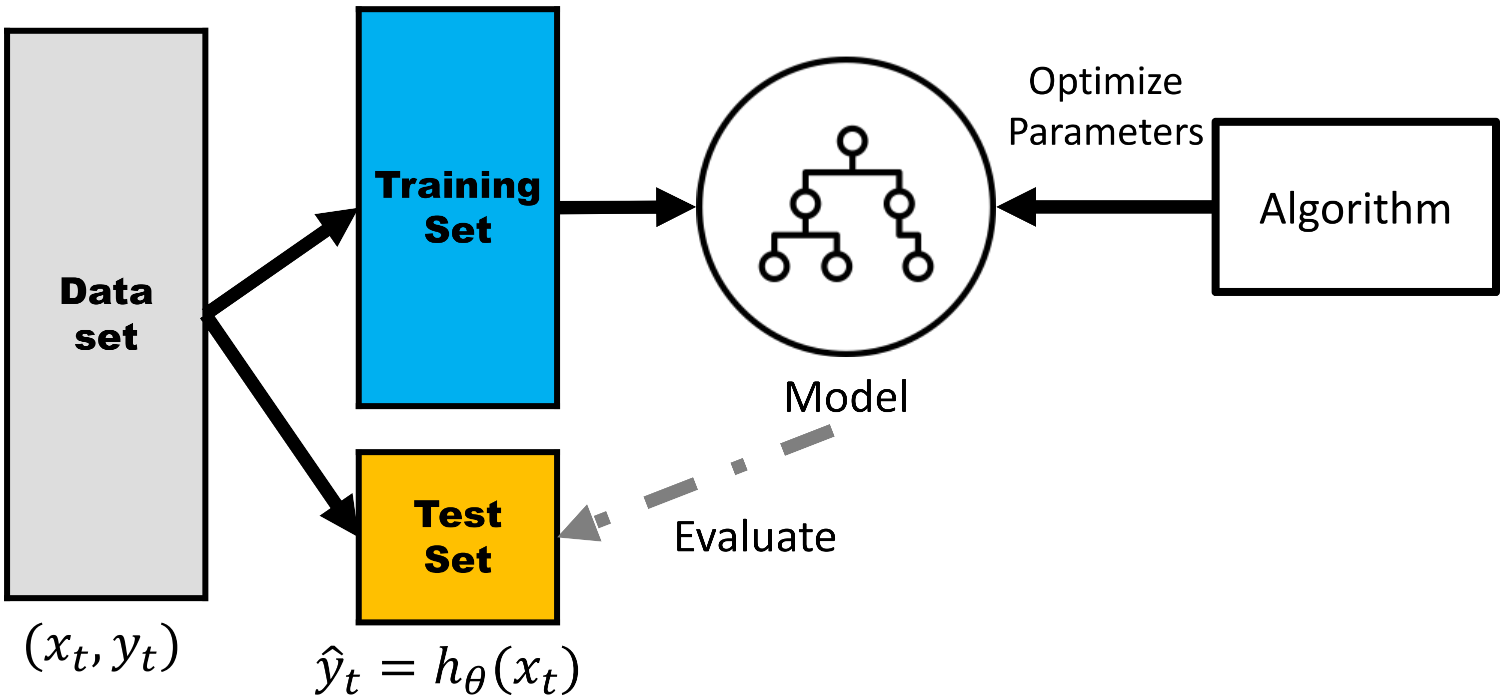
COMP4434 Big Data Analytics

Lecture 4 Overfitting & Support Vector Machines

HUANG Xiao

xiaohuang@comp.polyu.edu.hk

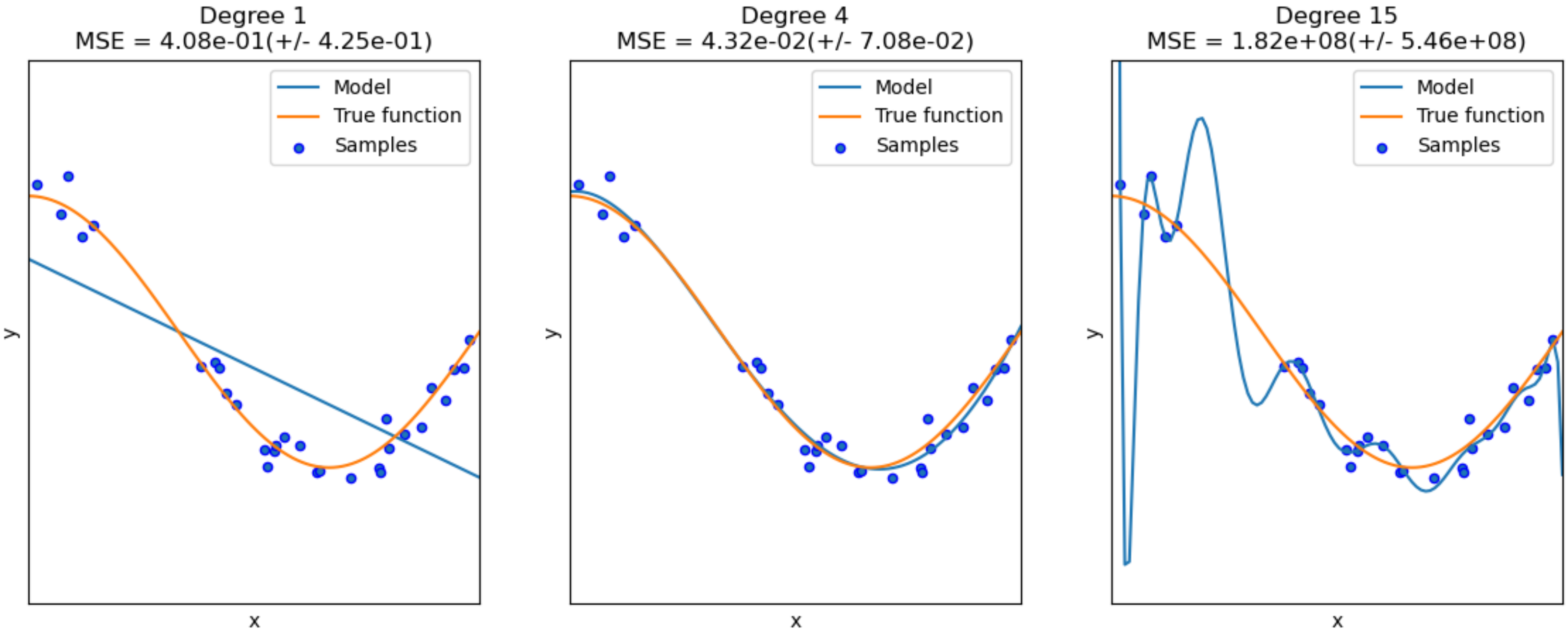
Model Evaluation



- When training the model, we can not use test set
- If we have several models, e.g., linear regression and quadratic regression, how could we evaluate them?

