Data Mining & Machine Learning

Yong Zheng

Illinois Institute of Technology
Chicago, IL, 60616, USA

ILLINOIS TECH

College of Computing

Review

- KNN Classifier
  - Lazy classifier
  - Have to specify the value of K (cannot be too small or large)
  - Cannot handle categorical data, have to transform data
- Naïve Bayes Classifier
- Tree-Based Learning

Review

- KNN Classifier
- Naïve Bayes Classifier
  - Assumption: conditionally independent
  - Cannot handle numeric, have to transform data
  - May have serious imbalance issues in labels (general issue in classification)
- Tree-Based Learning

Review

- KNN Classifier
- Naïve Bayes Classifier
- Tree-Based Learning
  - More complicated but much more effective sometimes
  - Tree-based learning: a machine learning method
  - Require feature selection
  - Require to handle overfitting problems (Stop-Earlier or Post-Pruning)

More Classification Algorithms

More classification algorithms and topics:

- Logistic Regression
- SVM for Classifications & SVR for Regressions
- Multi-Class Classification by Binary Classification
- Neural Networks
- Ensemble Classification
  - Bagging
  - Boosting
  - Random Forest
- Multi-Label Classifications

More Classification Algorithms

More classification algorithms and topics:

- Logistic Regression
- SVM for Classifications & SVR for Regressions
- Multi-Class Classification by Binary Classification
- Neural Networks
- Ensemble Classification
  – Bagging
  – Boosting
  – Random Forest
- Multi-Label Classifications

Logistic Regression

- Both Logistic regression and Linear SVM model can be considered as *linear* classification models. They tried to utilize linear models to solve the problem of classifications
- We discuss logistic regression and SVM by using a binary classification as an example
- Note that both of them can be applied to multi-class classifications too (discuss later)

Simple Logistic regression model

Relationship between qualitative binary variable Y and one x-variable:

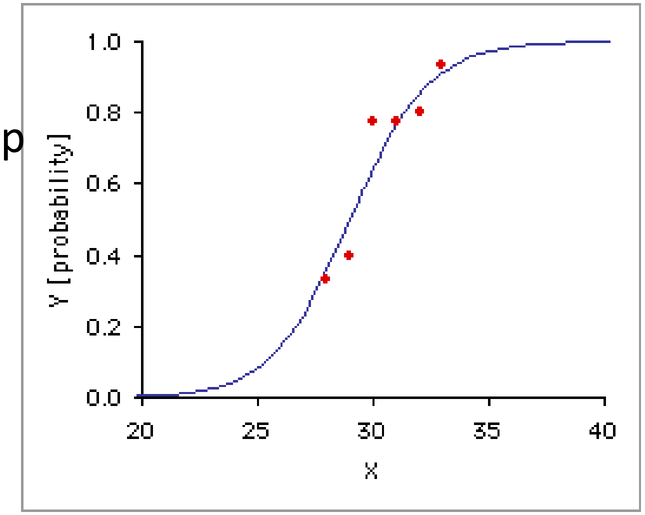
Model for probability p=Pr(Y=1) for each value x.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Odds= $\dfrac{p}{1-p} = \dfrac{P(Y=1)}{P(Y=0)}$

measures the odds that event Y = 1 occurs

p



In logistic regression, we use 1 and 0 to denote binary labels

Interpreting odds $\frac{p}{1-p} = \frac{P(Y=1)}{P(Y=0)}$

Let p=Pr(Y=1) the probability of “success”

- If odd>1 then pr(Y=1) **>** Pr(Y=0) ➔ Pr(Y=1) > 0.5
- If odd=1 then Pr(Y=1) **=** Pr(Y=0) ➔ Pr(Y=1)=0.5
- If odd<1 then p=pr(Y=1) **<** Pr(Y=0) ➔ Pr(Y=1) < 0.5

$$\log\left(\frac{p}{1-p}\right) = \underline{\beta_0 + \beta_1 x}$$ ➔ Regressions

General Logistic Regression

- We may have several x variables in the model

$$\log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 {x_1}^2 + \beta_6 x_1 x_2$$

- 0.5 is the default cut-off value, but we may improve the model by using other cut-off values
  - P(Y=1) >= alpha → predicted as 1
  - P(Y=1) < alpha → predicted as 0
  - Try different alpha values to see which one is the best
- The model is interpretable. P(Y=1) can be considered as a confidence value

General Logistic Regression

- Model fitting or building
  - The process is similar to the linear regression models
  - X must be numerical variable. Transformation is required if there are nominal variables
  - Feature selection methods, such as backward elimination, forward or stepwise selection, can also be applied
  - Residual analysis needs to be performed
  - The model is evaluated by classification metrics, such as accuracy, precision, recall, ROC curve, etc

Example: Logistic Regression

- Data: Case Study 3 - Admissions

```
admit,gre,gpa,rank
0,380,3.61,3
1,660,3.67,3
1,800,4,1
1,640,3.19,4
0,520,2.93,4
1,760,3,2
1,560,2.98,1
0,400,3.08,2
```

Example: Logistic Regression

- Example: hold-out evaluations, load and split data

```
> mydata=read.csv("case3_admission.csv",header=T)
> mydata=mydata[sample(nrow(mydata)),]
> select.data = sample (1:nrow(mydata), 0.8*nrow(mydata))
> train.data = mydata[select.data,]
> test.data = mydata[-select.data,]
> head(mydata)
    admit gre  gpa rank
265     1 520 3.90    3
140     1 600 3.58    1
120     0 340 2.92    3
172     0 540 2.81    3
247     0 680 3.34    2
168     0 720 3.77    3
> train.label=train.data$admit
> test.label=test.data$admit
```

Example: Logistic Regression

- Example: hold-out evaluations, build model by FS

```
> full=glm(admit~gre+gpa+rank, data=train.data, family=binomial())
> base=glm(admit~gpa, data=train.data, family=binomial())
> library(leaps)
Warning message:
package 'leaps' was built under R version 3.5.2
> step(base, scope=list(upper=full, lower=~1), direction="both", trace=F)

Call:  glm(formula = admit ~ gpa + rank + gre, family = binomial(),
    data = train.data)

Coefficients:
(Intercept)          gpa         rank          gre
  -2.861669     0.683853    -0.594686     0.002019

Degrees of Freedom: 319 Total (i.e. Null);  316 Residual
Null Deviance:      402.1
Residual Deviance: 370.7        AIC: 378.7
```

