Machine Learning & Data Mining

Classification

Kyung-Ah Sohn

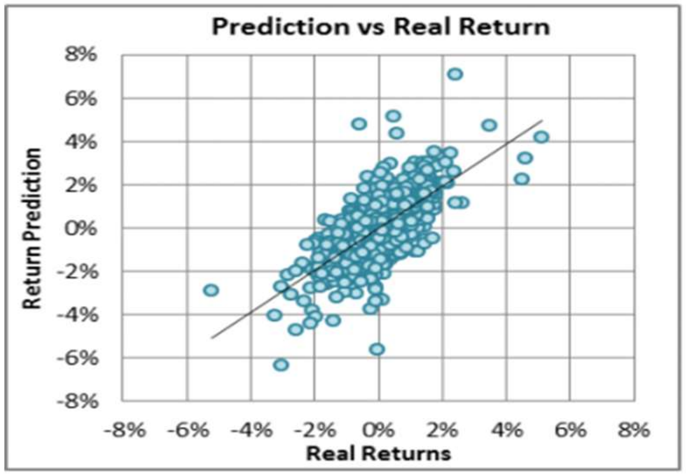
Ajou University

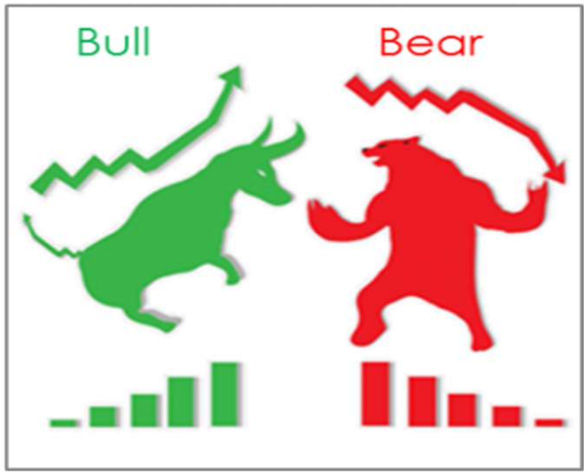
Content

INTRODUCTION TO SUPERVISED LEARNING

Supervised Learning

Regression vs **Classification**





Supervised learning approach

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$

Words in a document

“Sports”
“News”
“Science”
…

Discrete Labels
Classification

DJ INDU AVERAGE (DOW JONES & CO
as of 22-Jan-2010
11000
10500
10000
9500
Nov11
Dec01
Dec21
Jan08
Copyright 2010 Yahoo! Inc.
http://finance.yahoo.com/

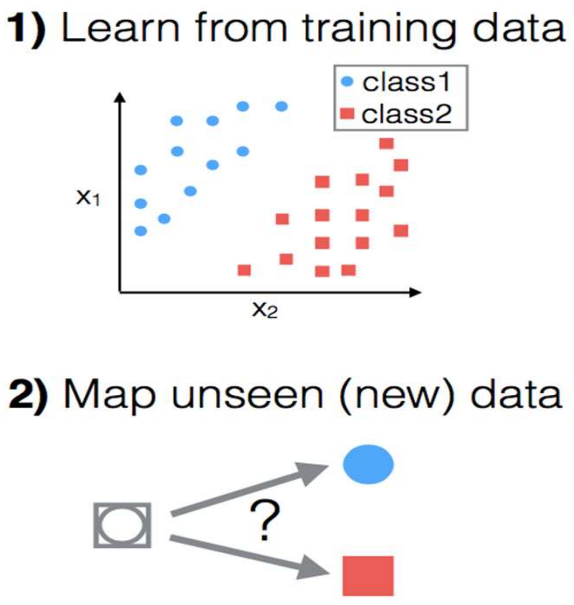
Market information
up to time t

Share Price
“$ 24.50”

Continuous Labels
Regression

Task: Given $X \in \mathcal{X}$, predict $Y \in \mathcal{Y}$.

Classification: Training vs. Test



Explain whether each scenario is a classification or regression problem, and provide n (# of samples) and p (# of features).