Underfitting and Overfitting

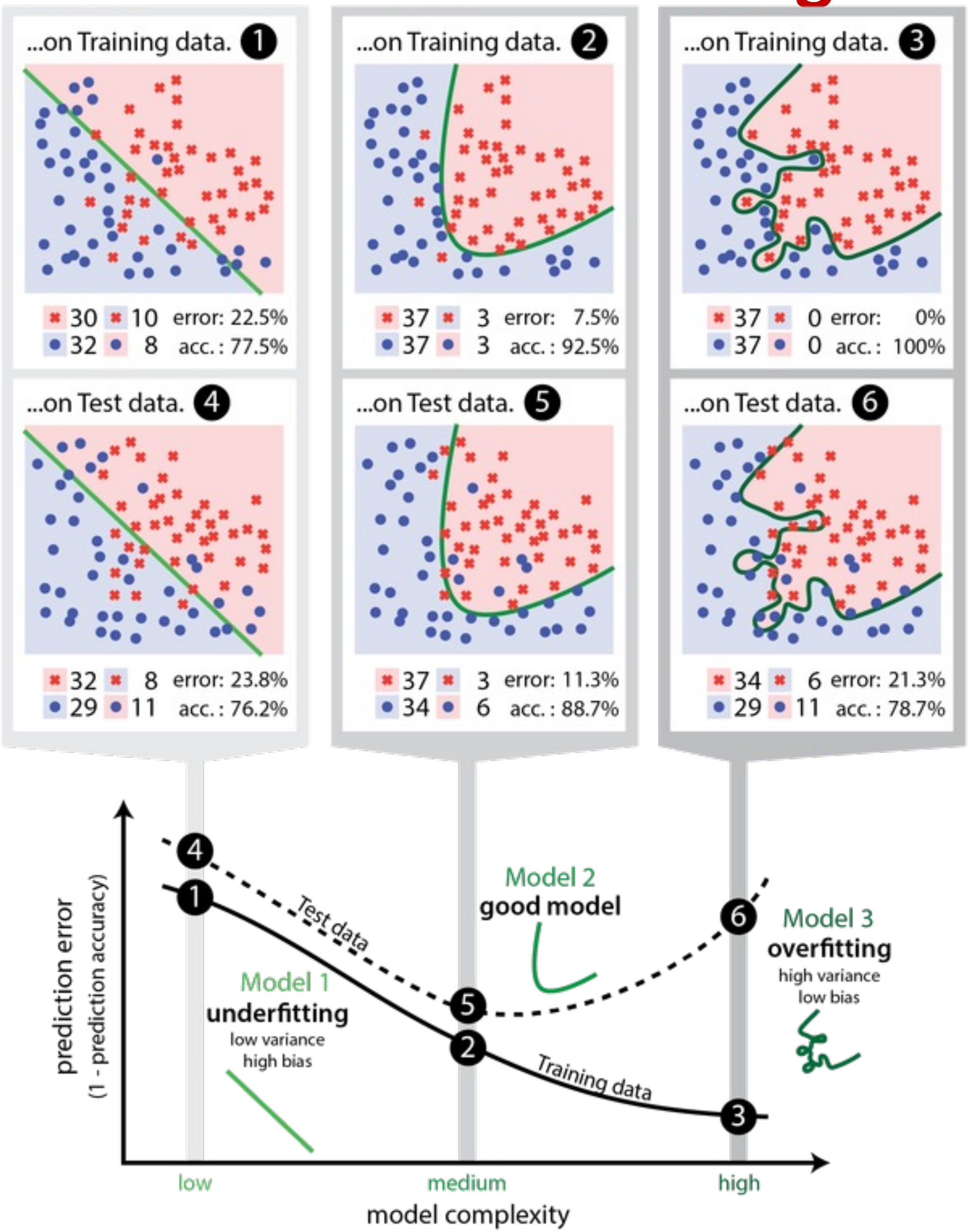
- Polynomial Regression with Degree = 4:
  - $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$




```
[-1.60931179]
[  0.46754142 -17.78954475  23.5926603   -7.26289872]
[-2.98293201e+03  1.03899495e+05 -1.87416410e+06  2.03716626e+07
 -1.44873631e+08  7.09317305e+08 -2.47066542e+09  6.24563114e+09
 -1.15676921e+10  1.56895529e+10 -1.54006639e+10  1.06457712e+10
 -4.91379699e+09  1.35920272e+09 -1.70381606e+08]
```

https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Underfitting and Overfitting



Two classes separated by an elliptical arc

Underfitting

a model does not fit the data well enough

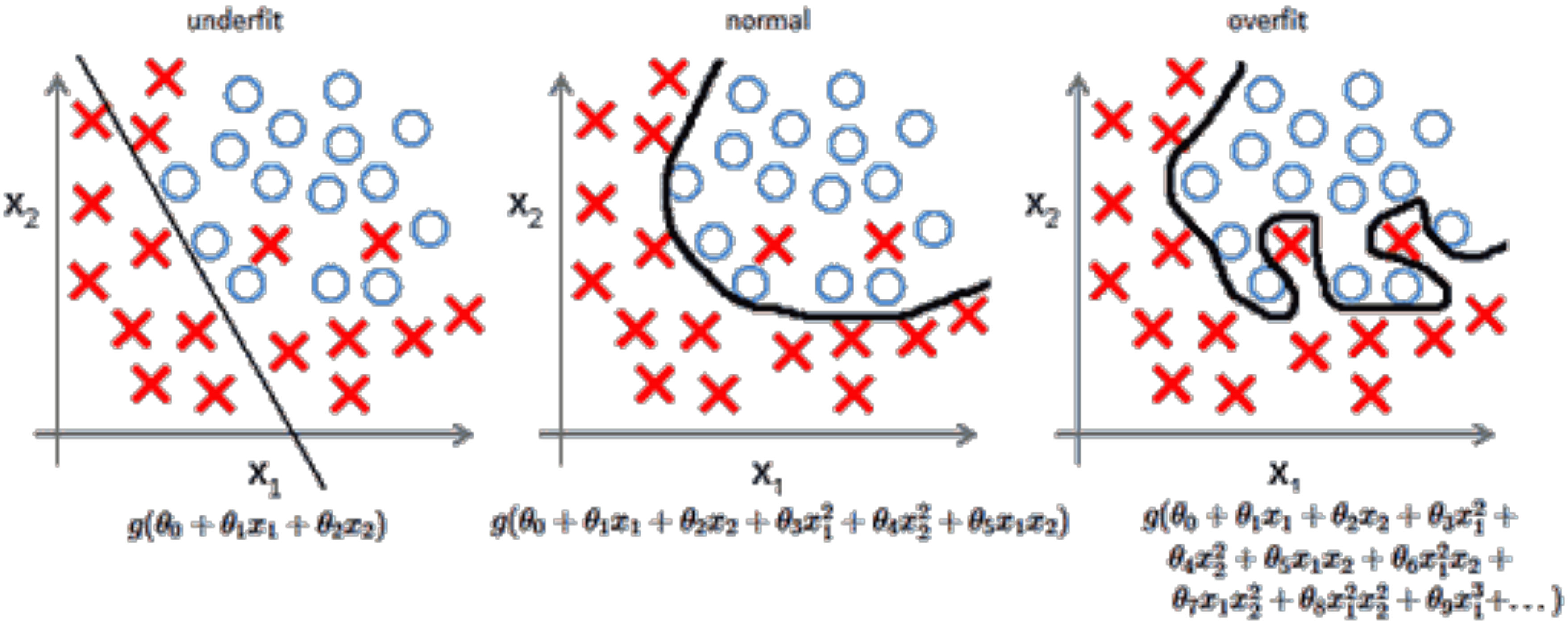
Overfitting

a model is too closely fit to a limited set of data and lose generalization ability

Overfitting

- If we have too many features, the hypothesis may fit the training set very well, but fail to generalize to new examples (high variance)
- More broadly, variance also represents how similar the results from a model will be, if it were fed different data from the same process
- The bias error is from erroneous assumptions in the learning algorithm
- The variance error is from sensitivity to small fluctuations in the training set