Example: Logistic Regression

- Example: hold-out evaluations, produce probabilities

```
> predict(full, type="response", newdata=test.data)
       168        153        147        392        349        184        339        399 
0.35114554 0.48386137 0.31932437 0.48369815 0.27974173 0.41706093 0.45565929 0.46461888 
       182        297        159          7        283          1        274        243 
0.17113226 0.45889138 0.41777034 0.42851806 0.17527994 0.19625871 0.53655916 0.22292808 
        95        266         12        108        217        271        187        224 
0.40947750 0.16875397 0.40949799 0.28059366 0.31286434 0.48556940 0.25990833 0.34126702 
        86        358         19        112        385          3        293        330 
0.27620968 0.56483108 0.53208385 0.11299118 0.21580479 0.70974819 0.46308072 0.09732698 
       379        176        113        109        255         29         25        304 
0.22793249 0.37877506 0.13384615 0.13770473 0.20762858 0.43184987 0.44379247 0.51088138 
        54        242         18        150        226        245        123        131 
0.39126377 0.54957227 0.10263655 0.57472852 0.31031481 0.42867812 0.16152450 0.34716567 
> prob=predict(full, type="response", newdata=test.data)
```

Example: Logistic Regression

- Example: hold-out evaluations
- Next, choose cut-off value to calculate accuracy

```
> for(i in 1:length(prob)){
+     if(prob[i]>0.5){
+         prob[i]=1
+      }else{
+ prob[i]=0
+ }
+ }
> library(Metrics)
> accuracy(test.label, prob)
[1] 0.6875
```

```
> prob=predict(full, type="response", newdata=test.data)
> for(i in 1:length(prob)){
+     if(prob[i]>0.4){
+         prob[i]=1
+      }else{
+ prob[i]=0
+ }
+ }
> accuracy(test.label, prob)
[1] 0.7
```

More Classification Algorithms

More classification algorithms and topics:

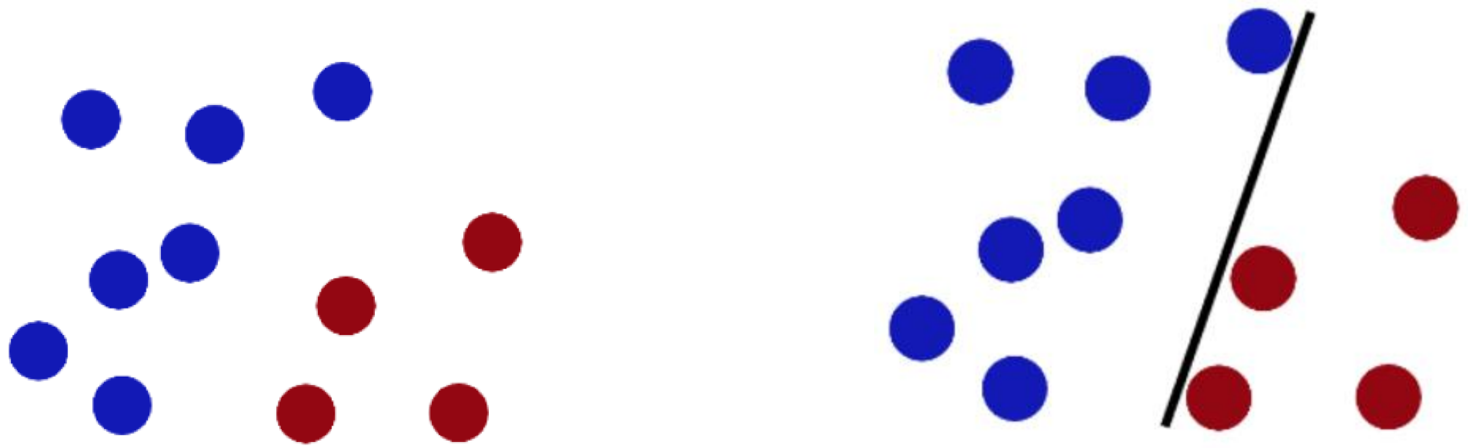
- Logistic Regression
- SVM for Classifications & SVR for Regressions
- Multi-Class Classification by Binary Classification
- Neural Networks
- Ensemble Classification
  - Bagging
  - Boosting
  - Random Forest
- Multi-Label Classifications

Classification Algorithms: Support Vector Machines (SVM)

Linear SVM

Linear SVM

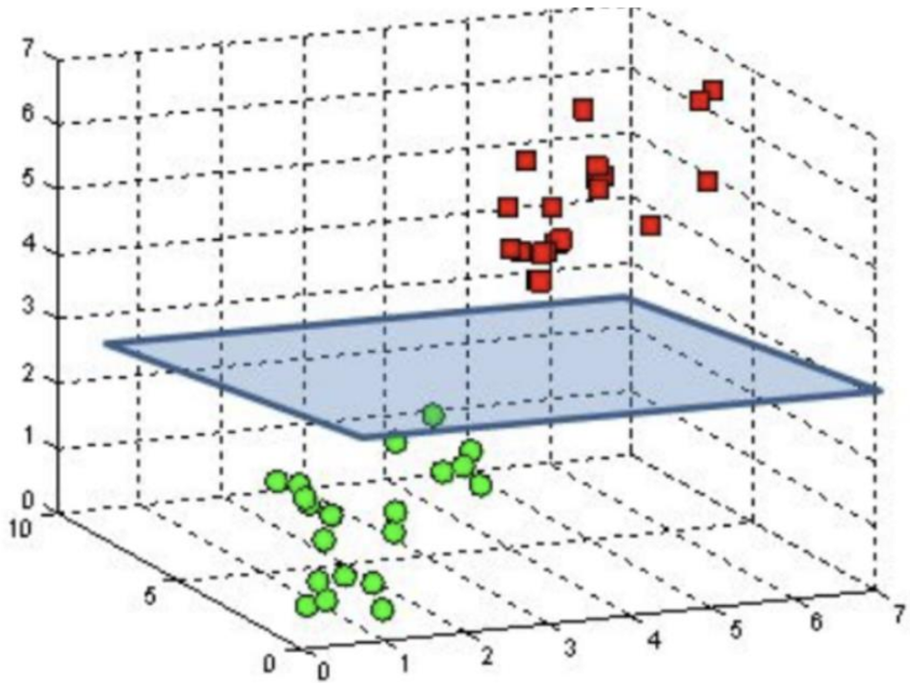
- Draw a linear model to separate two classes



- The model could be a straight line model (such as regression line in 2D space)

