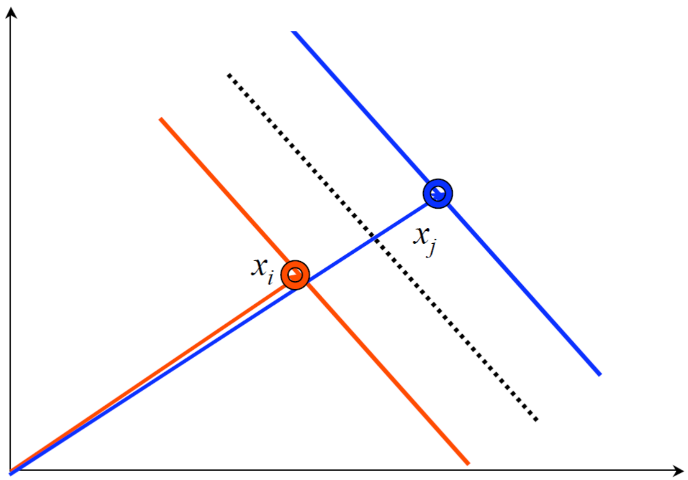
$$\text{subject to } \alpha_i \geq 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Focus on similar features that distinguish between classes!

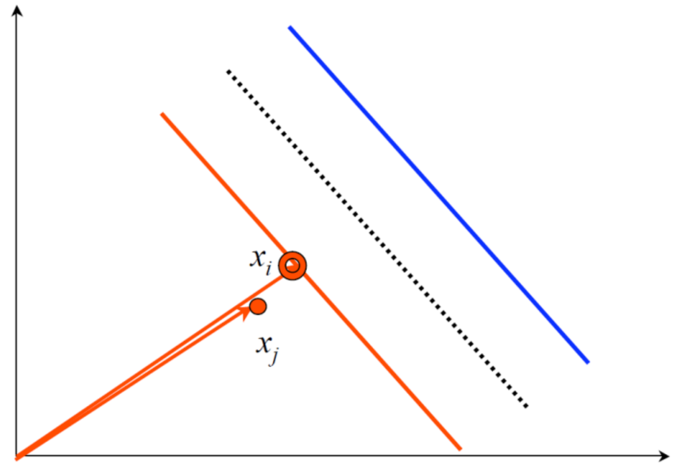
Similar features that distinguish between classes

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i > 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$



Samples that are similar and predict different classes are important



Samples that are similar and predict same class would be redundant (xj is not important)

Ref: Berwick

Characteristics of the solution

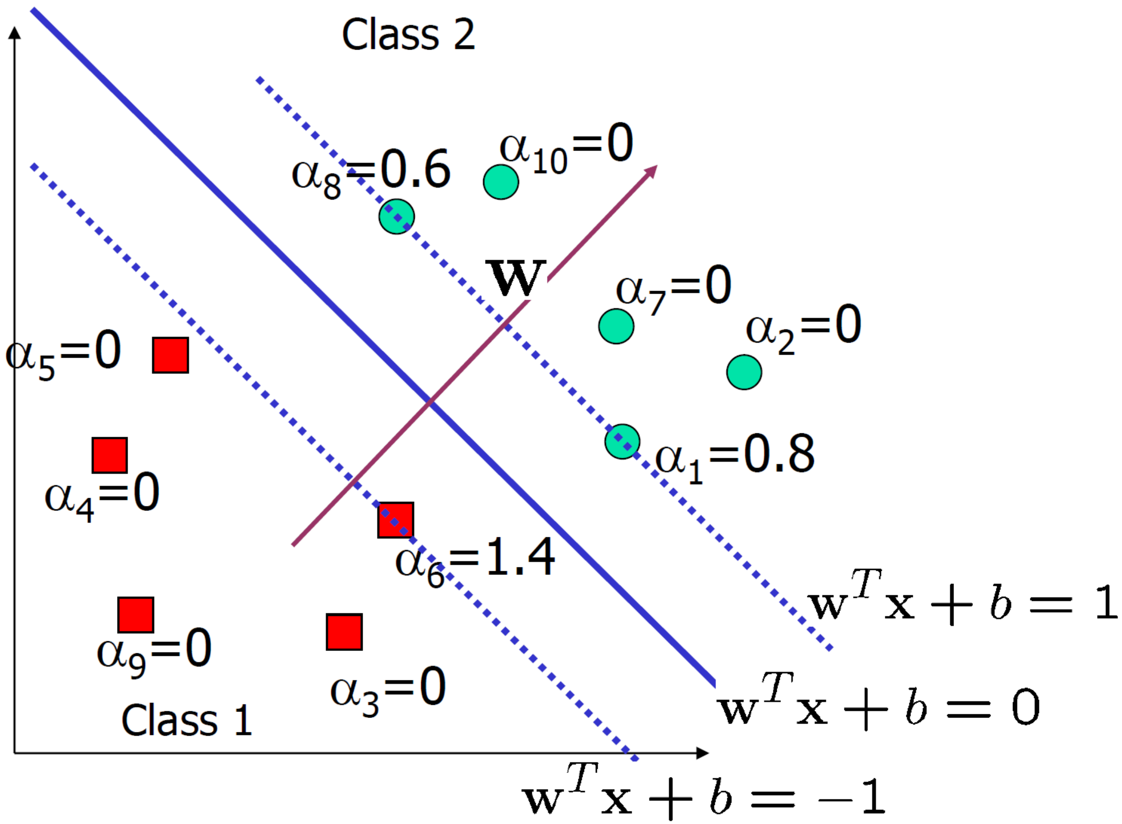
- Objective is

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \le 0$$

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \underbrace{\alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)}_{\text{Denote this term as } g_i(x)}$$

- From the conditions of the optimization: $g_i(x)$=0
  - Since 1- y_i $(w^T x_i + b)$ =0 on the margin -> for all samples on the margin, α_i have non-zero values.
- α_i =0 only for points that are not on the margin.
- Points on the margin has $\alpha_i \neq 0$ and these points are called support vectors

Solution is determined by few training data (support vectors)



