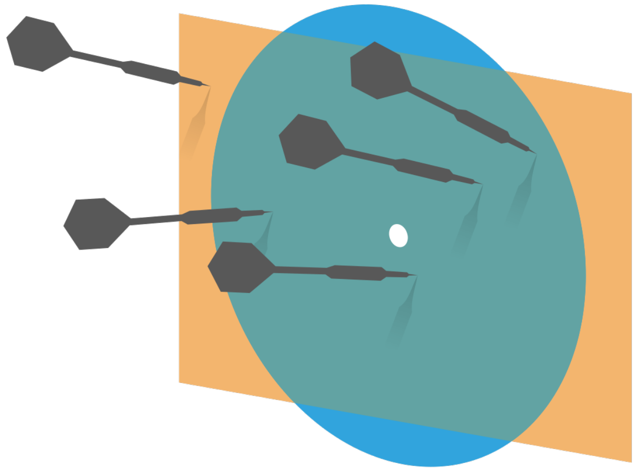
Probability and Statistics for Computer Science



"...many problems are naturally classification problems"---Prof. Forsyth

Credit: wikipedia

Hongye Liu, Teaching Assistant Prof, CS361, UIUC, 11.5.2020

Last time

- Demo of Principal Component Analysis
- Introduction to classification

Objectives

- Decision tree (II)
- Random forest
- Support Vector Machine (I)

Classifiers

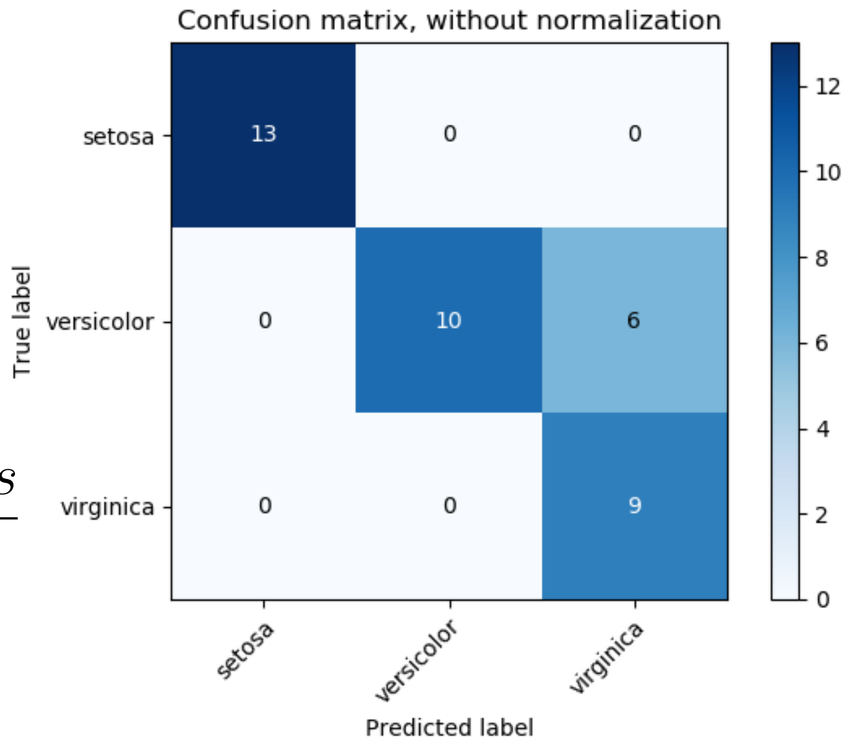
- Why do we need classifiers?
- What do we use to quantify the performance of a classifier?
- What is the baseline accuracy of a 5-class classifier using 0-1 loss function?
- What’s validation and cross-validation in classification?

Performance of a multiclass classifier

- Assuming there are **c** classes:
- The class confusion matrix is c × c
- Under the 0-1 loss function

  accuracy= $\dfrac{sum\ of\ diagonal\ terms}{sum\ of\ all\ terms}$

  ie. in the right example, accuracy = 32/38=84%
- The baseline accuracy is 1/c.