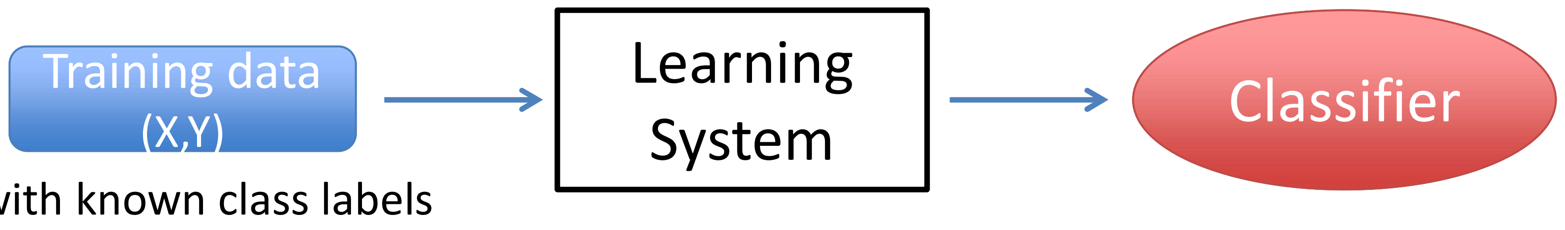
(a) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.

(b) We are considering launching a new product and wish to know whether it will be a *success* or a *failure*. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.

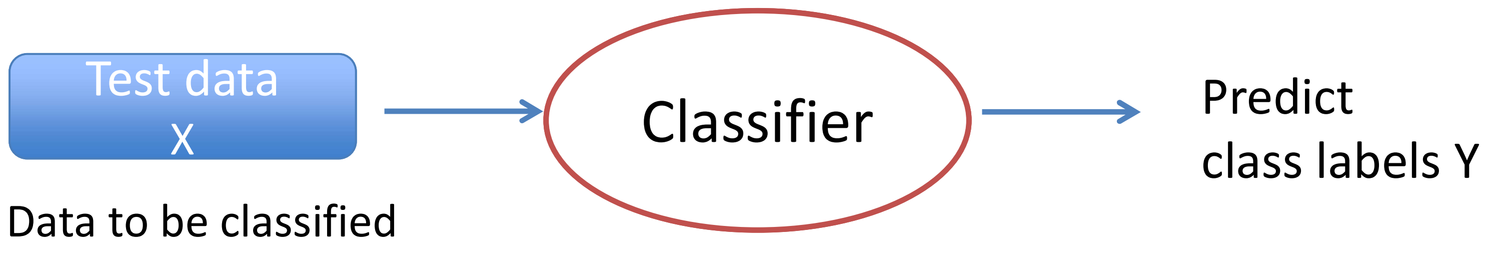
(c) We are interesting in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.

Classification

- **Learning**: Induce classifiers from training data



- **Prediction**: use the learned classifier to predict labels for new data



Training vs. Test data

- Training data: to learn a classifier (using known labels)

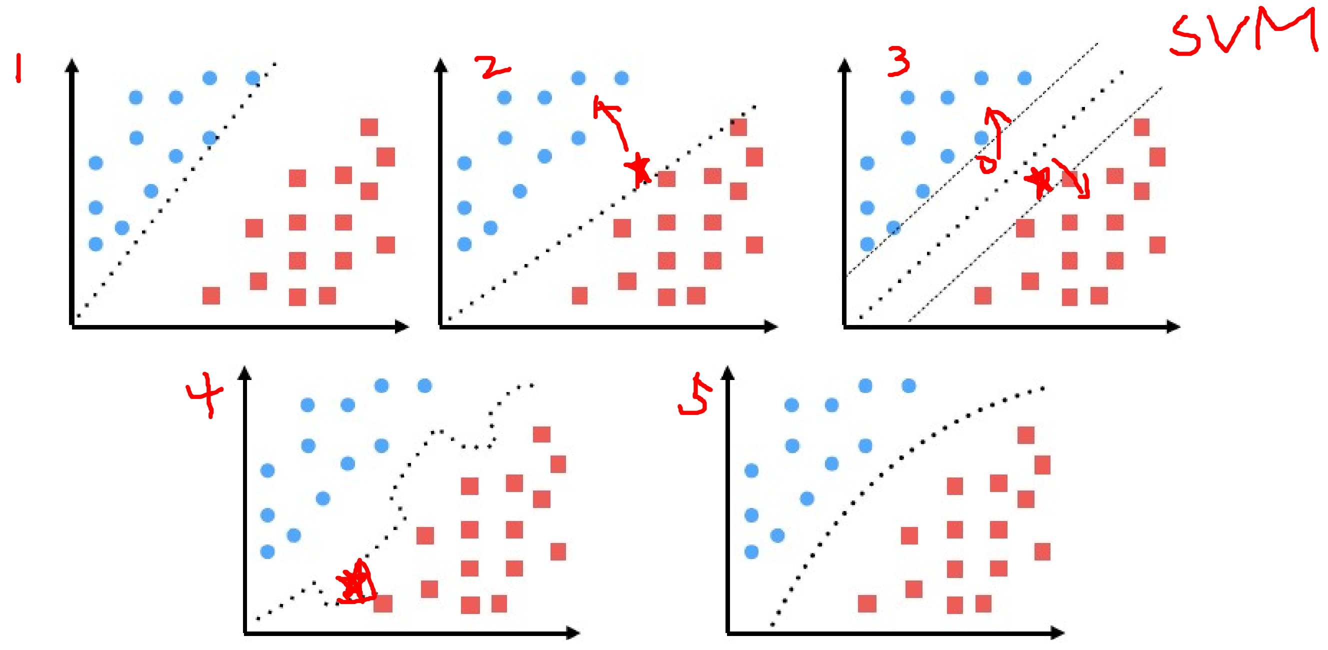
	team	coach	play	ball	score	game	win	goal	label
Doc_1	3	9	0	2	5	7	0	0	**1**
Doc_2	0	7	5	4	0	1	2	1	**0**
…				…					**…**
Doc_n	0	1	0	0	1	2	2	0	**1**