Ref: Dr. Huang

The weights & boundary as a function of α_i

- α_i =0 only for points that are not on the margin, and

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- Hence, $\boxed{\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}}$
  - s is the number of support vectors
  - Indices of support vector is tj , j=1,2,…s

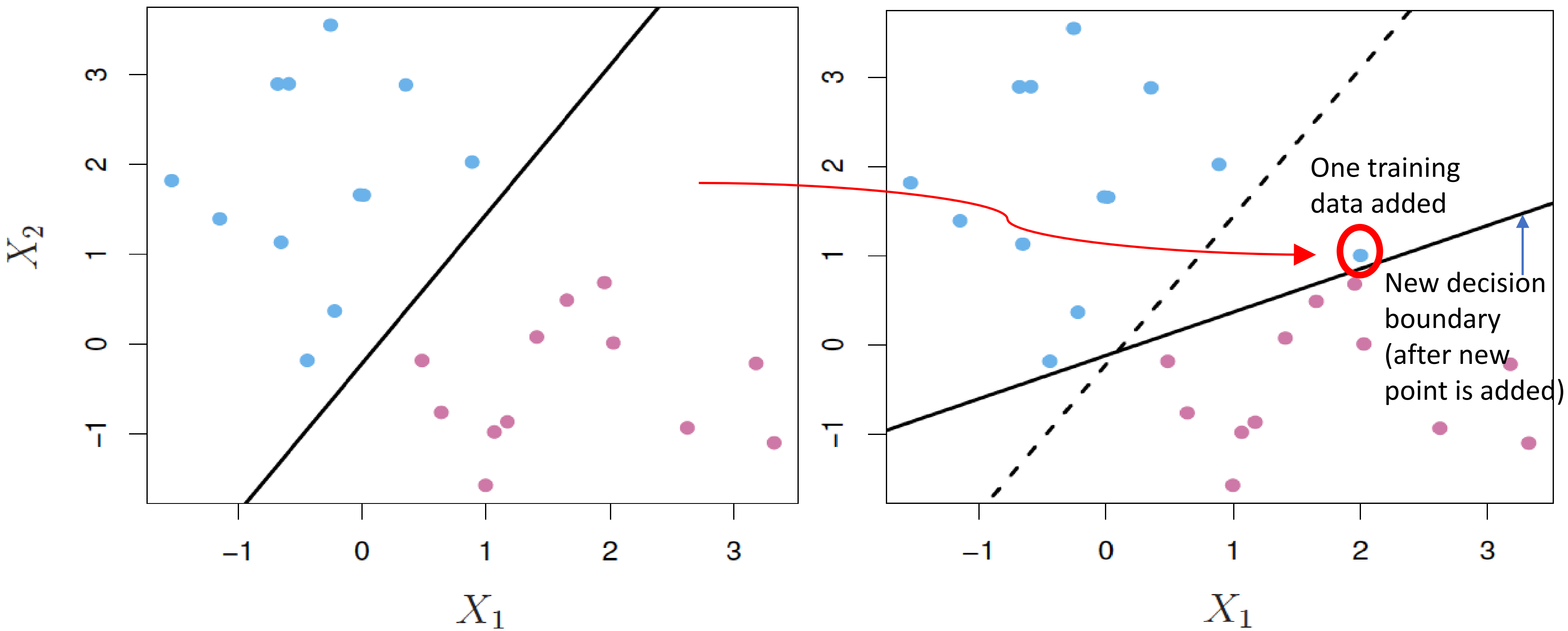
- For any sample z, we compute

$$\boxed{\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \overbrace{(\mathbf{x}_{t_j}^T \mathbf{z})}^{\text{Inner product between support vectors and sample } z \text{ is } < x_i, z >} + b}$$