Source: scikit-learn

Cross-validation

- If we don't want to "waste" labeled data on validation, we can use **cross-validation** to see if our training method is sound.

- Split the labeled data into training and validation sets in multiple ways

- For each split (called a **fold**)
  - Train a classifier on the training set
  - Evaluate its accuracy on the validation set

- Average the accuracy to evaluate the training methodology

Q1. Cross-validation

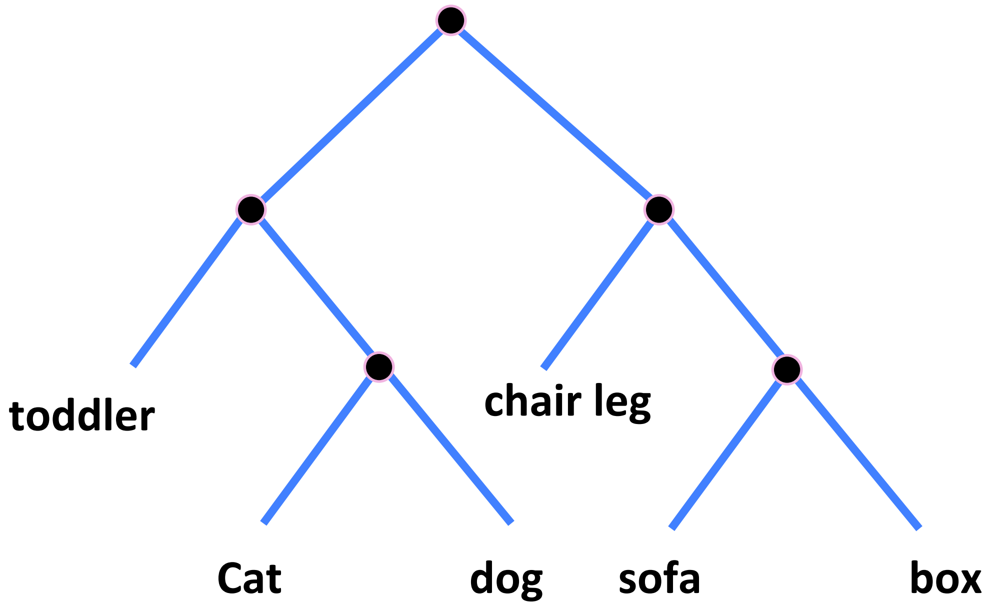
Cross-validation is a method used to prevent overfitting in classification.