- Test data: to predict a label or to evaluate the classifier

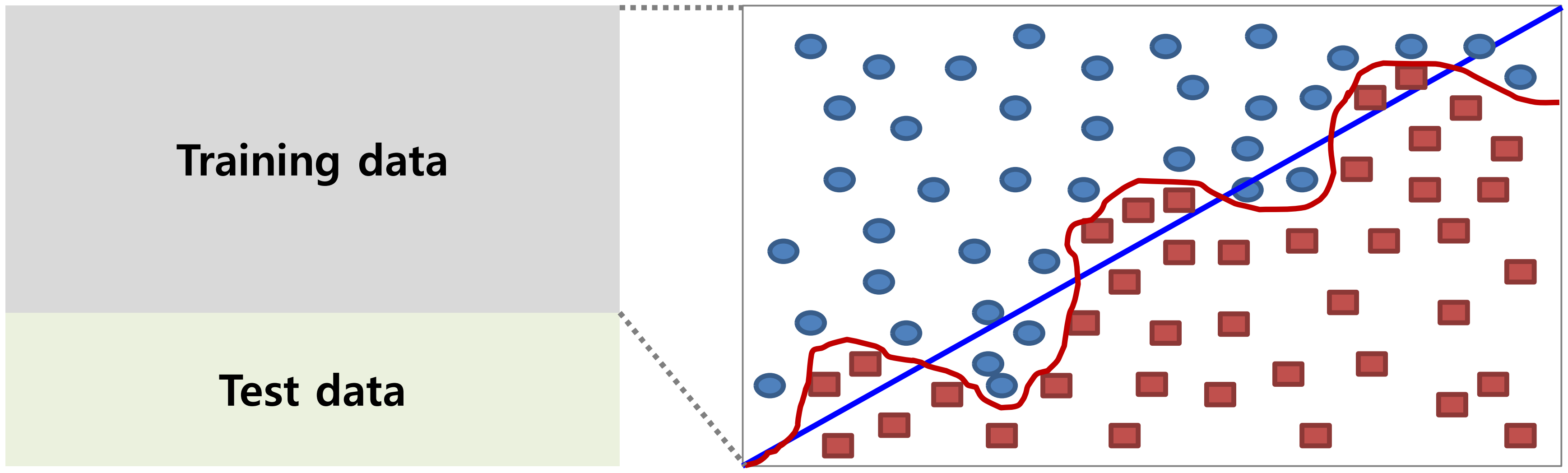
	team	coach	play	ball	score	game	win	goal	label
Doc_new	3	9	0	2	5	7	0	0	**?**

Model selection

- Which one is better?