Example in Logistic Regression



Underfitting
High bias

Just right

Overfitting
High variance

Address Overfitting

- Feature Reduction
  - Manual selecting which features to keep (by domain knowledge)
  - Okay esp. when some features are really useless

- Regularization
  - Keep all features, but reduce their influence by giving smaller values to the parameter θ_i
  - Okay when many features, each of which contributes a bit to predicting y

Regularized Linear Regression

- Linear Regression

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$J(\theta_0, \theta_1, \dots) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Regularized Linear Regression

$$J(\theta_0, \theta_1, \dots) = \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

- The value of the cost function is NOT equivalent to prediction error. Our goal is to make prediction errors on test data small

Understanding

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- Penalized term: penalize large parameter values $\theta_j, 1 \le j \le n$
- Parameter λ: control the tradeoff
  - Too small: degenerate to linear regression (overfitting)
  - Too large: penalize all features except θ_0, resulting in $h_\theta(x) = \theta_0$ (a horizontal line! underfitting)

Regularized Gradient Descent

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \lambda\theta_j\right)$$

Repeat until convergence {

$$\theta_j = \theta_j\left(1 - \lambda\frac{\alpha}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

}

Types of Regularization Regression

- $||\theta||_2$: Ridge Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- $||\theta||_1$: LASSO Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}|\theta_j|\right]$$

LASSO regression results in sparse solutions – vector with more zero coordinates. Good for high-dimensional problems – don't have to store all coordinates!

Supplement Material: Visual for Ridge Vs. LASSO Regression https://www.youtube.com/watch?v=Xm2C_gTAl8c

Regularized Logistic Regression

- Logistic Regression

$$h_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n)$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

- Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}$$

Regularized Gradient Descent

$$h_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n) = \frac{1}{1 + e^{-(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n)}}$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \lambda\theta_j\right)$$

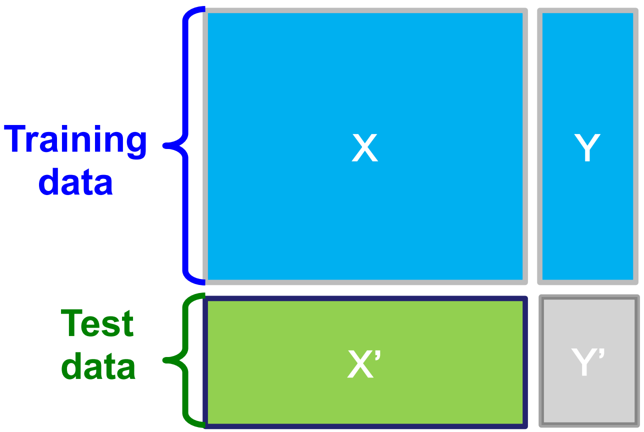
Repeat until convergence {

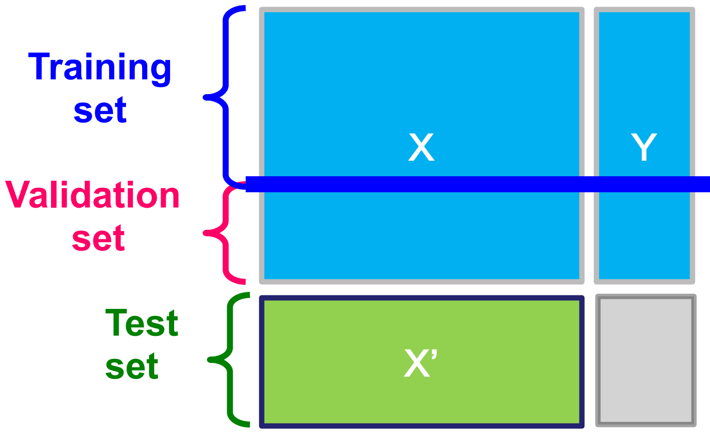
$$\theta_j = \theta_j\left(1 - \lambda\frac{\alpha}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

}

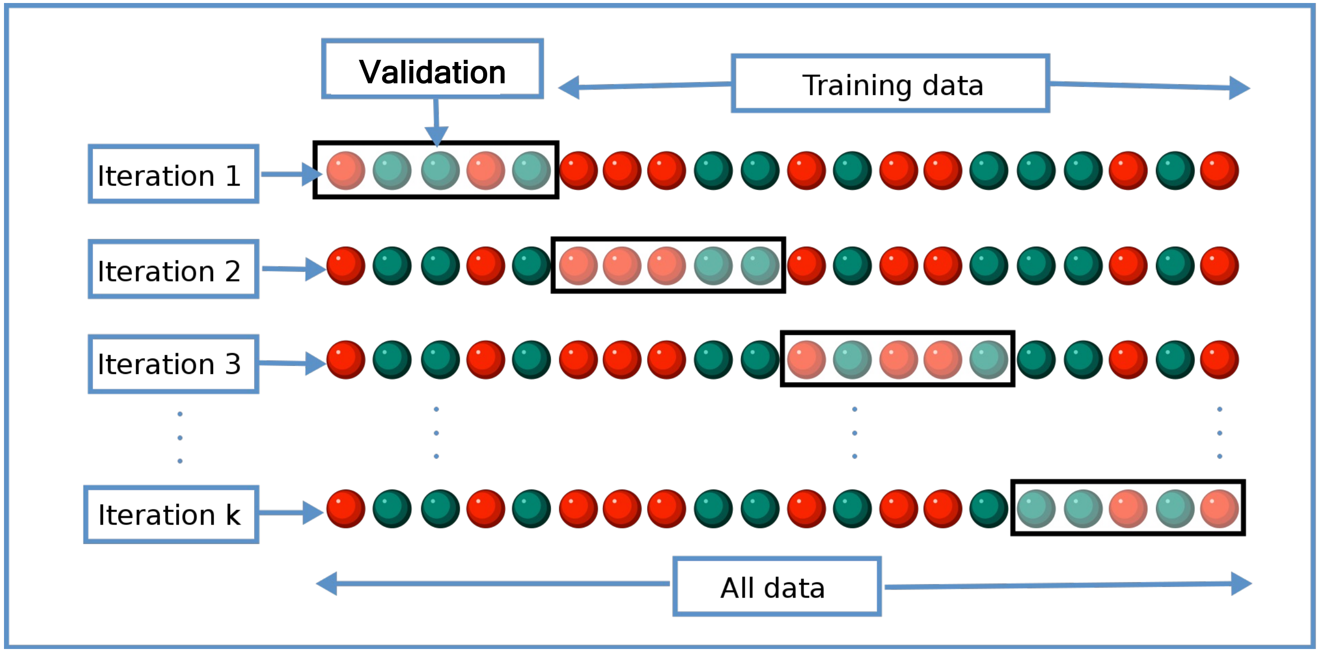
Validation set

- **Task:** Given data **(X,Y)** build a model **f()** to predict **Y'** based on **X'**
- **Strategy:**
  Estimate $y = f(x)$ on (X, Y)
  Hope that the same $f(x)$ also works to predict unknown Y'
  - The **"hope"** is called **generalization**
  - **Overfitting:** If f(x) predicts well Y but is unable to predict Y'
  - We want to build a model that generalizes well to unseen data
  - Solution: **k-fold Cross-validation**