  - If greater than 0 then decide positive class, otherwise decide negative class

Limitation: Very Sensitive to Training Data

Decision boundary changed dramatically when a training point is added

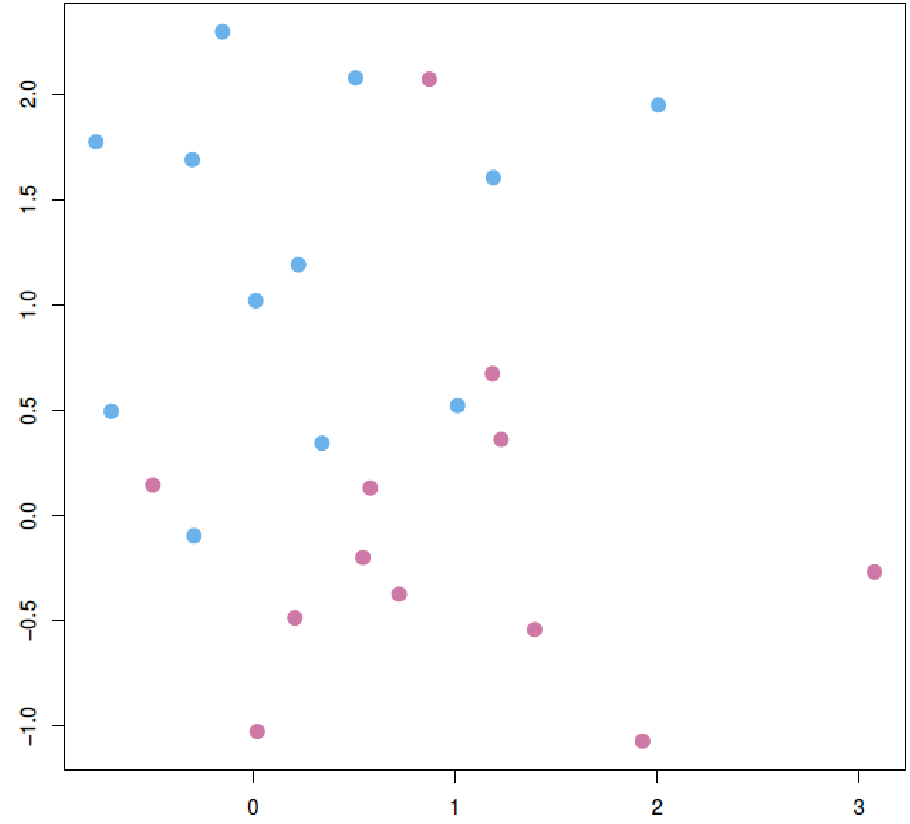


Drawbacks of Maximal Margin Classifier

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

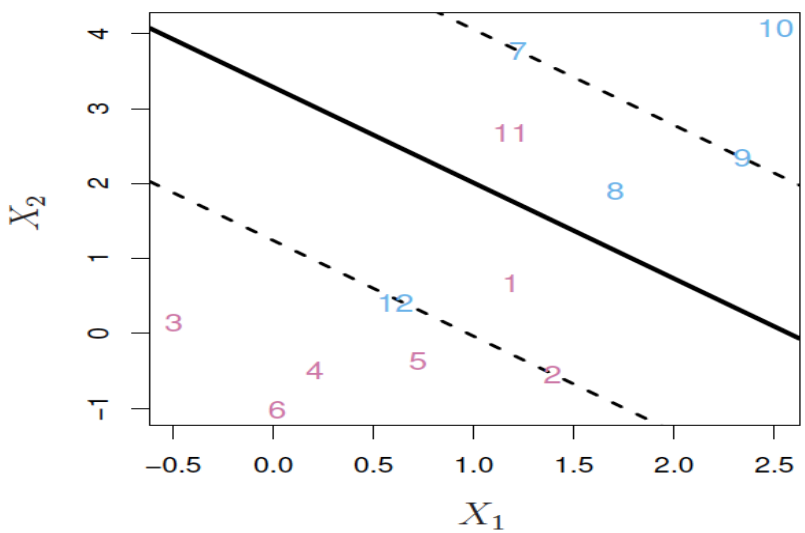
Non Separable Case

- 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)

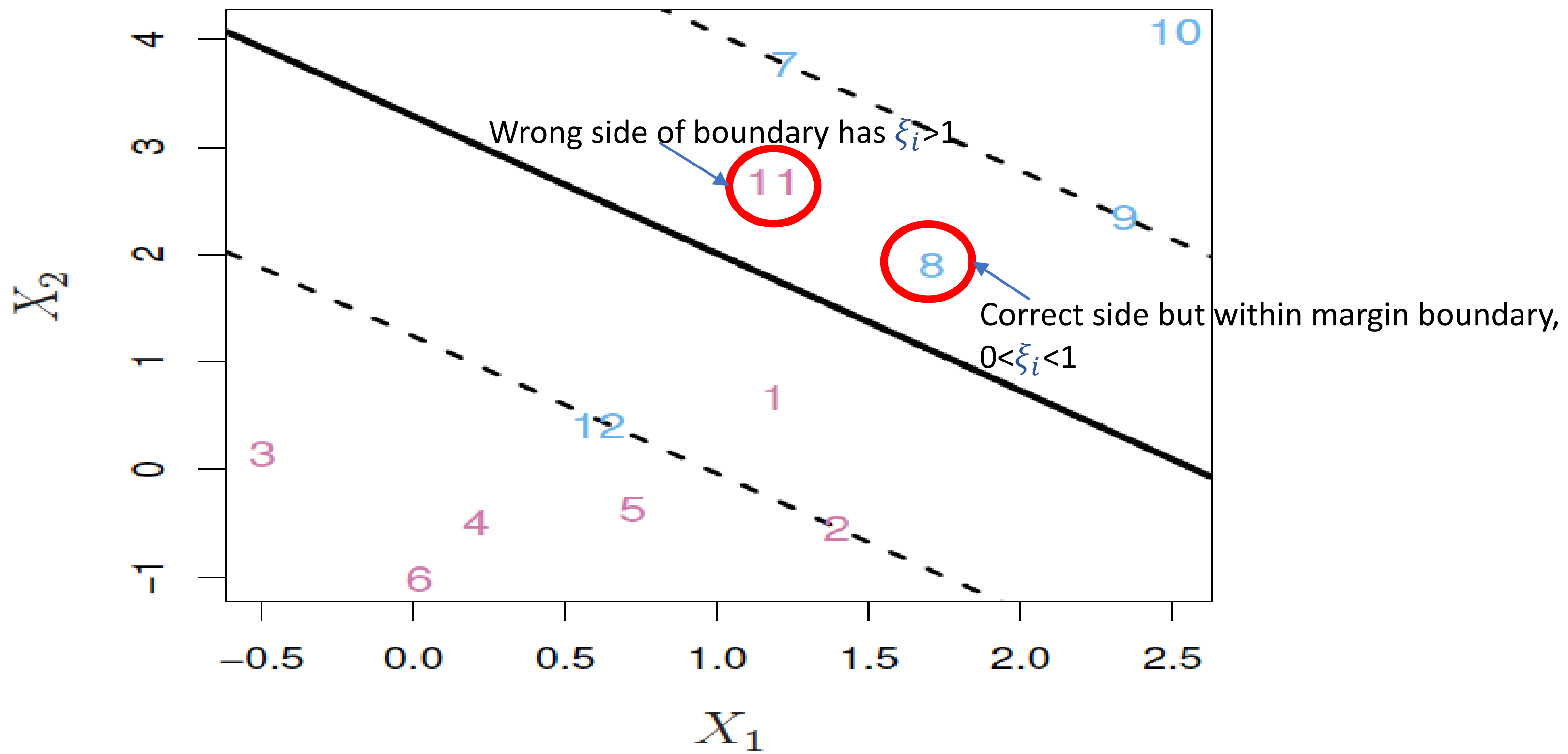
Slack Variables

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

$$y_i\ (w^T x_i + b) \geq 1 - \xi_i$$

  - ξ_i=0, no error for sample i
  - $0 < \xi_i\ < 1$, then ith point is **within** the **margin** (correct side of the boundary) – violates the margin
  - $\xi_i\ > 1$, 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



Reformulate the objective function to maximize the margin and allow some errors (not too much errors)

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- C is similar to the regularization parameter – controls the tradeoff between the margin and the slack penalty (minimize training error vs model complexity)
  - Large C => allow zero errors
- There will be upper bound on $\alpha_i \leq C$, but solution will be similar