A. TRUE

B. FALSE

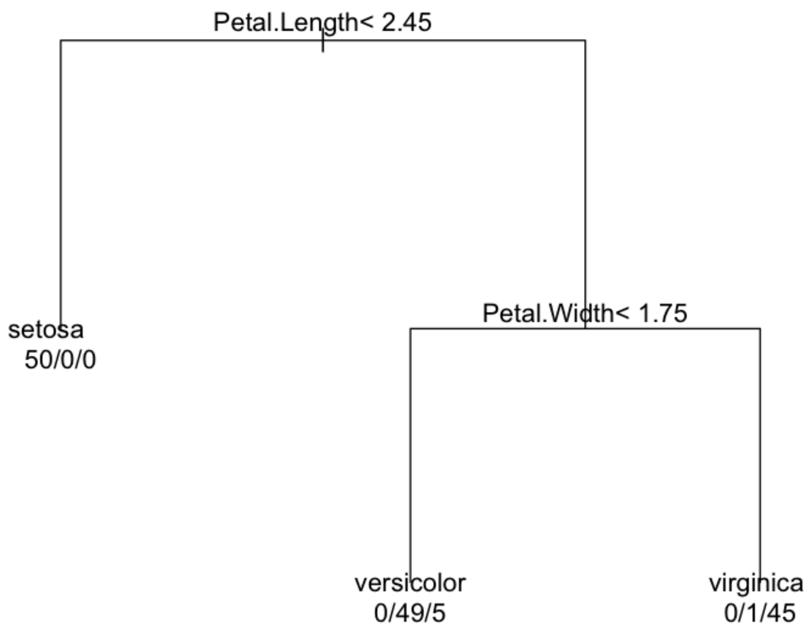
Decision tree: object classification

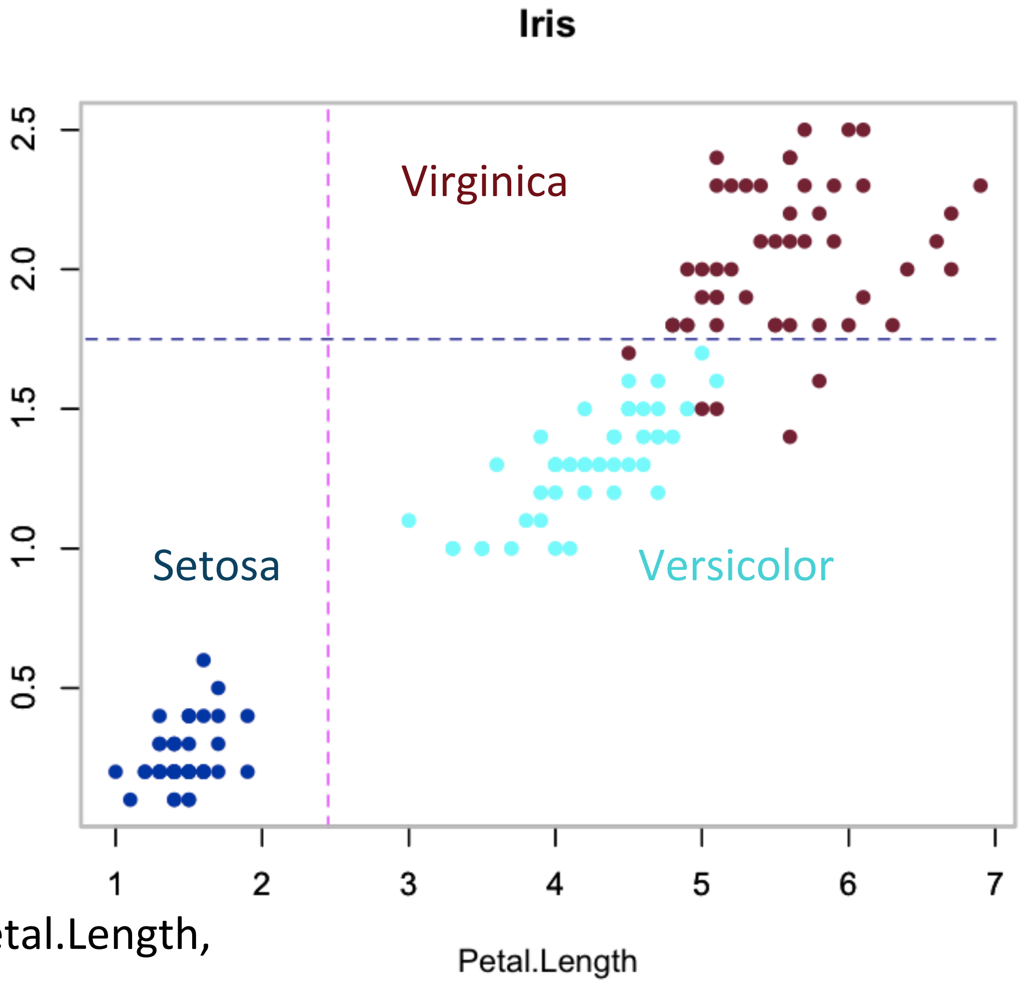
- The object classification **decision tree** can classify objects into multiple classes using sequence of simple tests. It will naturally grow into a tree.