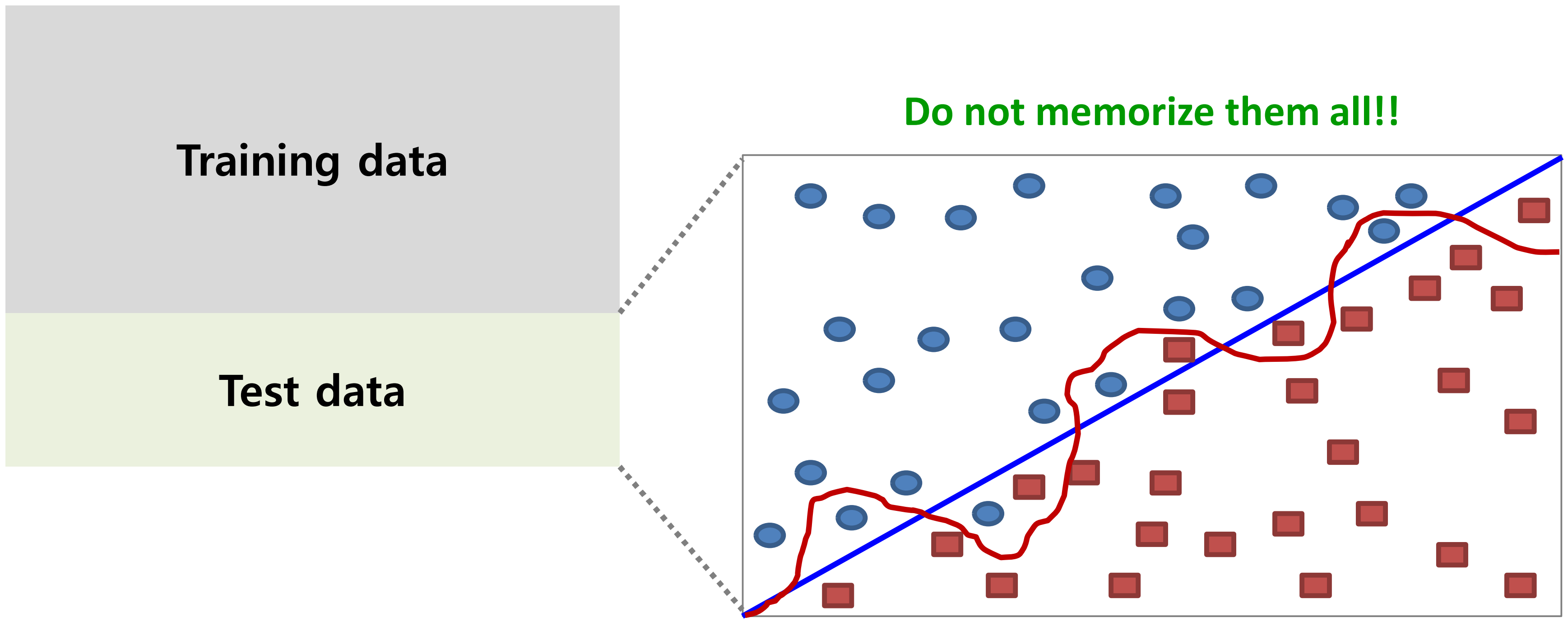


Over-fitting for training data



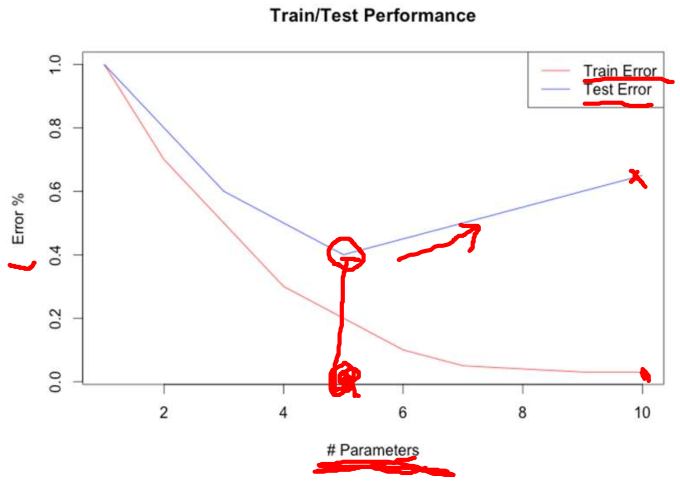
Is red boundary is better than blue one?