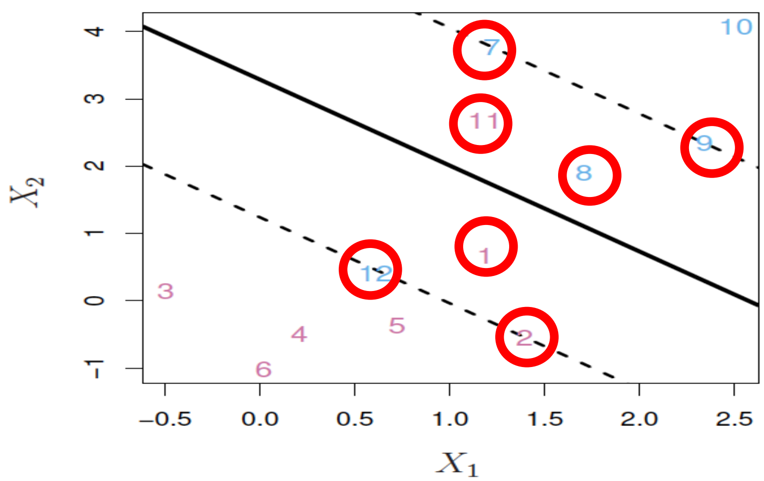
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

Bishop , Chapter 7

Support Vectors

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

- 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
  - **<u>Therefore decision is based on subset of training samples</u>**



Solution of the SVC

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

$$f(x) = b + \sum_{i=1}^{n} \alpha_i y_i \left(x_i^T x\right) + b$$

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

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

Dot product (inner product):
between x and points in training sample

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

$$f(x) = b + \sum_{j=1}^{s} \alpha_{tj} y_{tj} < x_{tj}, x >$$

Overall Procedure

- Overall, the SVC classifier is

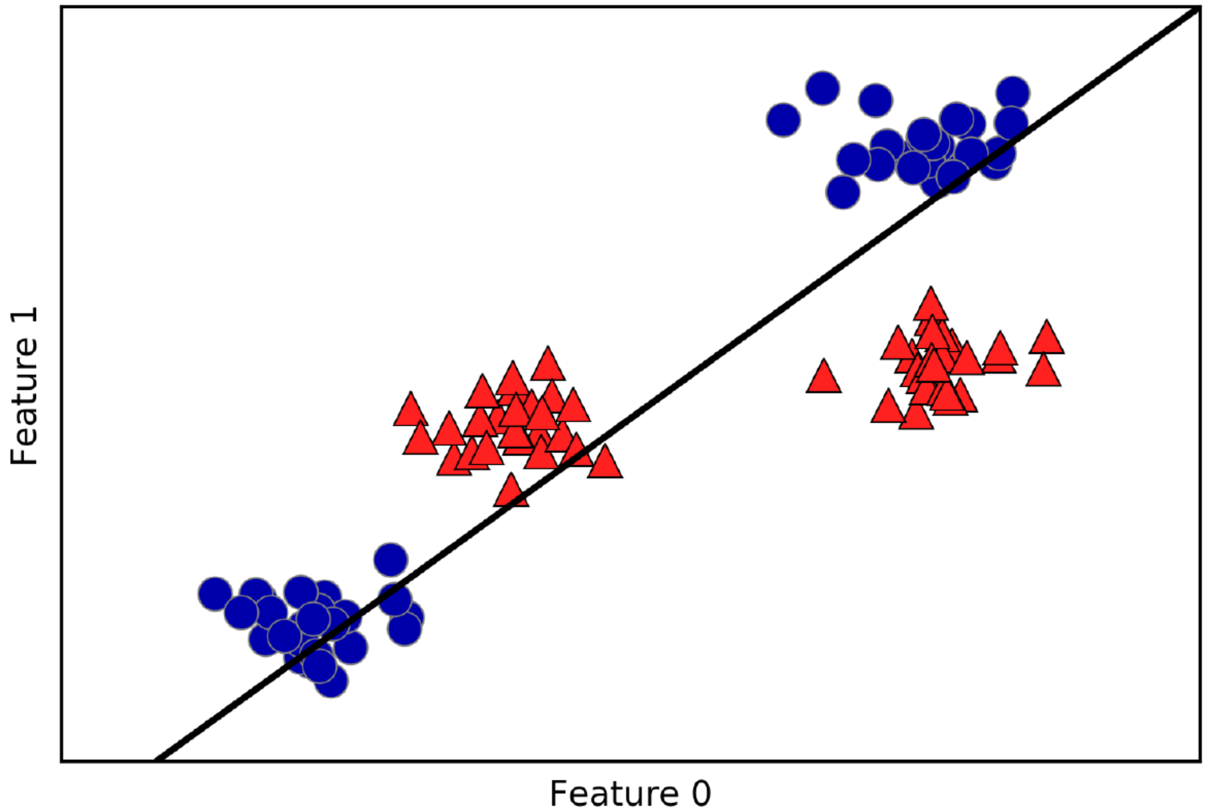
$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = f(x) = b + \sum_{j=1}^{s} \alpha_{tj} y_{tj} < x_{tj}, x >$$