Training a decision tree: example

- The “Iris” data set





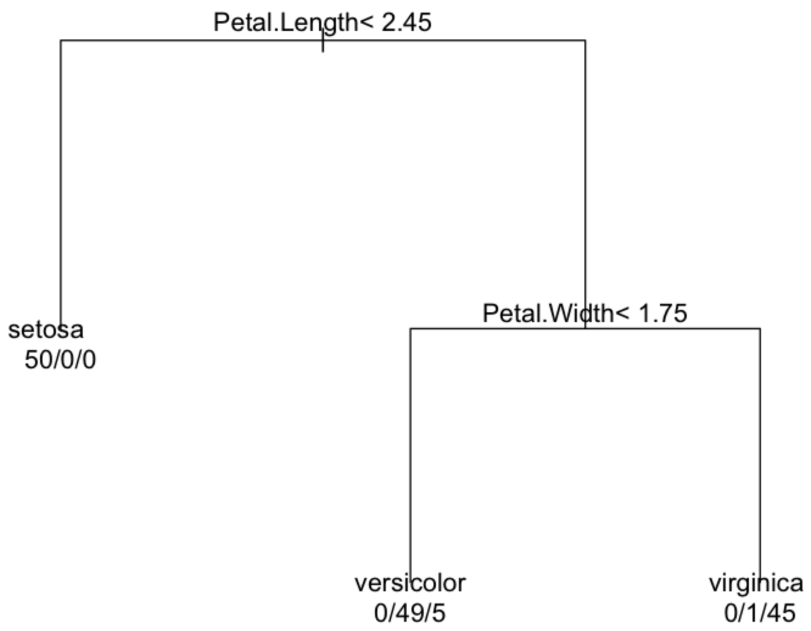
Features: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width **Label**: Species