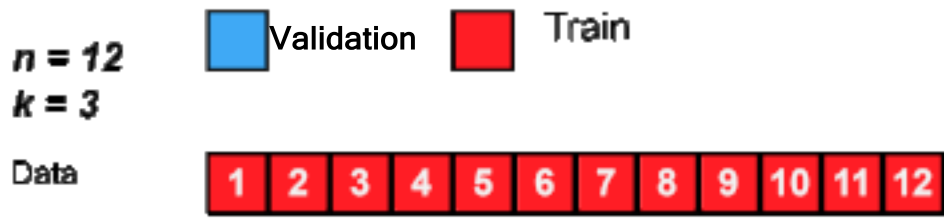


k-fold Cross-validation

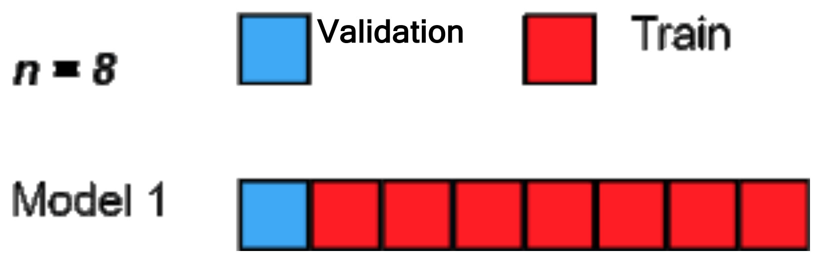


- The original sample is randomly partitioned into k equal sized subsamples
- Of the k subsamples, a single subsample is retained as the validation data for testing the model
- The remaining k – 1 subsamples are used as training data
- The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data
- The k results can then be averaged to produce a single estimation

Leave-one-out Cross-validation



k-fold cross-validation



leave-one-out cross-validation

- When k = n (the number of observations), k-fold cross-validation is equivalent to leave-one-out cross-validation

Boston Housing (has an ethical problem)

The Boston Housing Dataset consists of price of houses in various places in Boston. The Boston Housing Dataset has 506 cases. There are **13** Features in each case of the dataset. Alongside with price, the dataset also provide information such as Crime (CRIM), areas of non-retail business in the town (INDUS), the age of people who own the house (AGE), and there are many other attributes.

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTST	Price
0.006	18.0	2.31	0.0	0.538	6.575	65.2	4.090	1.0	296.0	15.3	396.9	4.98	24.0
0.027	0	7.07	0.0	0.469	6.421	78.9	4.967	2.0	242.0	17.8	396.9	9.14	21.6
…	…	…	…	…	…	…	…	…	…	…	…	…	…

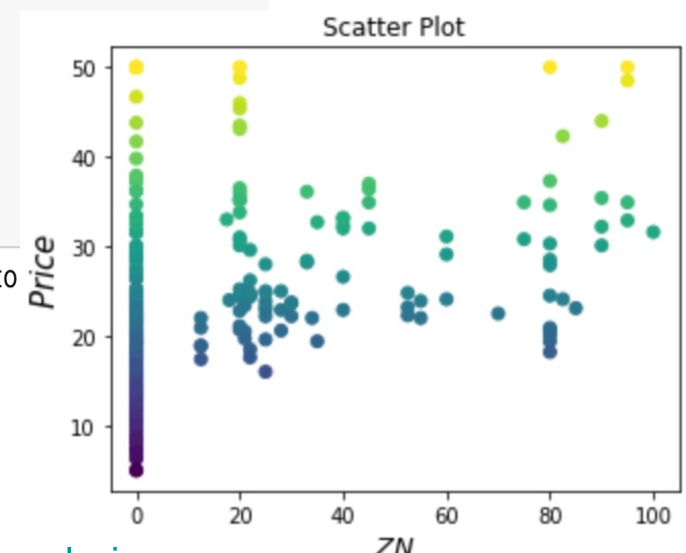
Generate Training Data

```
In [41]: import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.datasets import load_boston, load_diabetes
         from sklearn.model_selection import train_test_split

         np.random.seed(42)

         def load_data():
             dataset = load_boston()
             print(dataset.feature_names)
             return train_test_split(dataset.data, dataset.target, test_size=0.25, random_state=0)

         X_train, X_test, Y_train, Y_test = load_data()
         print(X_train.shape)

         plt.figure(figsize=(5,4))
         plt.scatter(X[:,1],y,c=y)
         plt.ylabel("$Price$", fontsize=15)
         plt.xlabel("$ZN$", rotation=0, fontsize=15)
         plt.title('Scatter Plot')
         plt.show()

         ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO
          'B' 'LSTAT']
         (379, 13)
```

Boston Housing Data

Split dataset

Plot figure



ZN: Proportion of residential land zoned for lots over 25,000 sq. ft

https://www.kaggle.com/tolgahancepel/boston-housing-regression-analysis

Build Model

```
In [56]: from sklearn.linear_model import Ridge
         from sklearn.model_selection import cross_val_score

         alpha = 0
         model = Ridge(alpha=alpha, solver='auto', random_state=42)

         model.fit(X_train, Y_train)
         Y_pred = model.predict(X_test)
         cross_valid = cross_val_score(model, data, target, scoring='neg_mean_squared_error', cv = 5)
         print('Cross Validation Errors:\n', -np.mean(cross_valid))
         print('theta 0: \n', model.intercept_)
         print('theta 1-13: \n', model.coef_)
```

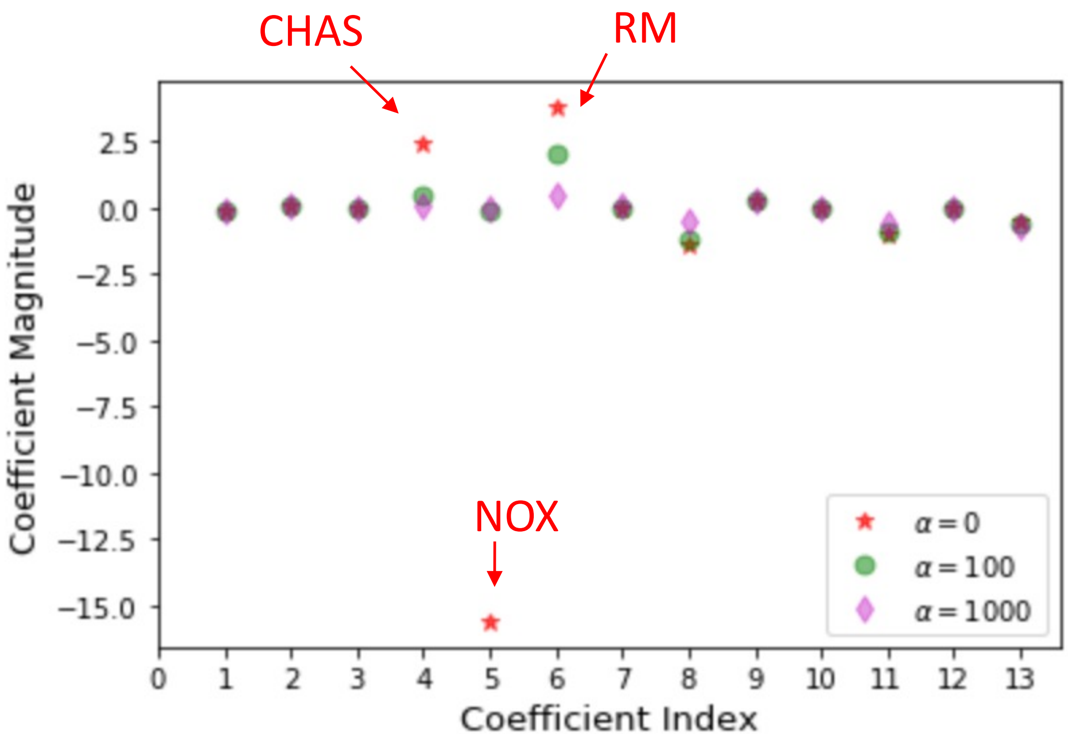
Train model → `model.fit(X_train, Y_train)`