❖ Over-fitting for training data

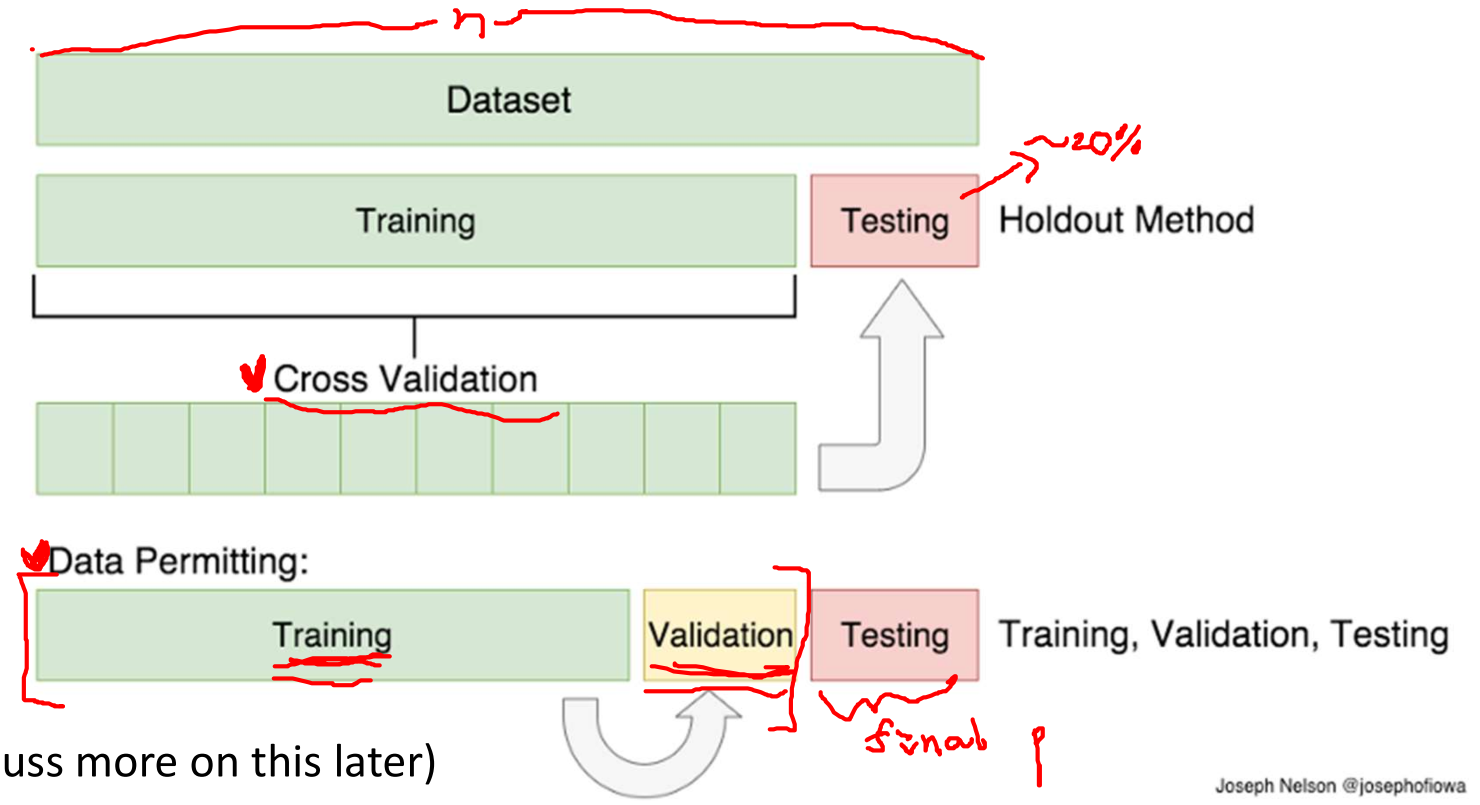


Train/Test split

- The performance of the classifier is measured on the test set
- What if the test set is the same as the training set?
  - 'training error', 'training 'accuracy' vs. 'test error', 'test accuracy'
- Is the training error always zero?
- Why do we need a separate test set?



Train/Validation/Test set



(will discuss more on this later)

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Classification algorithms

- K-Nearest Neighbor
- Support Vector Machine (SVM)
- Decision tree, Random Forest
- Naïve Bayes
- Logistic regression
- Neural networks, DL
- Bayesian networks
- …

K-NEAREST NEIGHBOR ALGORITHM

Classification example

Training data

No.	v_1=Strength	v_2=Smooth	y=Quality
x_1	5	5	Good
x_2	1	2	Bad
x_3	6	7	Good
x_4	3	5	Bad

Test data

x_5	2	3	?

k-Nearest Neighbors (k-NN) classification

- Determine k and the distance metric to use
- For a given test sample
  1. Calculate *distances* of all training samples to the test sample
  2. Pick *k* closest training samples
  3. Calculate majority