Linear SVM

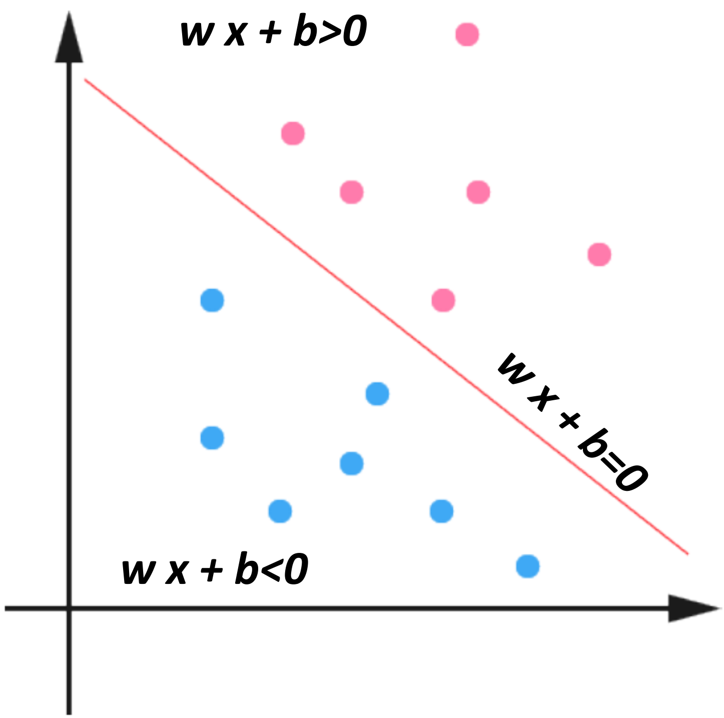
- The model could be a hyperplane model in multi-dimensional space



We use straight line model as an example in the class. But, you should also keep in mind that the hyperplane model is still linear SVM

Linear SVM

$\boldsymbol{f}(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$



denotes +1

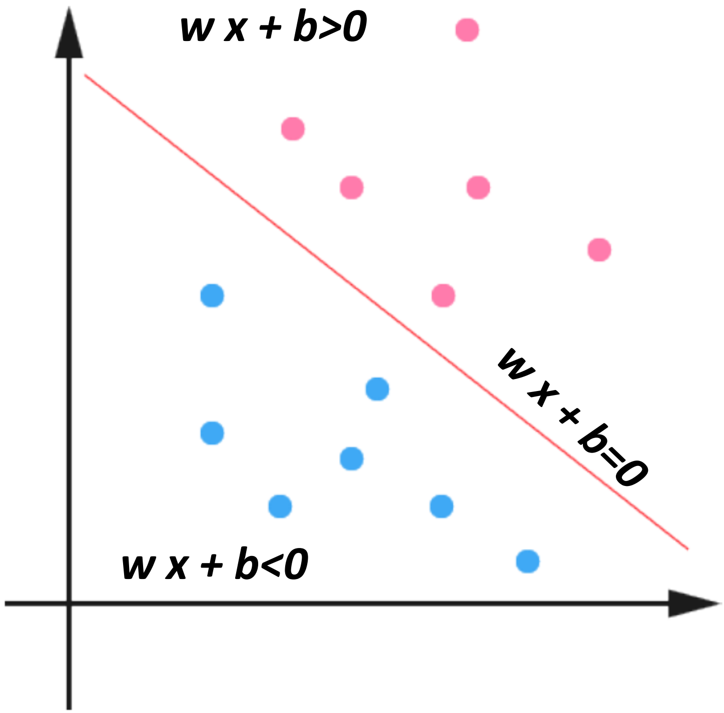
denotes -1

In logistic regression, we use 0 and 1 for binary labels.

In SVM, we use +1 and -1 as binary labels

Linear SVM

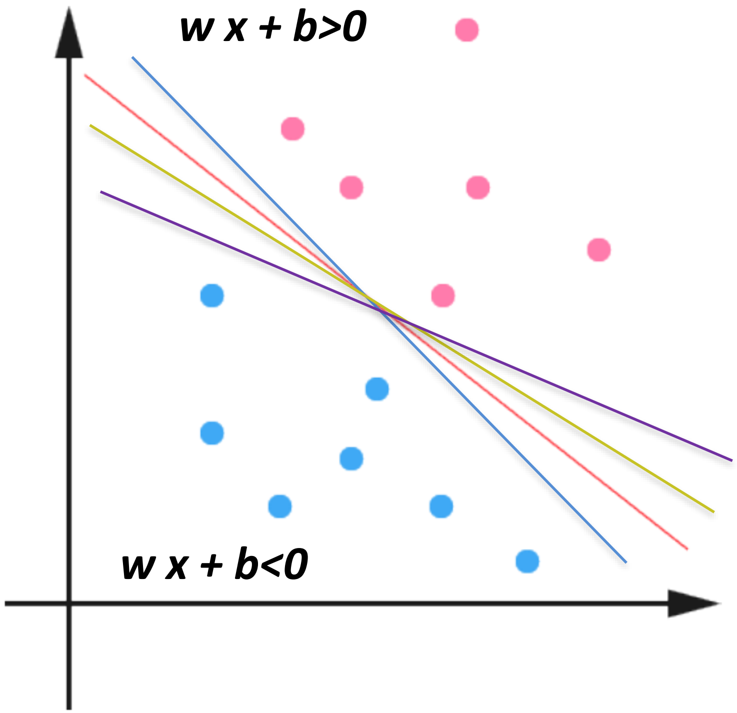
$\boldsymbol{f}(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$



How would you classify this data?

Linear SVM

f(***x***,***w***,*b*) = *sign*(***w x*** + *b*)

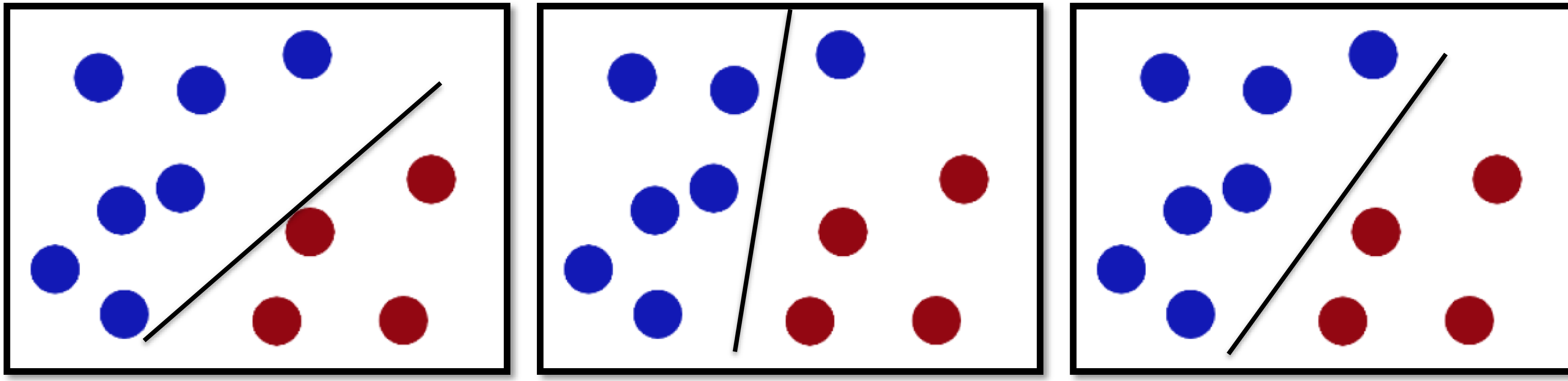


Any of these would be fine..