Cross validation → `cross_val_score`

```
Cross Validation Errors:
 37.13180746769889
theta 0:
 36.933255457119316
theta 1-13:
 [-1.17735289e-01  4.40174969e-02 -5.76814314e-03  2.39341594e+00
 -1.55894211e+01  3.76896770e+00 -7.03517828e-03 -1.43495641e+00
  2.40081086e-01 -1.12972810e-02 -9.85546732e-01  8.44443453e-03
 -4.99116797e-01]
```

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_{13} x_{13}$$

Regularization



α	MSE
0	37.1318 (overfitting)
100	29.9057
1000	32.8280 (underfitting)

The magnitudes of coefficient indices 4,5,6 are considerably reduced after regularization with α = 100, resulting in lower mean square error

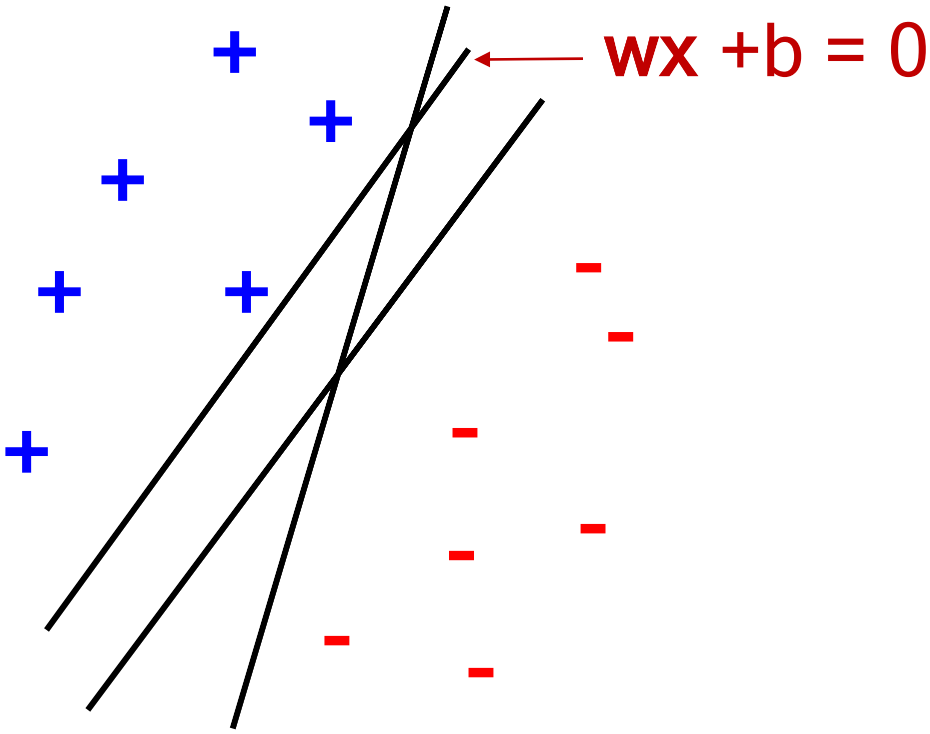
History of Support Vector Machines

- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details

[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

Support Vector Machines

- Want to separate "+" from "-" using a line

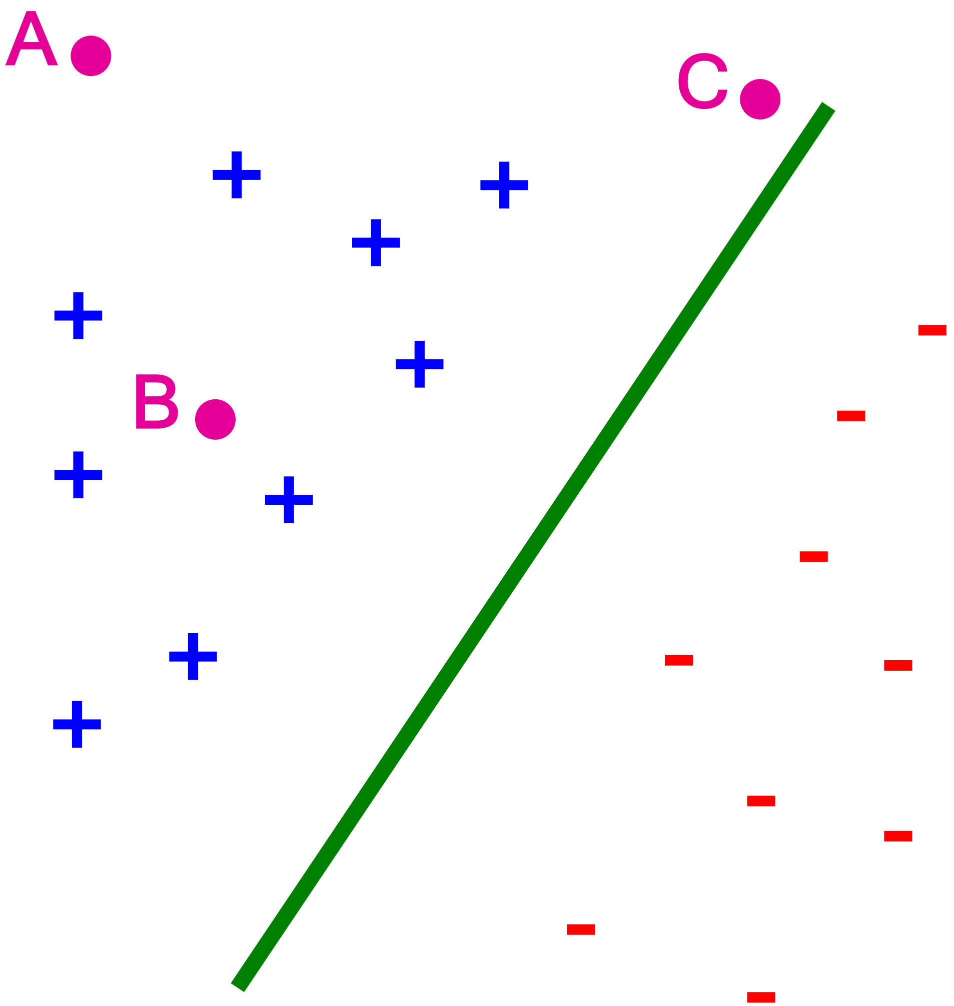


Data:

- **Training examples:**
  - $(x^{(1)}, y_1) \ldots (x^{(m)}, y_m)$
- **Each example *i*:**
  - $x^{(i)} = (x_1^{(i)},\ldots , x_d^{(i)})$
    - $x_j^{(i)}$ is real valued
  - $y_i \in \{ -1, +1 \}$

Which is best linear separator (defined by *w*)?