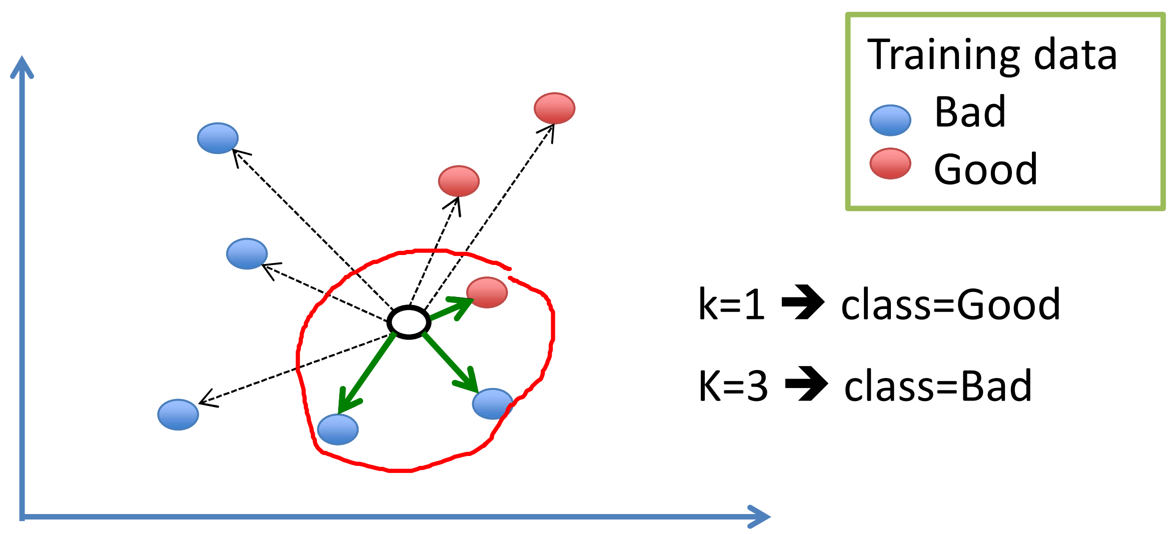
Classifiction example

- Compute the distance between test data and all the training samples

	No.	v_1=Strength	v_2=Smooth	y=Quality
	x_1	5	5	Good
1	x_2	1	2	Bad
	x_3	6	7	Good
2	x_4	3	5	Bad
	x_5	2	3	?

$$d(\mathbf{x}_1, \mathbf{x}_5) = \sqrt{(5-2)^2 + (5-3)^2} = \underline{3.6}$$
$$d(\mathbf{x}_2, \mathbf{x}_5) = \sqrt{(1-2)^2 + (2-3)^2} = \underline{1.4}$$
$$d(\mathbf{x}_3, \mathbf{x}_5) = \sqrt{(6-2)^2 + (7-3)^2} = 5.6$$
$$d(\mathbf{x}_4, \mathbf{x}_5) = \sqrt{(3-2)^2 + (5-3)^2} = 2.2$$

Example



k=1 ➔ class=Good

K=3 ➔ class=Bad

Computation for KNN

- What is done in training stage?

- How many computations to predict the label of each sample?

Instance-based learning, lazy learning

K-NN algorithm

- 1-NN
  – Predict the same class label as the *nearest* instance in the training set

- In general, k-NN
  – Find the *k* closest training points (according to some distance measures, e.g. Euclidean, …)
  – Predicted class: majority vote of k-neighbors

Neighbor size k

- Choice of k
  - Smaller k ➔ higher variance (less stable)
  - Larger k ➔ higher bias (less precise)

  - Proper choice of k depends on the dataset
    - Heuristics
    - Cross-validation

Neighbor size – extreme cases

- What is the training error (the error when each training sample is used as a test sample) when K=1?

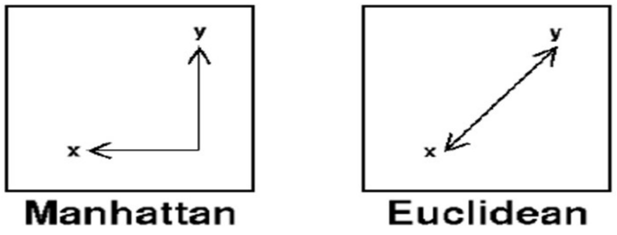
- What happens if K=n (the number of training samples)

k-NN

- No explicit training or model, “lazy learning”
- One of the simplest classification methods
- *Gives the maximum likelihood estimation of the class posterior probabilities*
- Can be used as a baseline model
- Many extensions

Choice of Distance Measure

- Euclidean distance
- Manhattan distance
- L_p norm
- Mahalanobis distance
- 1 - Correlation
- 1 - Cosine similarity
- …



Data and Application-dependent

Euclidean Distance

- n objects with d attributes
  - $\boldsymbol{x}_i = (x_1(i), x_2(i), \ldots, x_d(i))$, i=1,…,n
- Most common distance metric is Euclidean distance

$$d_E(i,j) = \left(\sum_{k=1}^{d} (x_{\underline{k}}(i) - \underline{x}_k(j))^2 \right)^{\frac{1}{2}}$$

Example

No.	v_1=Strength	v_2=duration	y=Quality
1	5	5000	Good
2	1	4900	Bad
3	6	2900	Good
4	3	5200	Bad
5	2	3300	?