..but which is best?

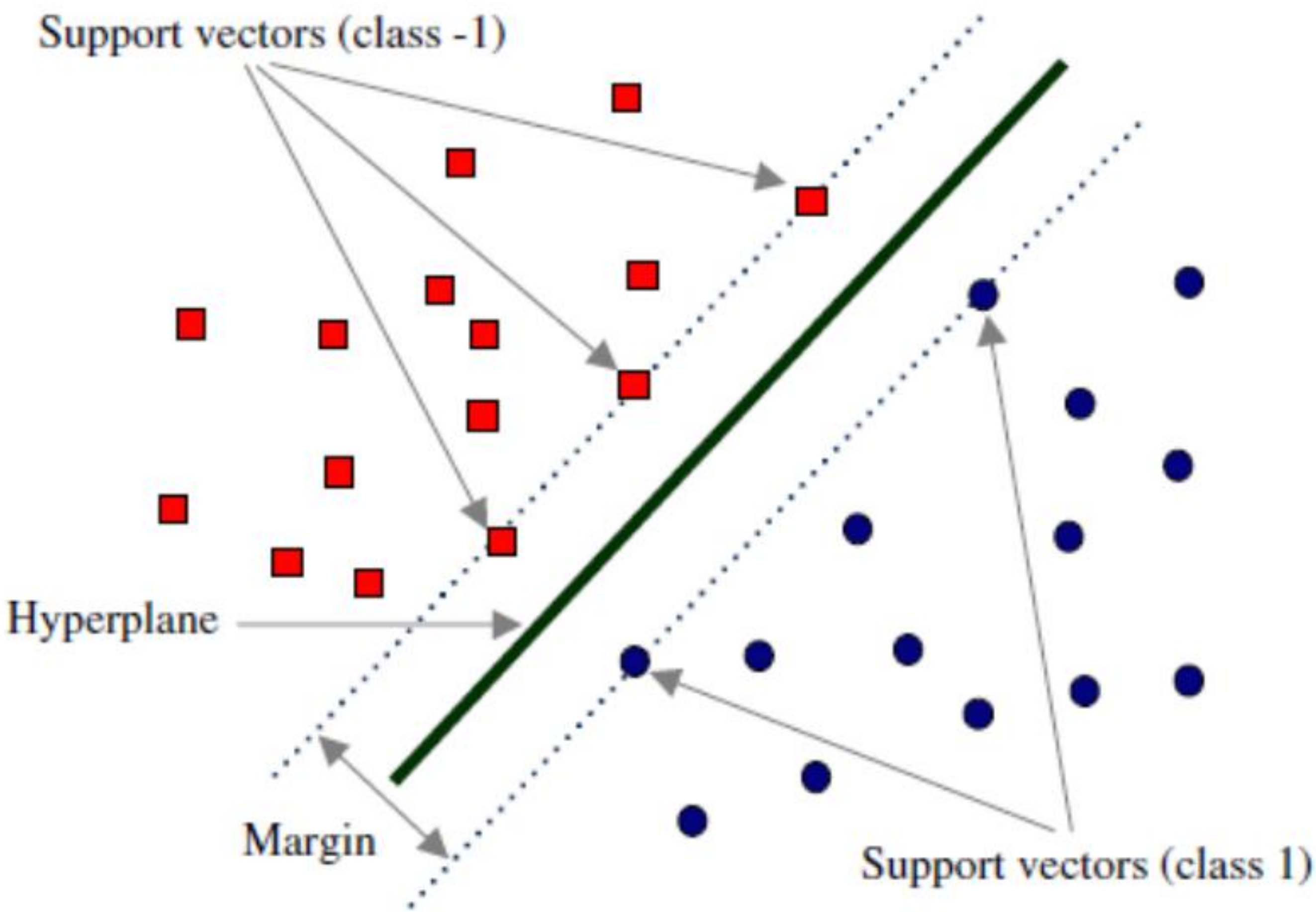
Linear SVM

- Which one do you think is the best?



A B C

Linear SVM

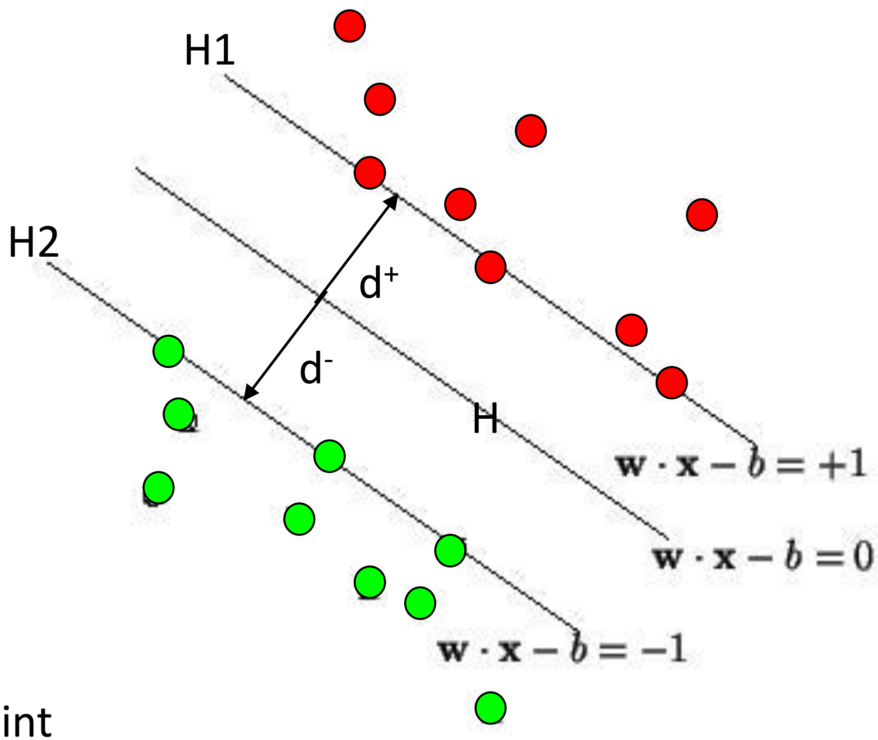

Support vectors (class -1)
Hyperplane
Margin
Support vectors (class 1)

Definition: Margin

Define the hyperplane H such that:

$\mathbf{x}_i \bullet \mathbf{w} + b \geq +1$ when $y_i = +1$

$\mathbf{x}_i \bullet \mathbf{w} + b \leq -1$ when $y_i = -1$



H1 and H2 are the planes:

H1: $\mathbf{x}_i \bullet \mathbf{w} + b = +1$

H2: $\mathbf{x}_i \bullet \mathbf{w} + b = -1$

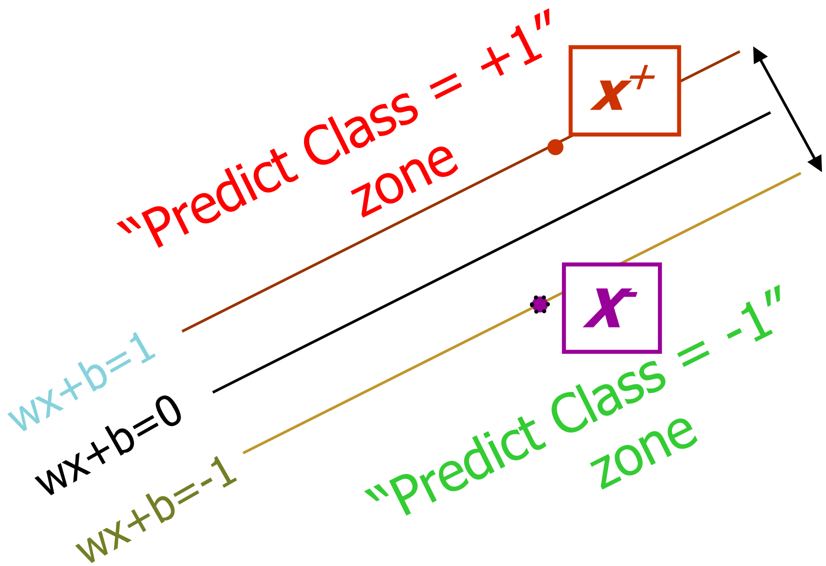
The points on the planes H1 and H2 are the points in two classes (+1, -1) on the boundary. They are also called the *Support Vectors*

d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin of a separating hyperplane is $d^+ + d^-$.

Definition: Margin



M=Margin Width

What we know:

- $\boldsymbol{w} \,.\, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^- + b = -1$
- $\boldsymbol{w} \,.\, (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

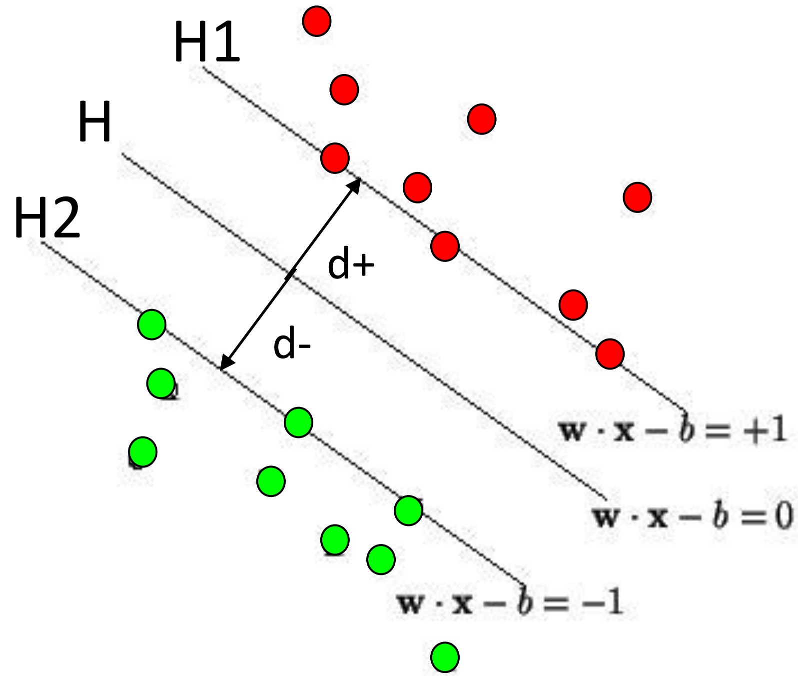
$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Objective: Maximal Margin in SVM Classification

Method: Maximizing the Margin

We want a classifier with as big margin as possible.

$$M = \frac{(x^{+} - x^{-}) \cdot w}{|w|} = \frac{2}{|w|}$$



Maximize M ➔ Minimize |w| ➔

Minimize $\frac{1}{2} w^t w$ = objective function

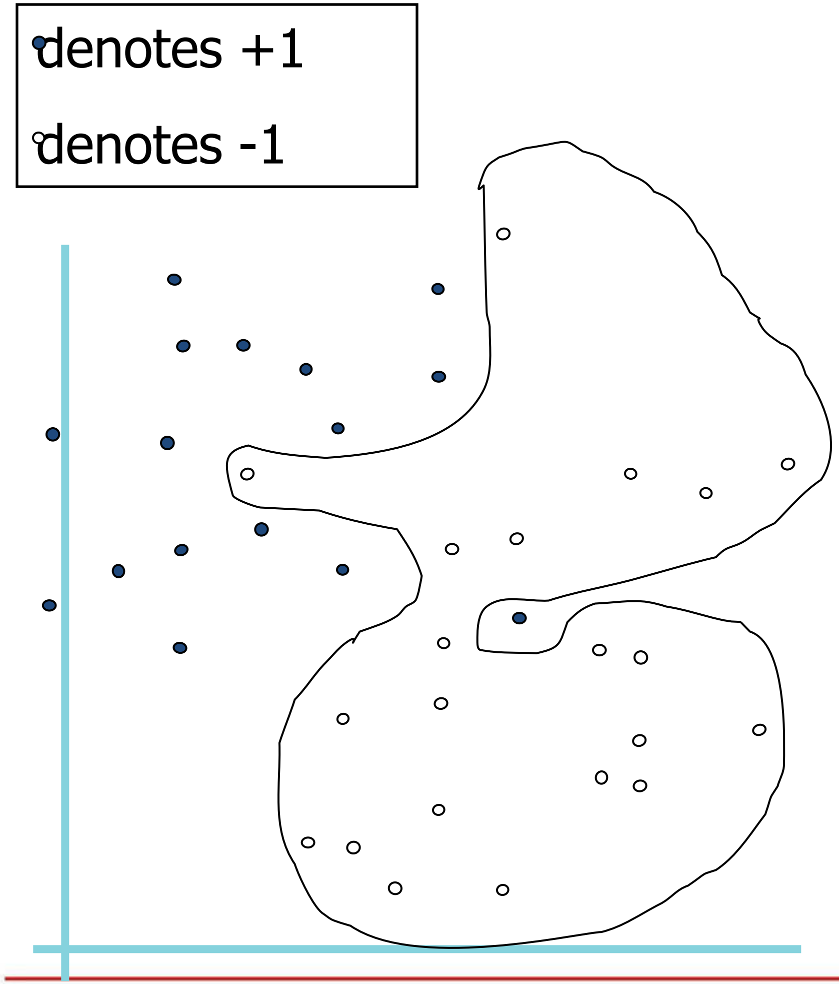
Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i (\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **Need to optimize a *quadratic* function subject to *linear* constraints.**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