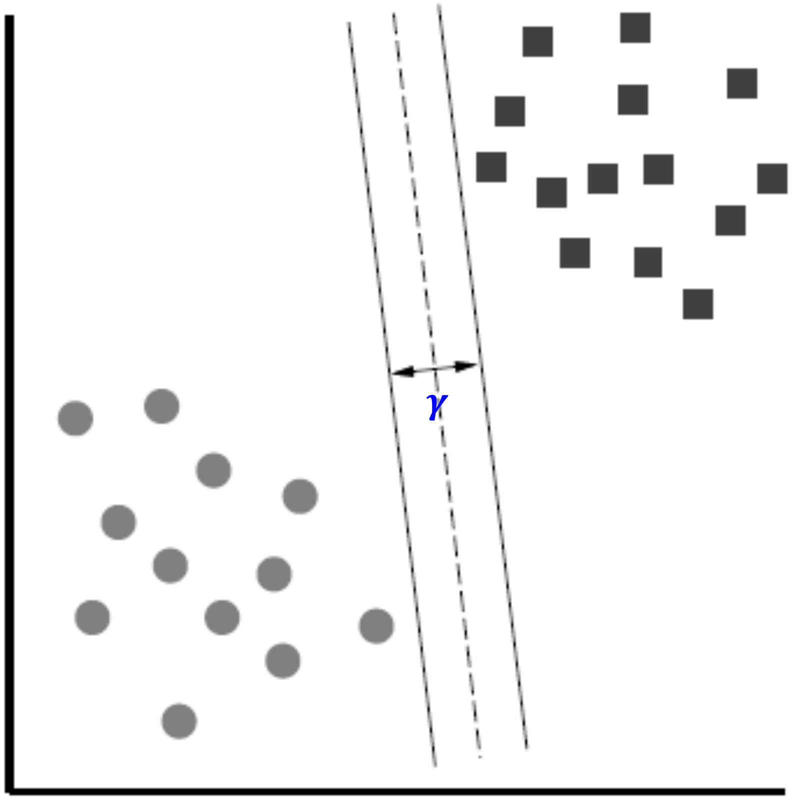
Largest Margin

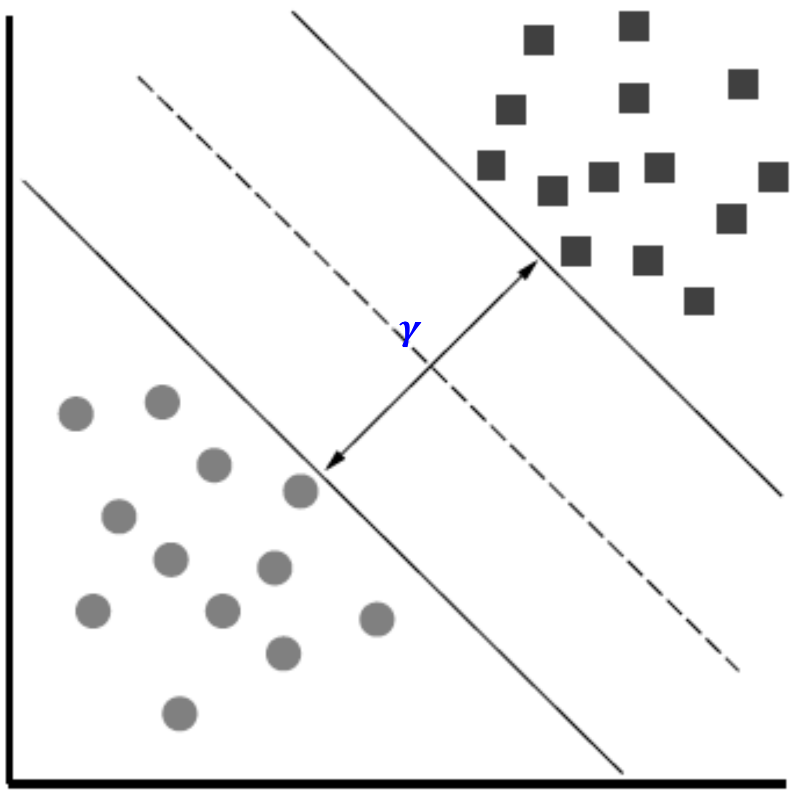


- Distance from the separating hyperplane corresponds to the "confidence" of prediction
- **Example:**
  - We are more confident about the class of **A** and **B** than of **C**

Largest Margin

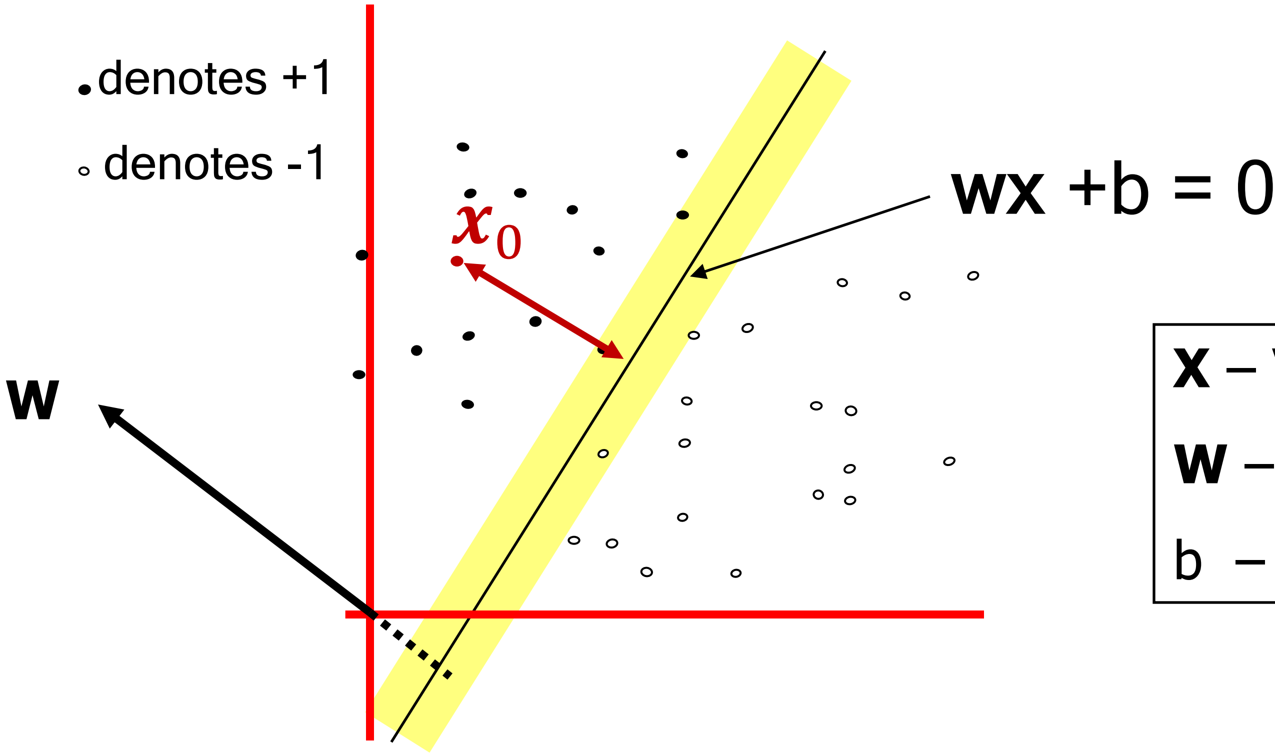
- **Margin γ (gamma): Distance of closest example from the decision line/hyperplane**





The reason we define margin this way is due to theoretical convenience and existence of generalization error bounds that depend on the value of margin.