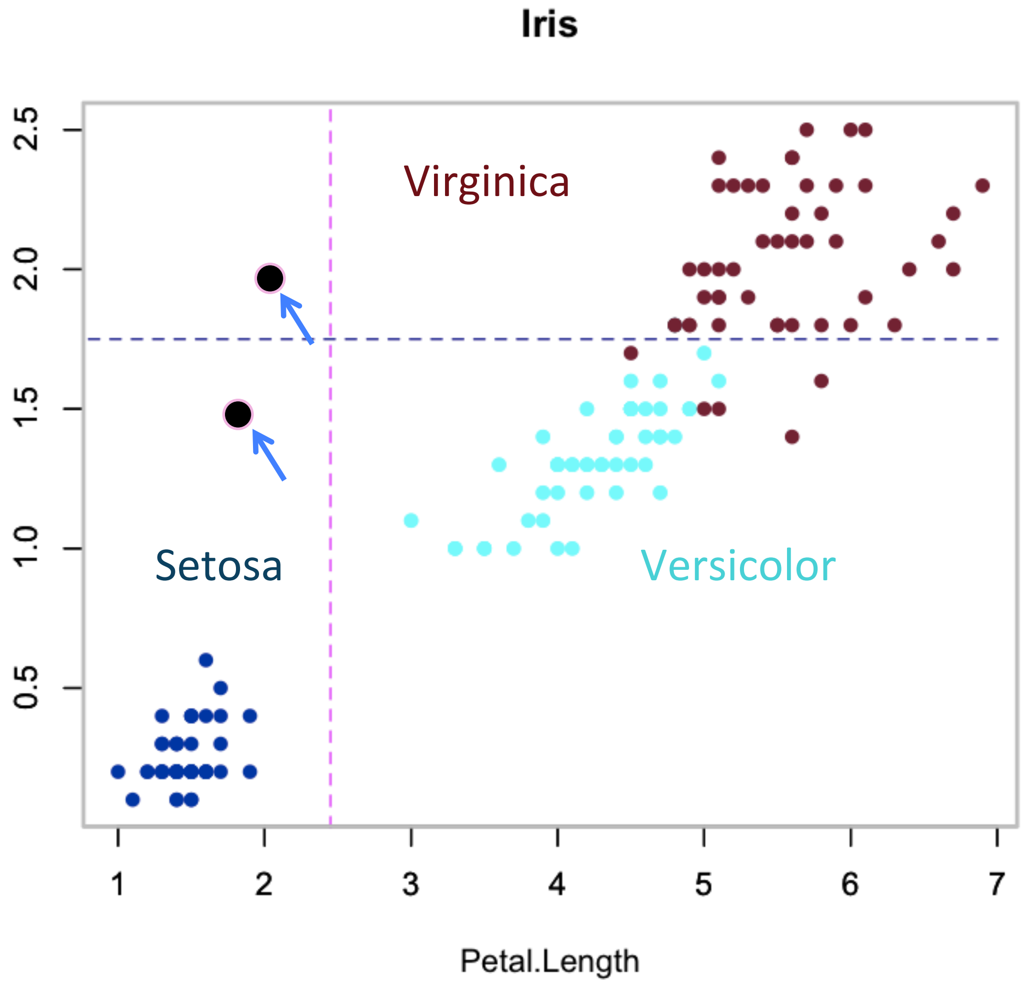
Training a decision tree

- Choose a dimension/feature and a split
- Split the training Data into left- and right-child subsets D_l and D_r
- Repeat the two steps above recursively on each child
- Stop the recursion based on some conditions
- Label the leaves with class labels