Dataset with noise

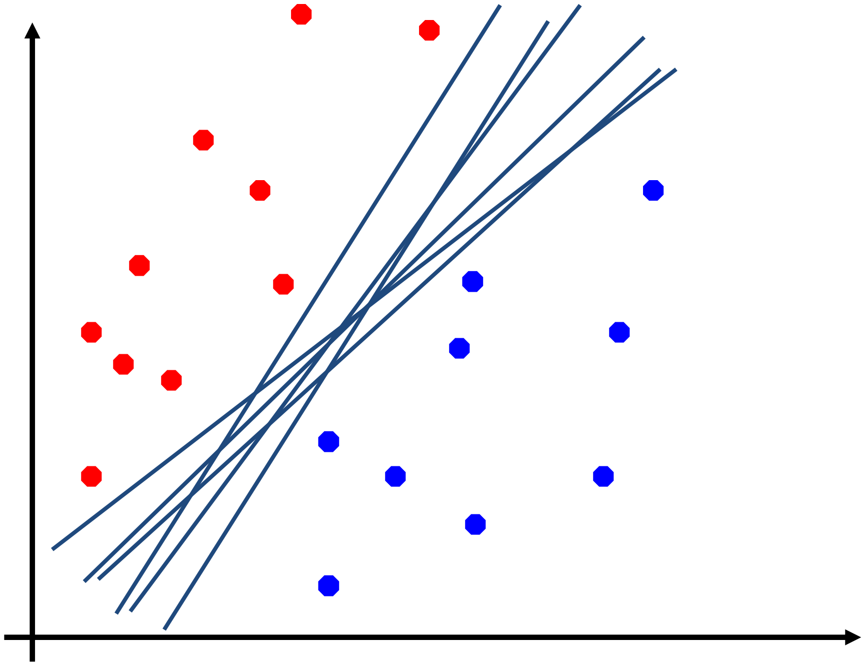


- **Hard Margin:** **So far we require all data points be classified correctly**

  - No training errors are allowed
- **Soft Margin:** **we allow errors but we want to minimize the errors**
- **Hard margin will build models without errors, which may introduce overfitting**
- **Soft margin allows errors in the model, which may help build a more general model**

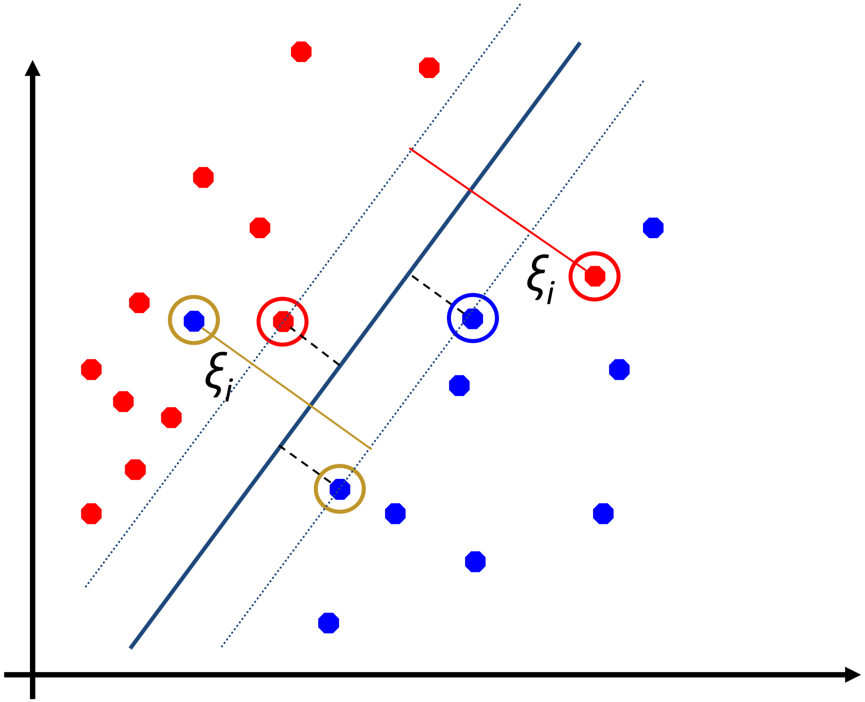
Hard Margin Classification

- No misclassifications



Soft Margin Classification

Slack variables ξi **can be added to allow misclassification of difficult or noisy examples.**



New objective function

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w}^T\mathbf{x_i} + \mathrm{b}) \geq 1$

- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

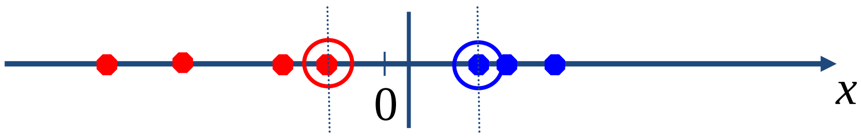
- **Parameter C can be viewed as a way to control overfitting.**

Classification Algorithms: Support Vector Machines (SVM)

Non-Linear SVM

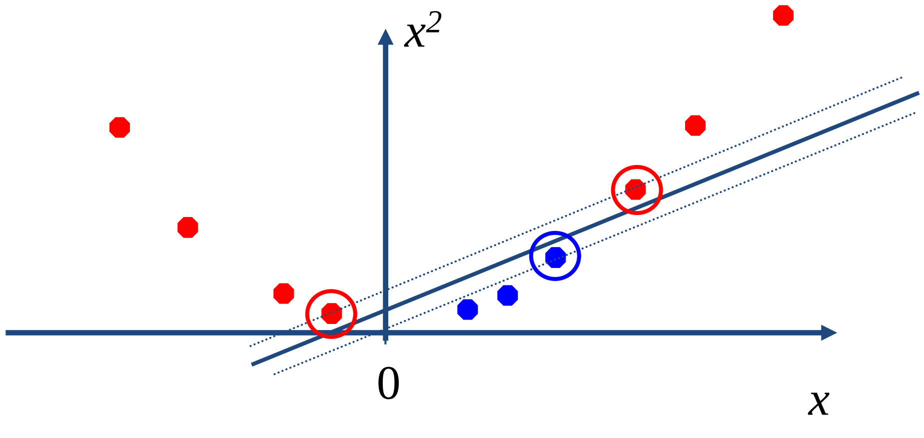
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:



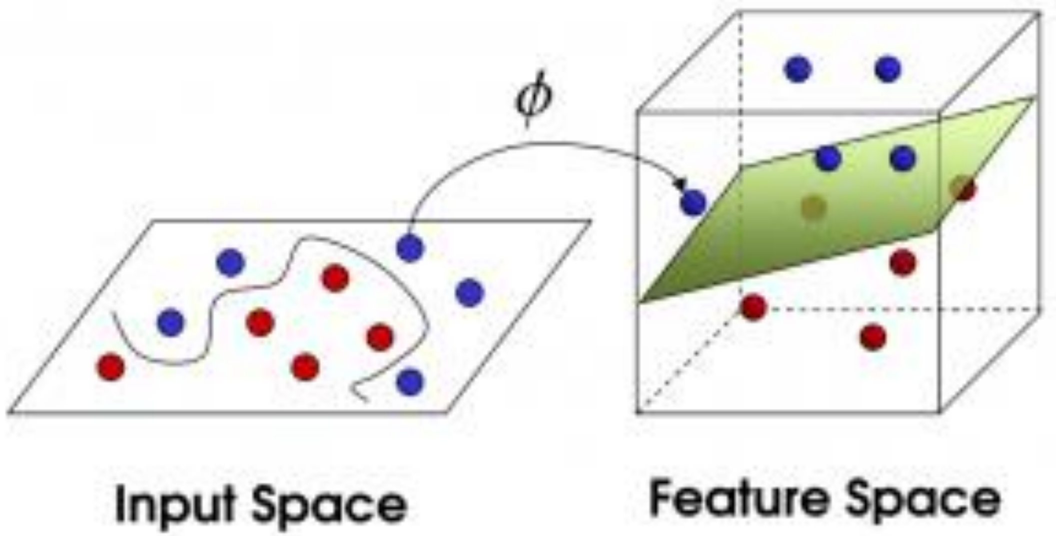
- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:



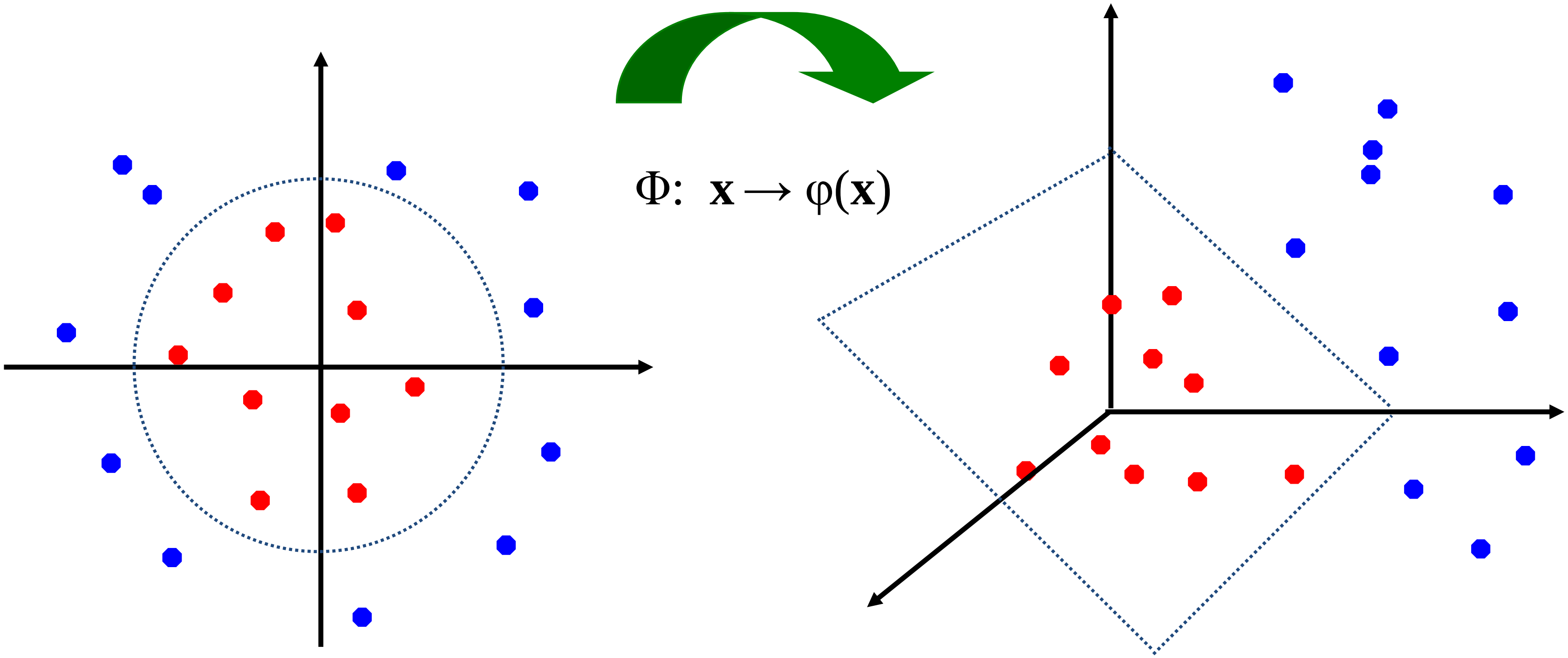
Solution: Non-linear SVMs

- We can map the original data to higher-dimensional space



Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



The Kernel Function

- In the 2D space, our linear SVM model is f(x) = wx + b
- In the MD space, our model becomes f(x) = w φ(x) + b
- φ(x) are the new vectors mapped to a higher-dimensional space
- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- It helps us convert the input space from lower-dimension to higher-dimension by using the inner product $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$

The Kernel Function

- It helps us convert the input space from lower-dimension to higher-dimension by using the inner product $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$
- One example: 2-dimension space, $x = [x_1 \; x_2]$
  let Polynomial Kernel $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$
  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$$

$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

$$= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j),$$

where $\boldsymbol{\varphi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Examples of Popular Kernel Functions

- Linear Kernel: $K(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$
- Polynomial Kernel of power p: $K(\mathbf{x_i},\mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^p$
- Gaussian (radial-basis function network) Kernel:

$$K(\mathbf{x_i},\mathbf{x_j}) = \exp\left(-\frac{\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2}{2\sigma^2}\right)$$

- Sigmoid Kernel: $K(\mathbf{x_i},\mathbf{x_j}) = \tanh(\beta_0 \mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$\boxed{f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x}_i, \mathrm{x}_j) + b}$$

Still linear formula

- **Optimization techniques for finding α_i's remain the same!**

Nonlinear SVM - Overview

- SVM finds a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Weakness of SVM

- **It is sensitive to noise**

  - A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

  - how to do multi-class classification (MCC) with SVM?