- We get the parameters that solve the optimization problem

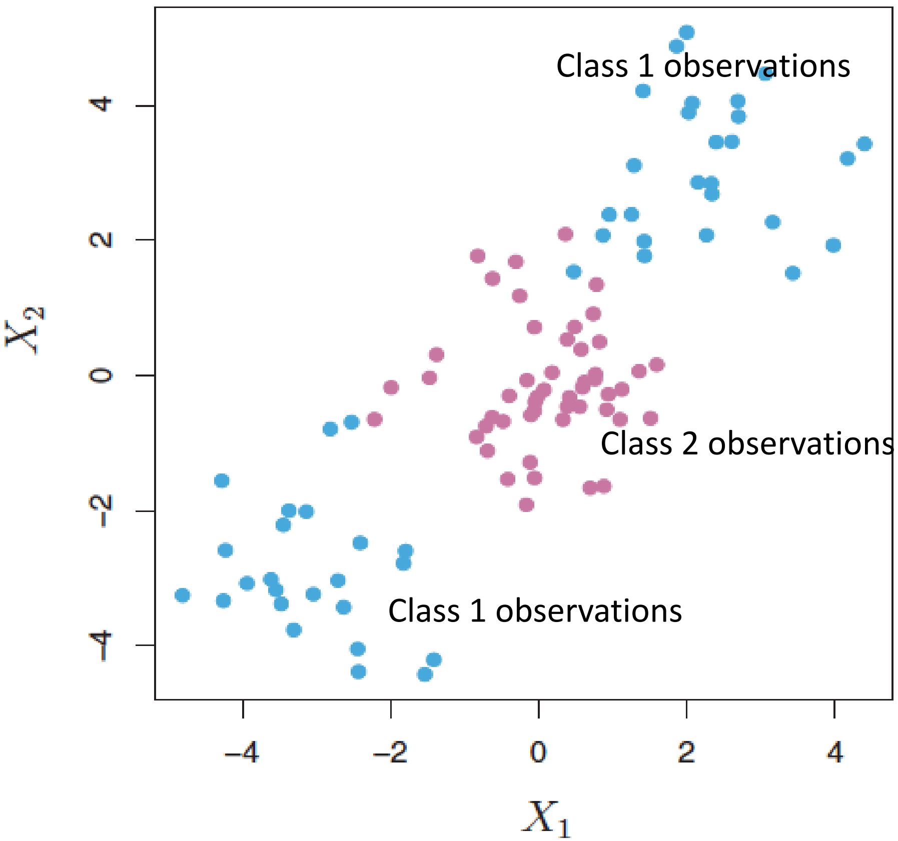
- Then classify:
    - Positive class: Y=+1, if $f(X) > 0$
    - Negative class: Y=-1, if $f(X) < 0$

Linear Decision Boundaries Will Not Always Work

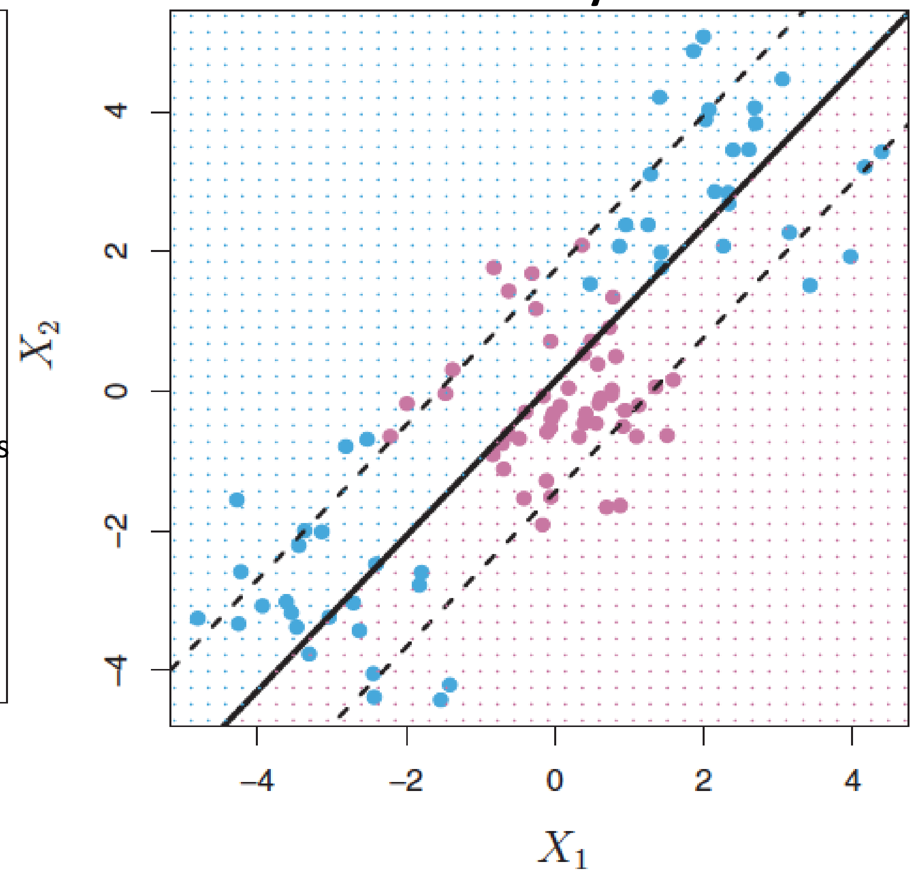
- In some cases, linear boundaries will not work
- The figure shows one example of this case:
    - Class 1 (red triangle) and class 2 (blue circle), a simple linear boundary will not separate them properly



Example:



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



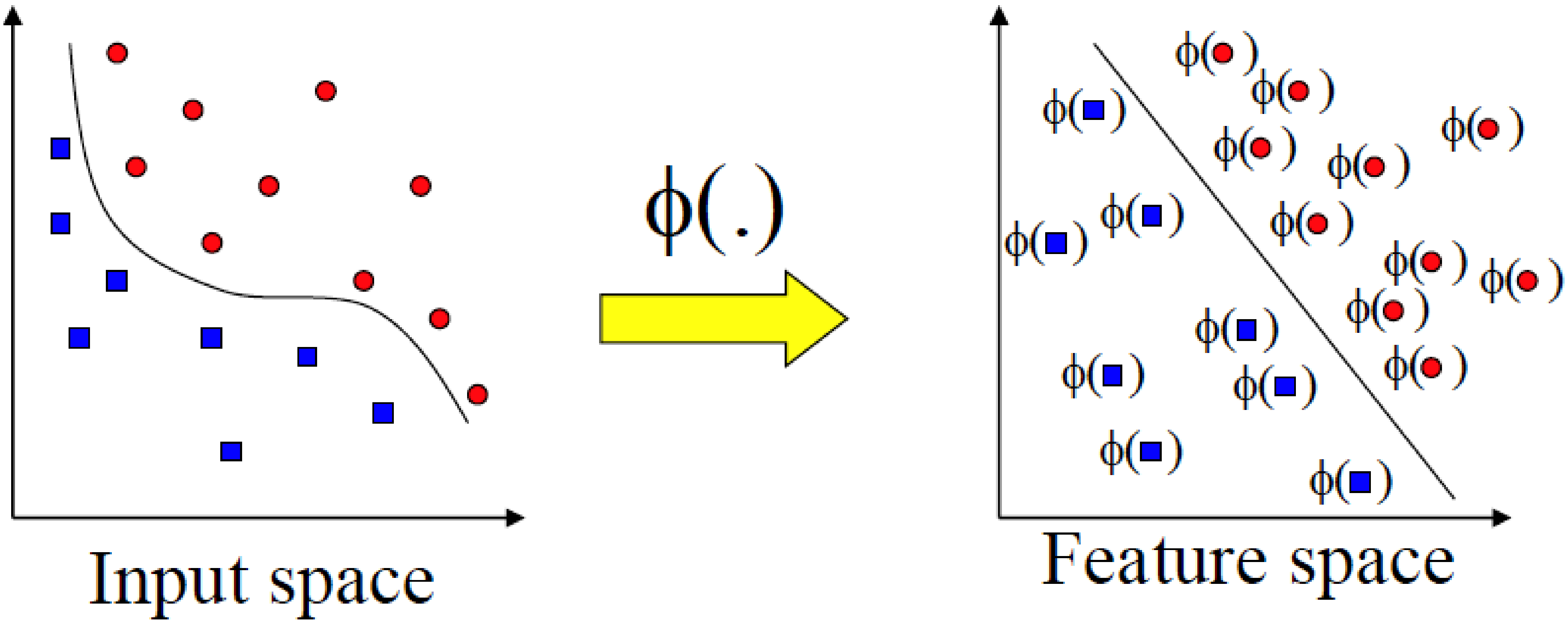
One Possible Solution: Feature Transformation & Expansion

- 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)

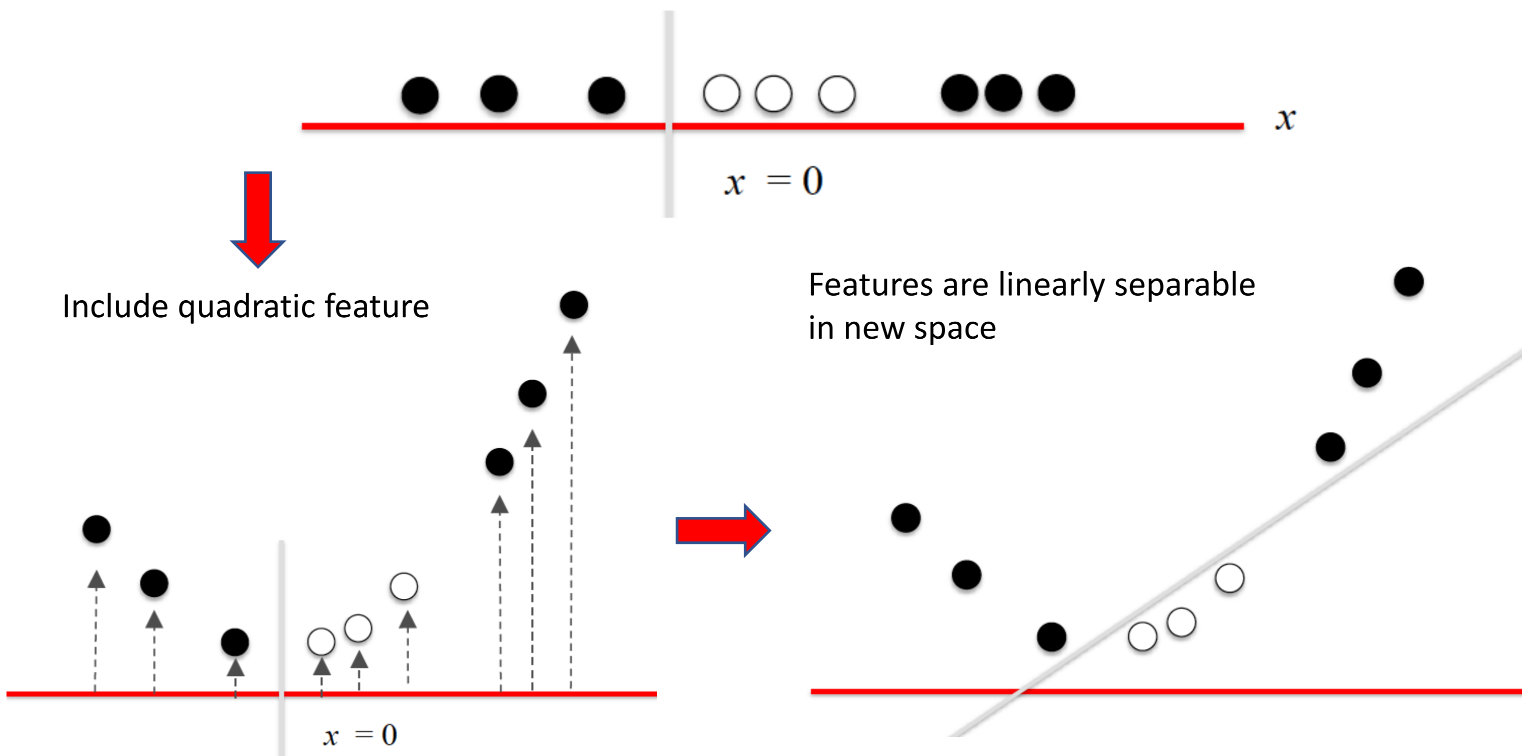
- With quadratic and interaction terms, the decision boundary has the form:

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

Feature transformation



Example 1 D



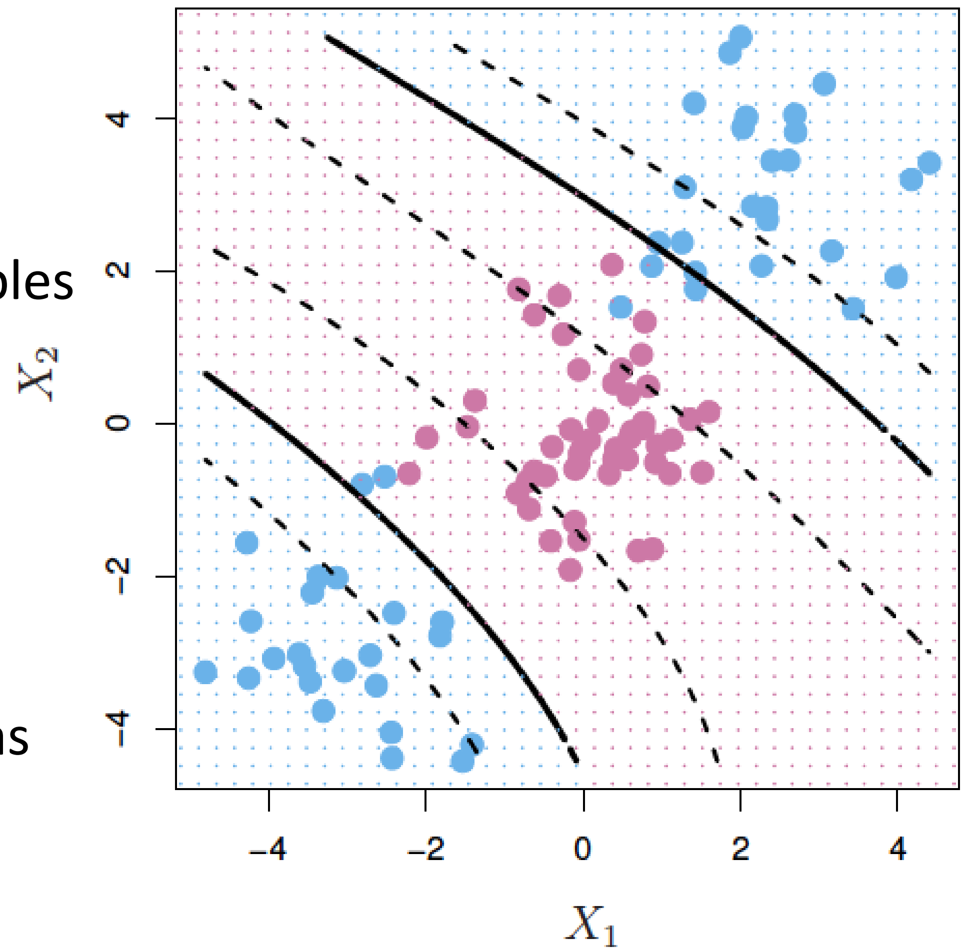
Modified from Kevyn Collins-Thompson

Example

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

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

- We can get non-linear boundaries as shown in figure



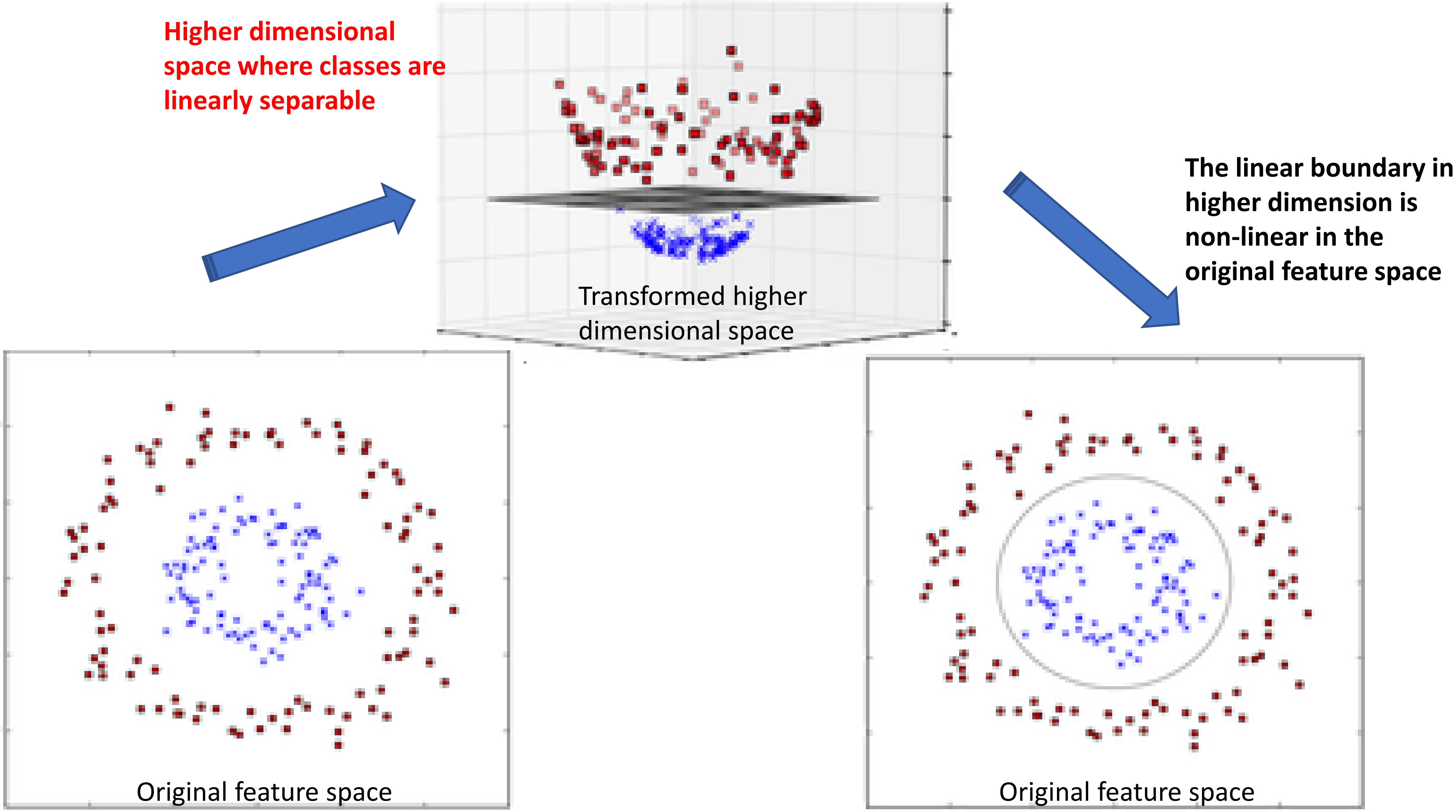
More Efficient Solution: Kernels

- Feature transformation/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 **Kernels**
  - Explicit feature transformation not needed

- **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 enlarges the feature space using Kernels
  - The new feature space is allowed to get very large without explicitly defining new features



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

Modification of objective function - Kernel Trick!