Euclidean distance??

Euclidean distance

- Makes sense in the case where the different measurements are commensurate; each variable measured in the same units

- If the measurements are different, say length in different units, Euclidean distance is not necessarily meaningful
  - Standardization!

Normalization

- Usually assume features should contribute equally.
  - Min-max normalization

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

  - Z-score standardization

$$\mathrm{X}_{new} = \frac{\mathrm{X} - \mu}{\sigma} = \frac{\mathrm{X} - \mathrm{Mean}(\mathrm{X})}{\mathrm{StdDev}(\mathrm{X})}$$

Extensions

- L_p metric:

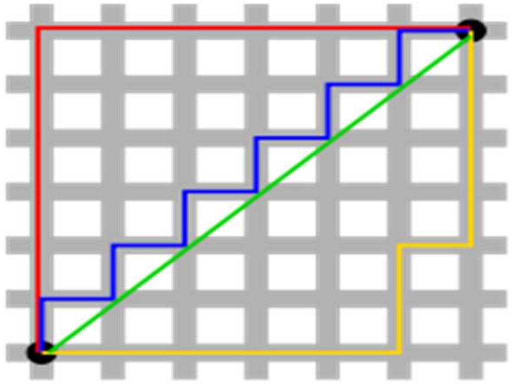
$$dist(i,j) = \left(\sum_{k=1}^{d} | x_k(i) - x_k(j) |^p \right)^{\frac{1}{p}}$$ where p >=1

- Manhattan, city block or L_1 metric:

$$dist(i,j) = \sum_{k=1}^{d} \left| x_k(i) - x_k(j) \right|$$

- L_∞

$$dist(i,j) = \max_k \left| x_k(i) - x_k(j) \right|$$



Mahalanobis distance

- The Mahalanobis distance takes into account the correlations of the data set.
- If each of these axes is re-scaled to have unit variance, then the Mahalanobis distance corresponds to the standard Euclidean distance

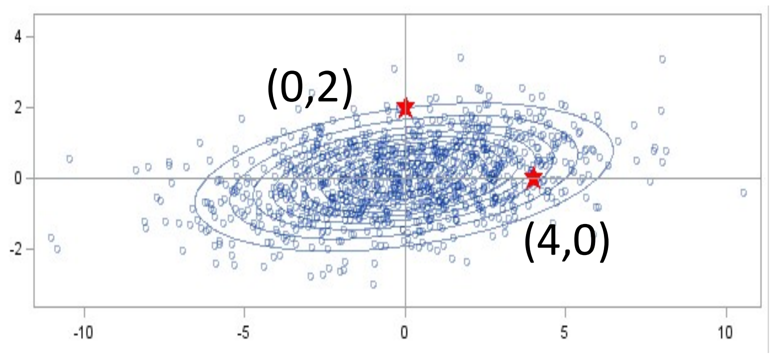
$$d(X,Y) = \sqrt{(X-Y)^T S^{-1} (X-Y)}$$

S: Covariance matrix

- If the covariance matrix is the identity matrix:

$$d(X,Y) = \sqrt{\sum_{i=1}^{m} \frac{(x_i - y_i)^2}{{s_i}^2}}$$

- Which point is closer to the origin (0,0)?



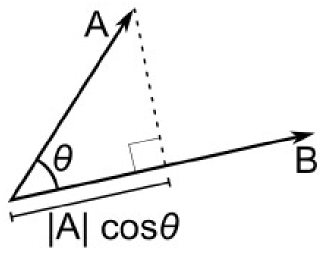
Widely used similarity measures

- Cosine similarity
  - Consider only the direction between the two vectors (not the magnitude)
  - Compute using dot product (inner product)