Distance from a point to a line



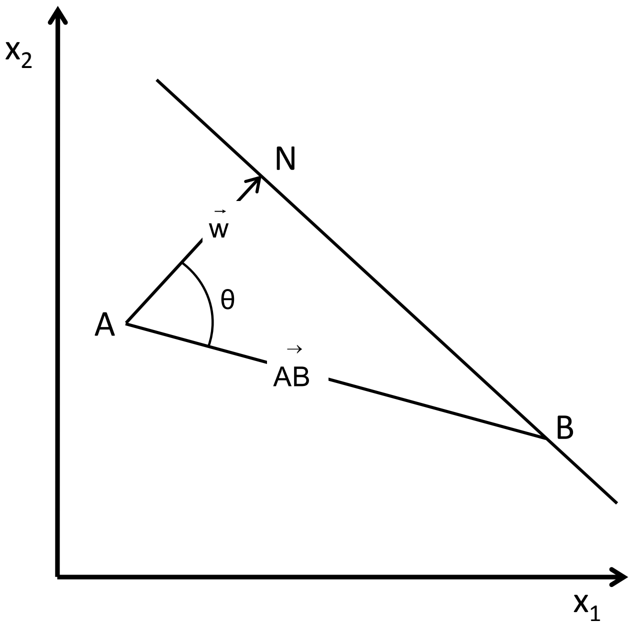
x – Vector
w – Normal Vector
b – Scale Value

- What is the distance expression for a point x_0 to a line **wx**+b= 0?

$$d(\mathbf{x}_0) = \frac{|\mathbf{x}_0 \cdot \mathbf{w} + b|}{\sqrt{||\mathbf{w}||_2^2}} = \frac{|\mathbf{x}_0 \cdot \mathbf{w} + b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

- http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

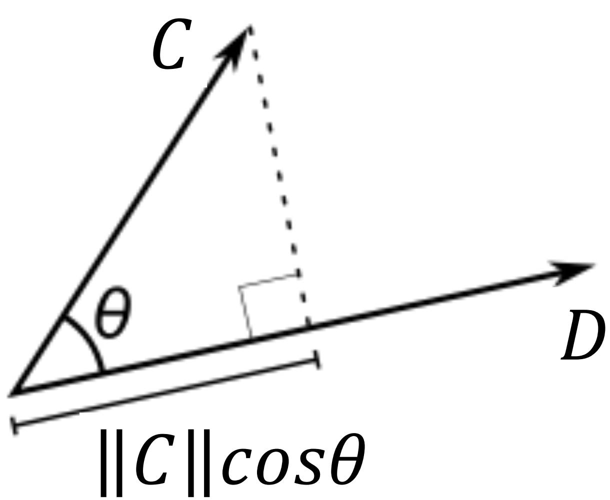
Distance from a point to a line (method 2)



$$\|\overrightarrow{AN}\| = \|\overrightarrow{AB}\|\cos\theta = \|\overrightarrow{AB}\| \frac{\overrightarrow{AB}\,\vec{w}}{\|\overrightarrow{AB}\|\,\|w\|} = \frac{\overrightarrow{AB}\,\vec{w}}{\|w\|}$$

$$= \frac{(x_{B1} - x_{A1}, x_{B2} - x_{A2})^\top(-w_1, -w_2)}{\|w\|}$$

$$= \frac{w^T x_A - \overbrace{w^T x_B}^{=-b}}{\|w\|} = \frac{w^T x_A + b}{\|w\|}$$

- 

$$C \cdot D = \|C\| \cdot \|D\| \cdot \cos\theta$$

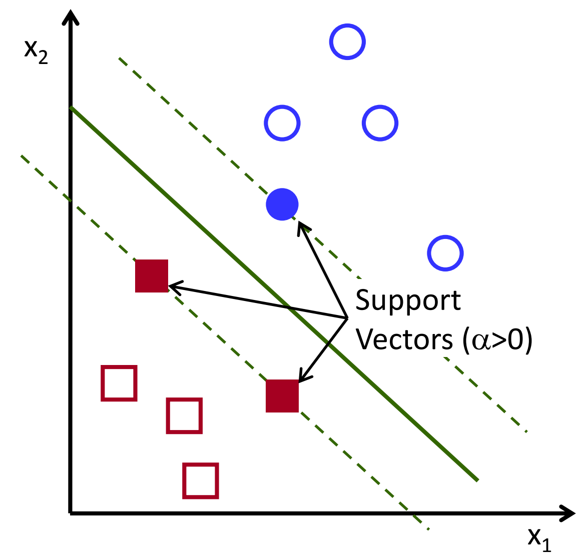
Linear SVM Mathematically

- Let training set $\{(\mathbf{x}^{(i)}, y_i)\}_{i=1..n}$, $\mathbf{x}^{(i)} \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin γ. Then for each training example $(\mathbf{x}^{(i)}, y_i)$:

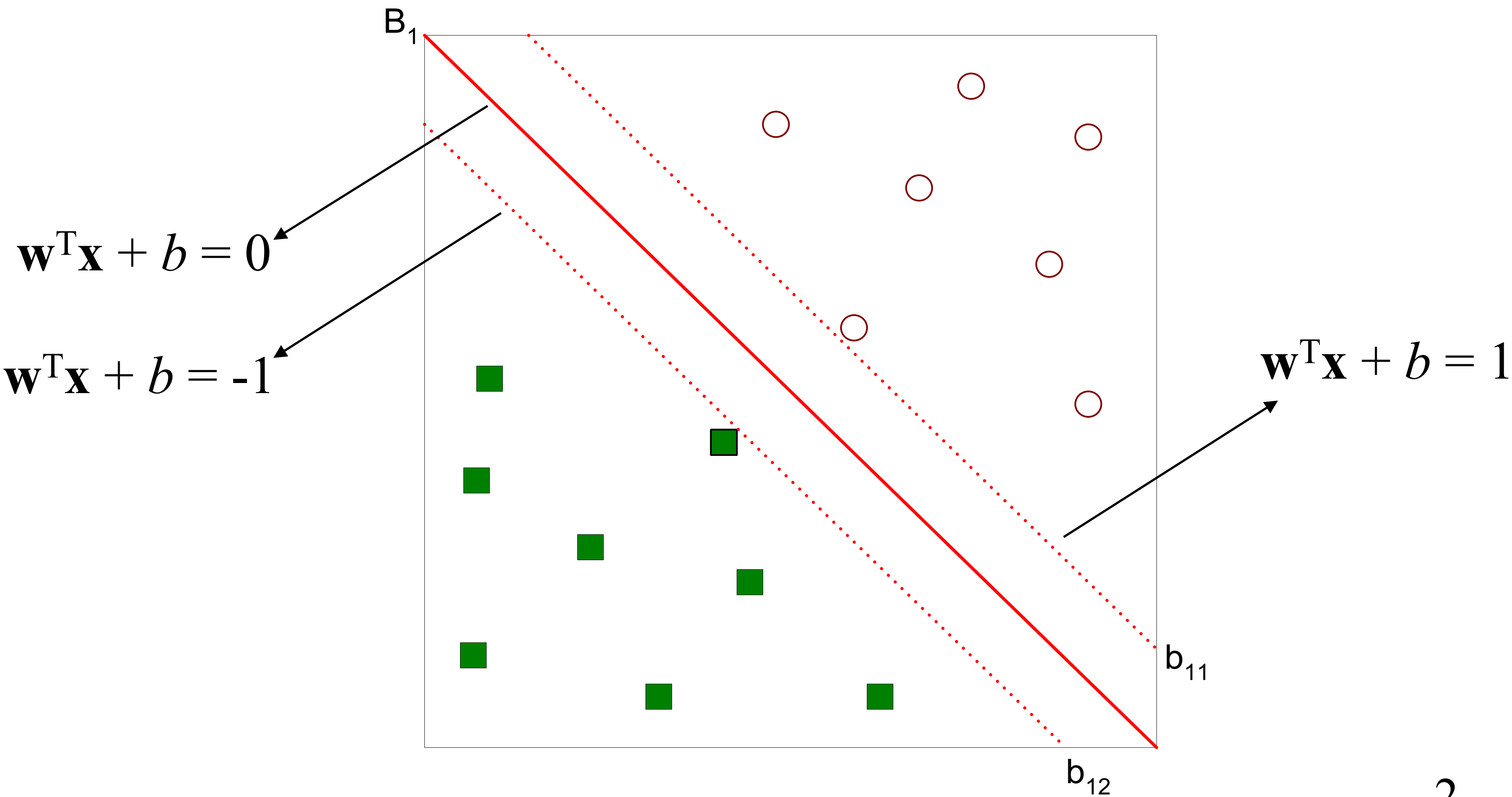
$$\begin{aligned} \mathbf{w}^T\mathbf{x}^{(i)} + b &\leq -\gamma/2 \quad \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}^{(i)} + b &\geq \gamma/2 \quad \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq \gamma/2$$