  - There are many methods to convert MCC to binary classification Below is one of these methods:

  1) with output arity m, learn m SVM's

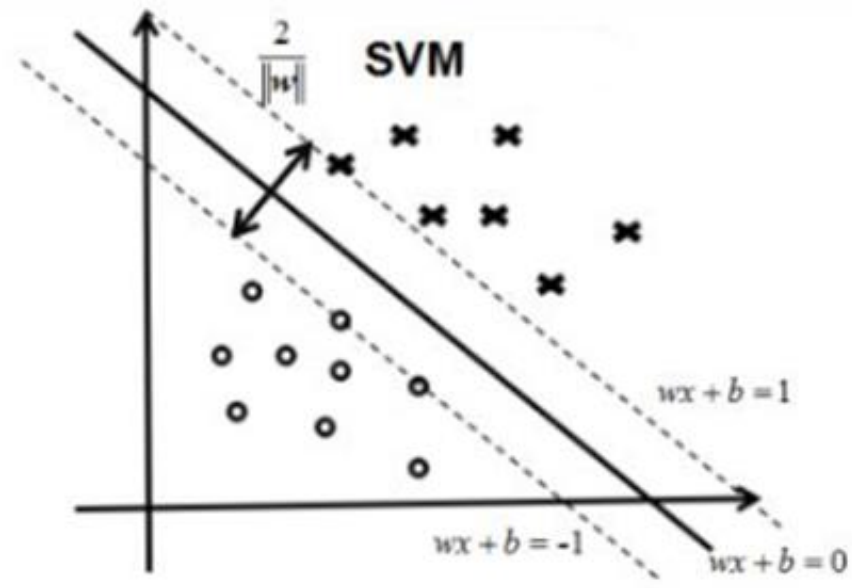
  - **SVM 1 learns "Output==1" vs "Output != 1"**
  - **SVM 2 learns "Output==2" vs "Output != 2"**
  - **:**
  - **SVM m learns "Output==m" vs "Output != m"**

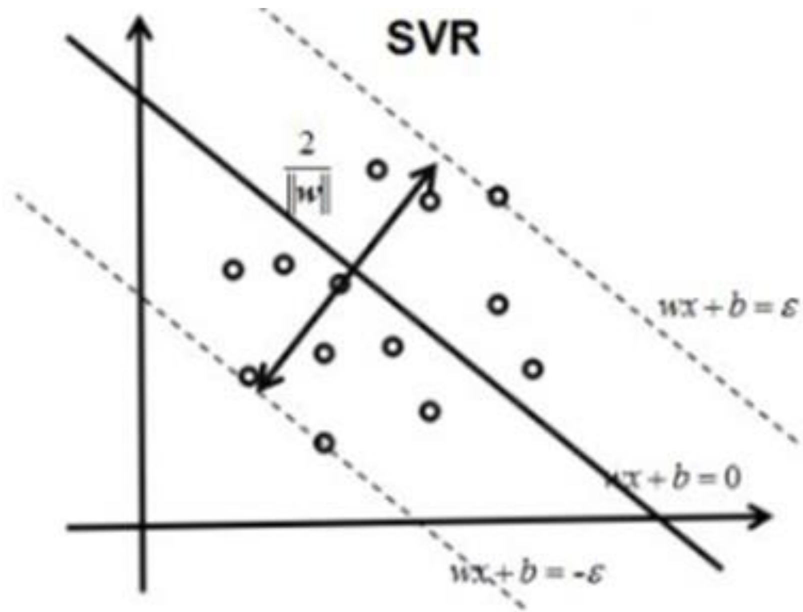
  2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

SVM for Regressions

Support Vector Regression (SVR)

Support Vector Regression (SVR)





$$\begin{cases} \min \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t.\ y_i(wx_i+b) \geq 1, \quad \forall i \end{cases}$$

$$\begin{cases} \min \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t.\ |y_i-(wx_i+b)| \leq \varepsilon, \quad \forall i \end{cases}$$

SVR vs SVM

- In SVM, we have two lines on the boundary – they are the lines closest to the hyper-plane (i.e., the SVM classifier line or plane). Our model is the hyper-plane which wants to maximize the margin/distances
- In SVR, we have two lines on the boundary – they are the lines farthest to the hyper-plane (i.e., the regression line). Our model is the hyper plane or regression line which minimizes the distances
- Same characteristics: the hyper-plane are the models in between the boundary lines

More Classification Algorithms

More classification algorithms and topics:

- Logistic Regression
- SVM for Classifications & SVR for Regressions
- Multi-Class Classification by Binary Classification
- Neural Networks
- Ensemble Classification
  - Bagging
  - Boosting
  - Random Forest
- Multi-Label Classifications

Multi-Class Classifications

- All the classification techniques we discussed can be applied to multi-class classifications
- Multi-Class classification can be solved by multiple binary classifications
  - One vs. One
  - One vs. Rest
  - Many vs. Many

Multi-Class Classifications

- Strategy 1: One vs. One
  - Assume we have N labels
  - We will choose unique pair of these labels, and perform N(N - 1)/2 binary classifications
  - We will get N(N - 1)/2 classification results
  - Finally, we use voting to get the final prediction results
  - Notes: one label as positive, another as negative

Multi-Class Classifications

- Strategy 1: One vs. One
- Example: Assume we have 4 labels: c1, c2, c3, c4
- We will get N(N - 1)/2 = 6 unique pairs
  - c1, c2 → Binary Classification
  - c1, c3 → Binary Classification
  - c1, c4 → Binary Classification
  - c2, c3 → Binary Classification
  - c2, c4 → Binary Classification
  - c3, c4 → Binary Classification

Predictions

Multi-Class Classifications

- Strategy 2: One vs. Rest
  - Assume we have N labels
  - We will perform N binary classifications
  - In each classification, we predict C vs. Not-C
  - Finally, we use voting to get the final prediction results
  - Notes: one label as positive, others as negative

Multi-Class Classifications

- Strategy 2: One vs. Rest
- Example: Assume we have 4 labels: c1, c2, c3, c4
- We will perform N = 4 binary classifications
  - c1, ¬c1 → Binary Classification
  - c2, ¬c2 → Binary Classification
  - c3, ¬c3 → Binary Classification
  - c4, ¬c4 → Binary Classification

→ Predictions

Multi-Class Classifications

- Strategy 3: Many vs. Many
  - Assume we have N labels
  - We will perform N binary classifications
  - They encode labels into new ones
  - Example: Error Correcting Output Codes, ECOC http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/lecture_notes/ecoc/ecoc.pdf
  - Notes: one set as positive, another set as negative