- Change all inner product to Kernel function
- Original :

$$\text{max.} \;\; W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

- Using Kernel

$$\text{max.} \;\; W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Support Vector Machine

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

  Decision boundary equation: $$f(x) = b + \sum_{j=1}^{S} \alpha_{tj} y_{tj} < x_{tj}, x >$$

- 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

$$f(x) = b + \sum_{j=1}^{S} \alpha_{tj} y_{tj} K(x_{tj}, x)$$

- **Support vector machine (<u>SVM</u>) is a support vector classifier (<u>SVC</u>) combined with <u>non-linear Kernel function</u>**
  - Regarded as weighted sum of the **similarity** between sample x and the support vectors

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

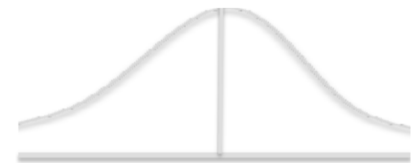
$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(1 + \mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\ &= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z}). \end{aligned}$$

Kernels: RBF

Demo: https://cs.stanford.edu/people/karpathy/svmjs/demo/

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

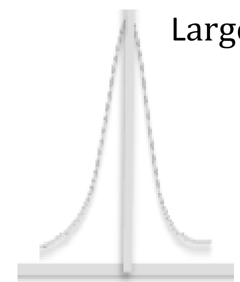
Small γ



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

γ is a constant

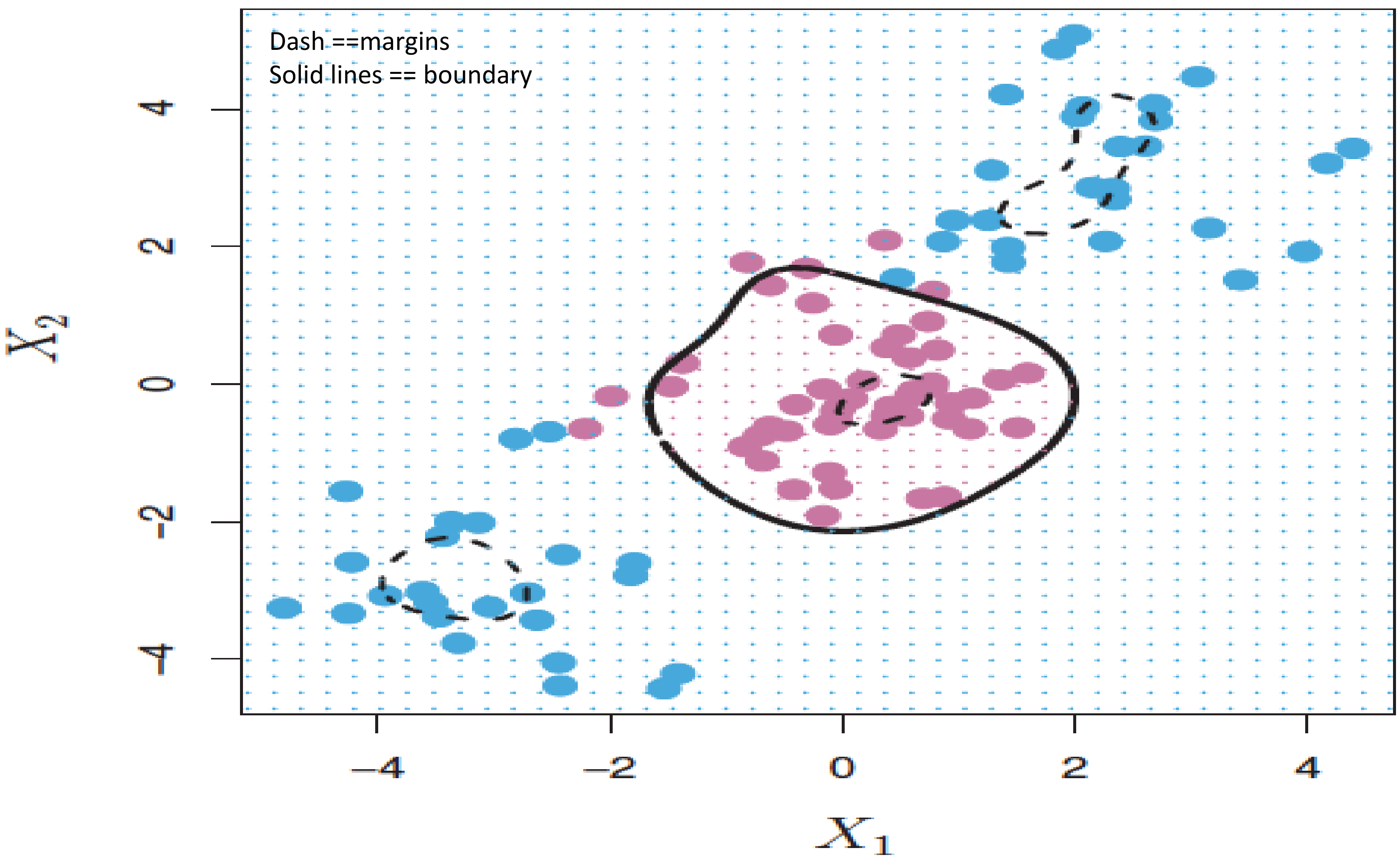
Large γ



  - 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

Non-linear Boundaries by Radial Basis Kernel (SVM)

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



Multiclass Classification (K Classes)

- **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', .. (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
            - Finding the right Kernel could be tricky – cross-validation