- For every support vector $\mathbf{x}^{(s)}$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\gamma/2$ in the equality, we obtain that distance between each $\mathbf{x}^{(s)}$ and the hyperplane is

$$\frac{y_s(\mathbf{w}^\top \mathbf{x}^{(s)} + b)}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||}$$



Linear Support Vector Machine (SVM)



$$y_i = \begin{cases} -1, & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{i})} + b \leq -1 \\ 1, & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{i})} + b \geq 1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Linear SVM Mathematically (cont.)

- Then the margin can be expressed through (rescaled) **w** and b as:

$$\text{New margin } \gamma' = \frac{2}{\|\mathbf{w}\|}$$

- Then we can formulate the *quadratic optimization problem:*

Find **w** and b such that

$$\gamma' = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}^{(i)}, y_i)$, $i = 1 \ldots m$: $y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$

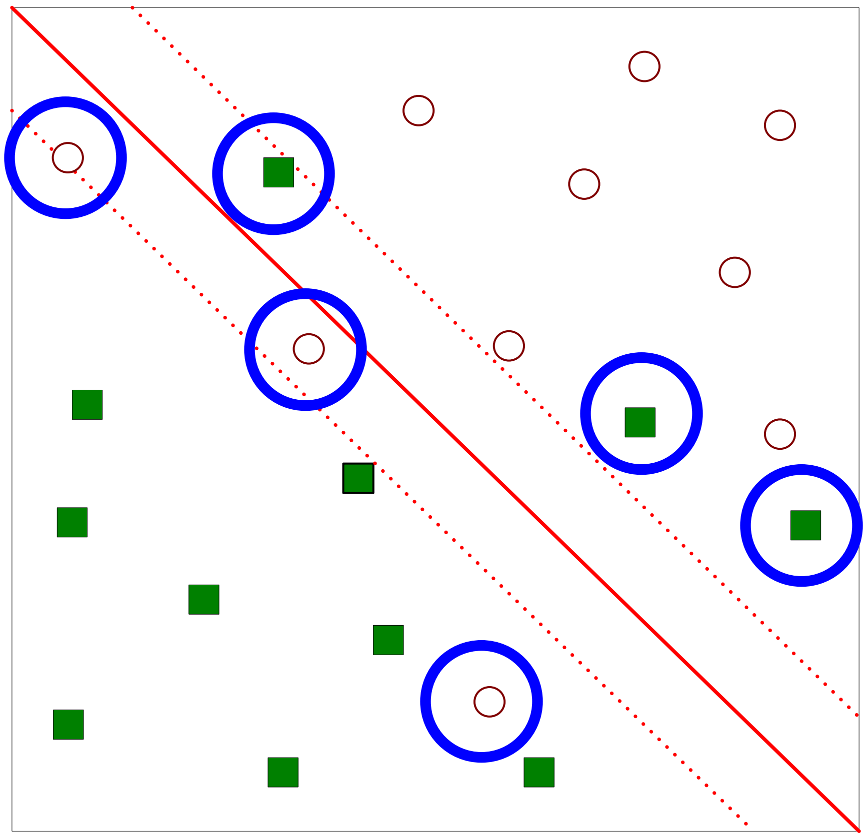
- Which can be reformulated as:

Find **w** and b such that

$$\boldsymbol{\Phi}(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \text{ is minimized}$$

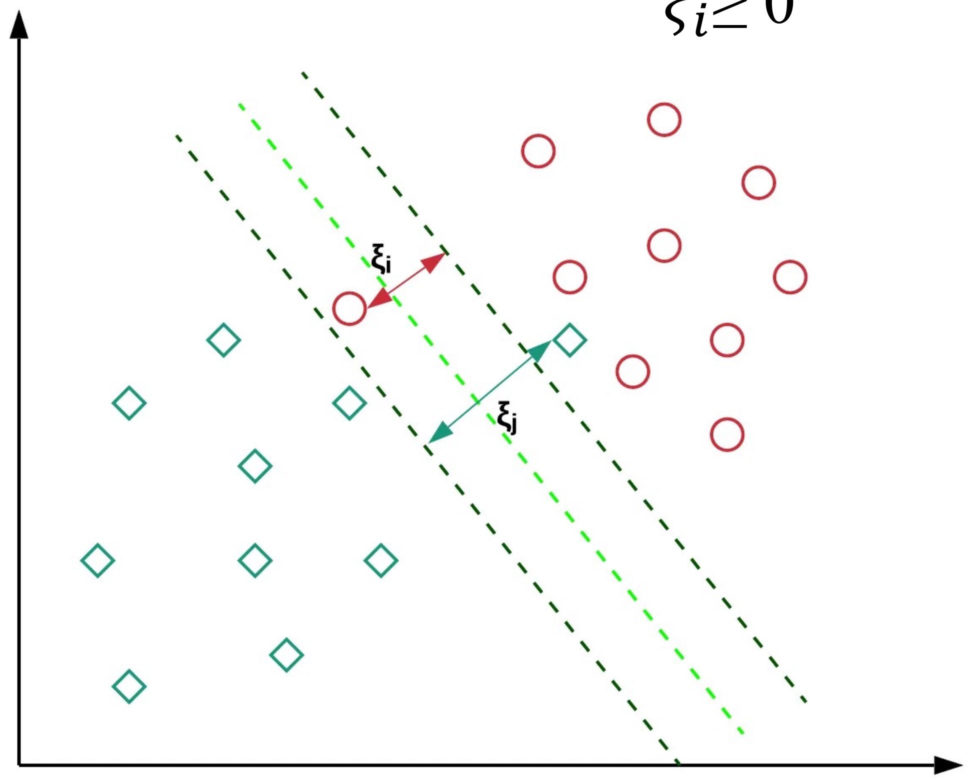
and for all $(\mathbf{x}^{(i)}, y_i)$, $i=1 \ldots m$: $y_i (\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$

What if the problem is not linearly separable



SVM with soft margin

- Need to minimize: $$\frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{i=1}^{m} \xi_i^2\right)$$
- subject to: $$\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{i})} + b \leq -1 + \xi_i \quad \text{if } y_i = -1$$ $$\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(\mathrm{i})} + b \geq 1 - \xi_i \quad \text{if } y_i = 1$$ $$\xi_i \geq 0$$



Characteristics of SVM

- The learning problem is formulated as a convex optimization problem
- Efficient algorithms are available to find the global minima
- High computational complexity for building the model
- Robust to noise
- Overfitting is handled by maximizing the margin of the decision boundary
- SVM can handle irrelevant and redundant attributes better than many other techniques
- The user needs to provide the type of kernel function & cost function (for nonlinear SVM)
- Difficult to handle missing values