$$S_{cos}(X,Y) = cos\theta = \frac{X \cdot Y}{||X||\,||Y||}$$

$$\mathrm{X} \cdot Y = x_1 y_1 + x_2 y_2 + \cdots + x_m y_m$$

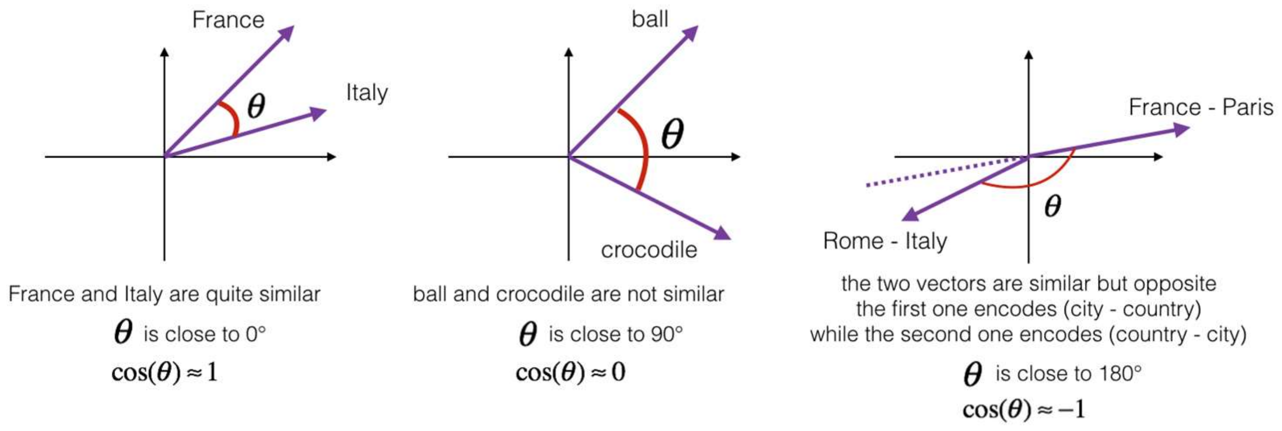
$$||\mathrm{X}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_m^2}$$



Cosine similarity

- $S_{cos}((1,1),(2,2)) =$
- $S_{cos}((1,1),(3,3)) =$
- $S_{cos}((1,1),(-1,1)) =$

$$S_{cos}(X,Y) = cos\theta = \frac{X \cdot Y}{||X||||Y||}$$


France
Italy
θ
France and Italy are quite similar
θ is close to 0°
$\cos(\theta) \approx 1$
ball
crocodile
θ
ball and crocodile are not similar
θ is close to 90°
$\cos(\theta) \approx 0$
France - Paris
Rome - Italy
θ
the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)
θ is close to 180°
$\cos(\theta) \approx -1$

https://datascience-enthusiast.com/DL/Operations_on_word_vectors.html

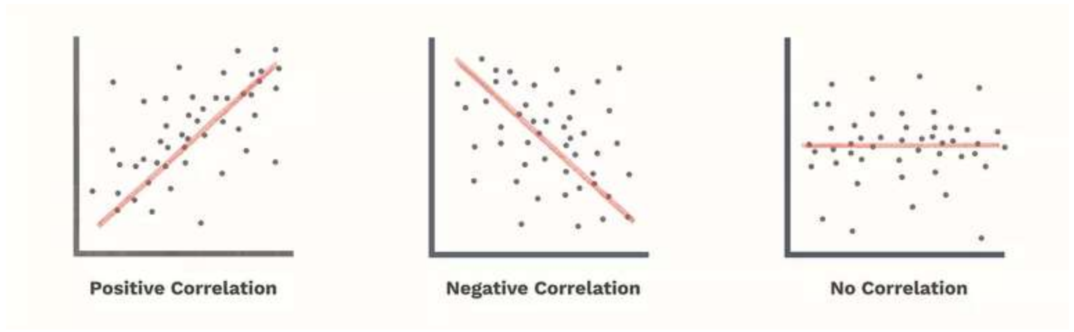
(Pearson's) correlation coefficient

- Similarity measure
  - Ranges between -1 and 1
  - measures the strength and direction of the linear relationship between two variables.

$$\rho(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}}$$

$$\overline{x} = \frac{1}{n}\sum_{i}^{n} x_i, \quad \overline{y} = \frac{1}{n}\sum_{i}^{n} y_i$$

$$s_\rho(X,Y) = 0.5\times(1-\rho(X,Y))$$



K-NN (Pros)

- + Easy to understand and program
- + Explicit reject option
  - o If there is no majority agreement
- + Easy handing of missing values
- + Close to optimal
  - o *Asymptotic misclassification rate (as the number of data points goes to infinity) is bounded above by twice the Bayes error rate*

K-NN (Cons)

- *Sensitive to noise, irrelevant features*
- Computationally expensive O(*np*)
- *Large memory requirements*
- More frequent classes dominate result
- Curse of dimensionality
    - “nearest” neighbor may be very far
    - In high dimensions, “nearest” becomes meaningless

Curse of dimensionality

- The size of the data *space* grows exponentially with the number of dimensions
- The size of your data set must also grow exponentially in order to keep the same density…

- In KNN algorithm, it requires two points to be very close on *every* axis
  - Adding a new dimension creates another chance for points to be farther apart
- How to overcome in KNN
  - Add more data…
  - Reduce dimension (feature selection, feature extraction)