Classifying with a decision tree: example

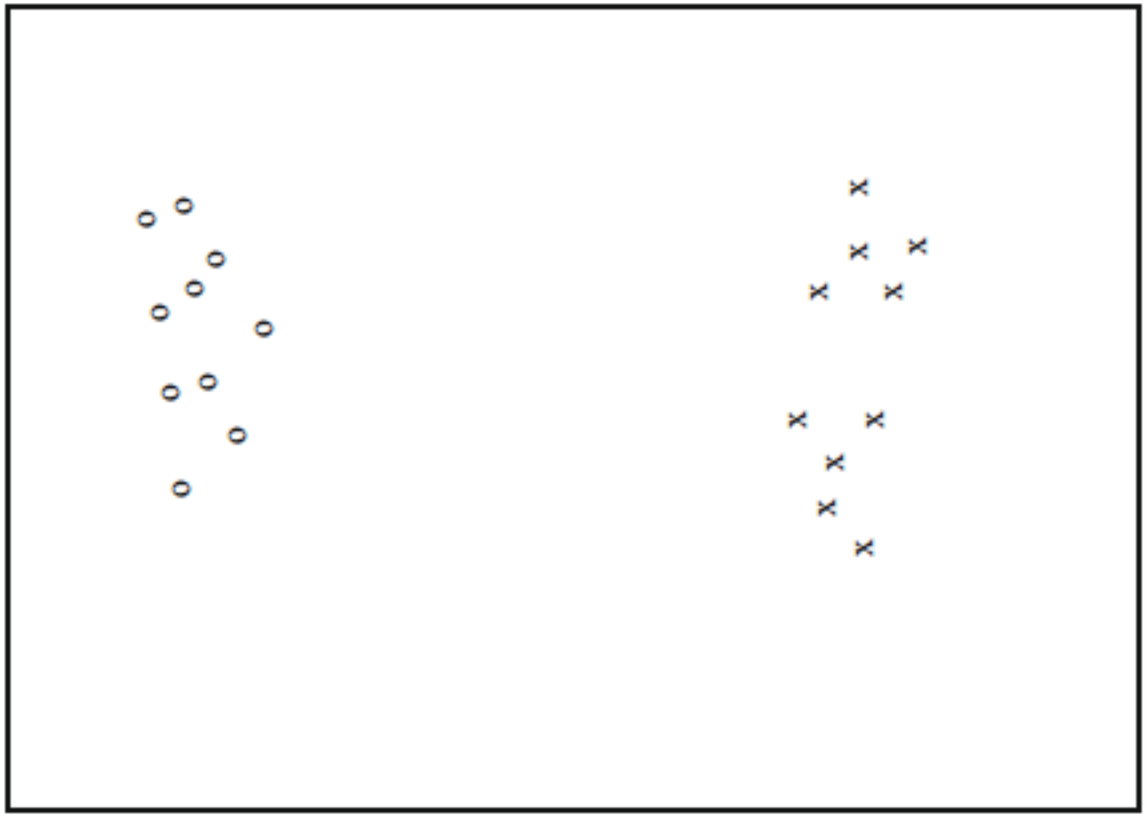
- The "Iris" data set





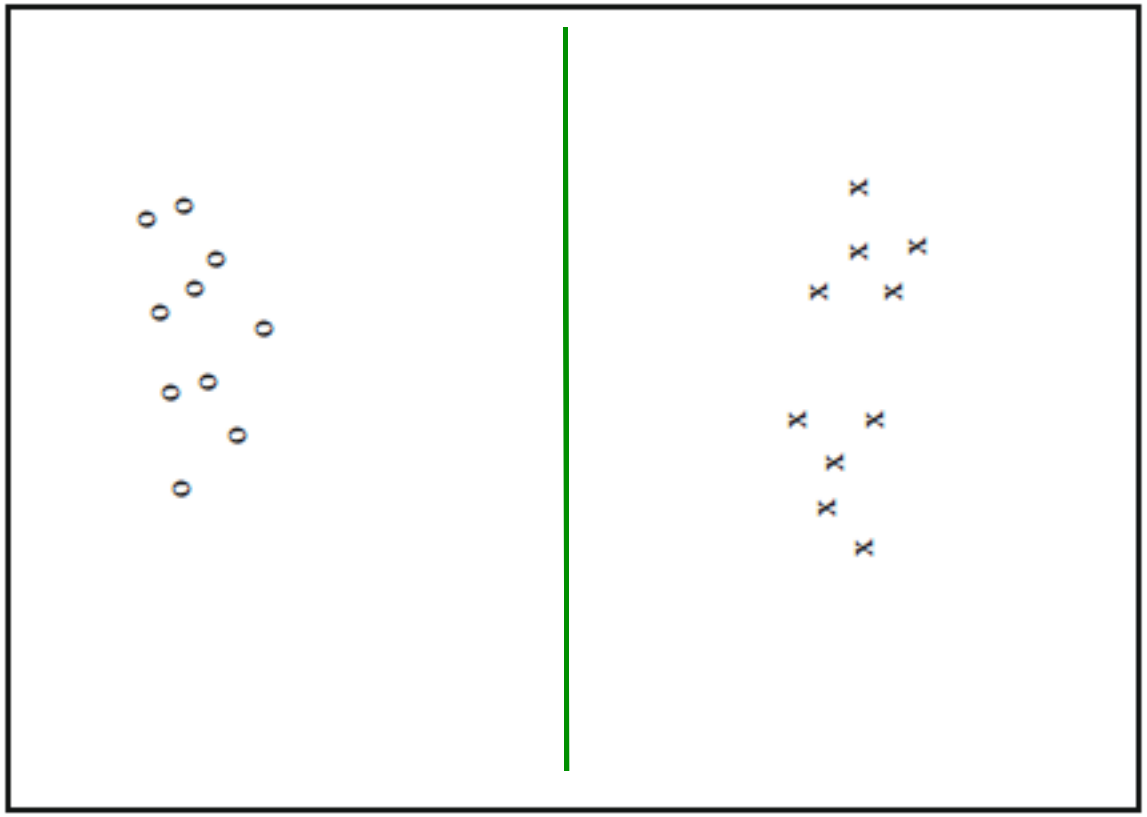
Choosing a split

- An informative split makes the subsets more concentrated and reduces uncertainty about class labels



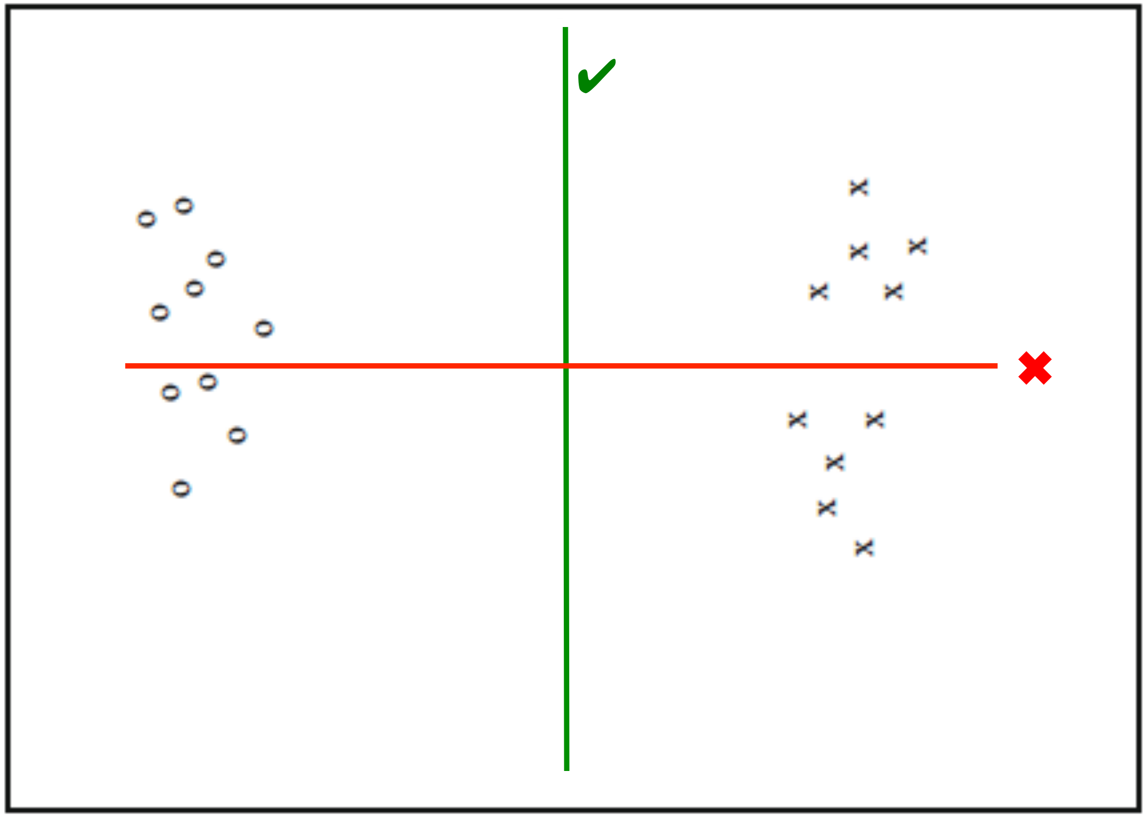
Choosing a split

- An informative split makes the subsets more concentrated and reduces uncertainty about class labels

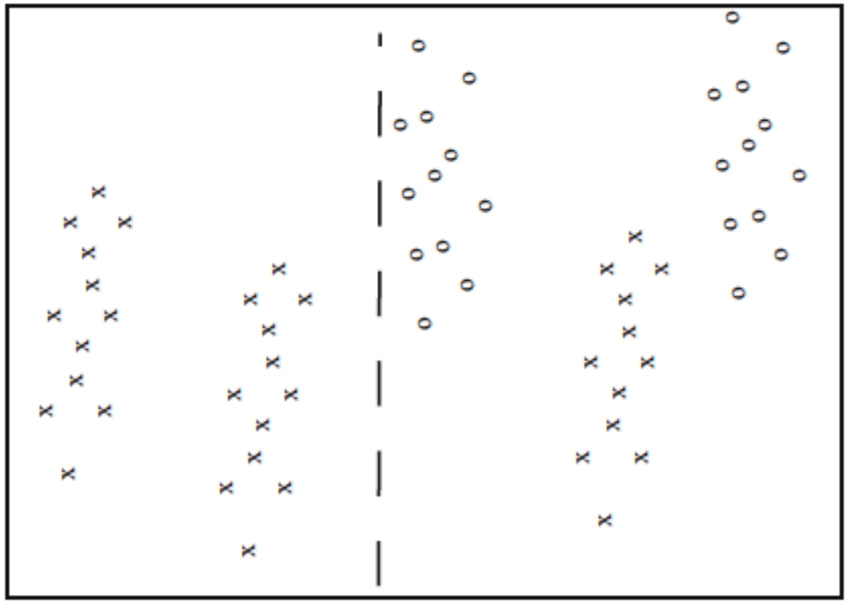


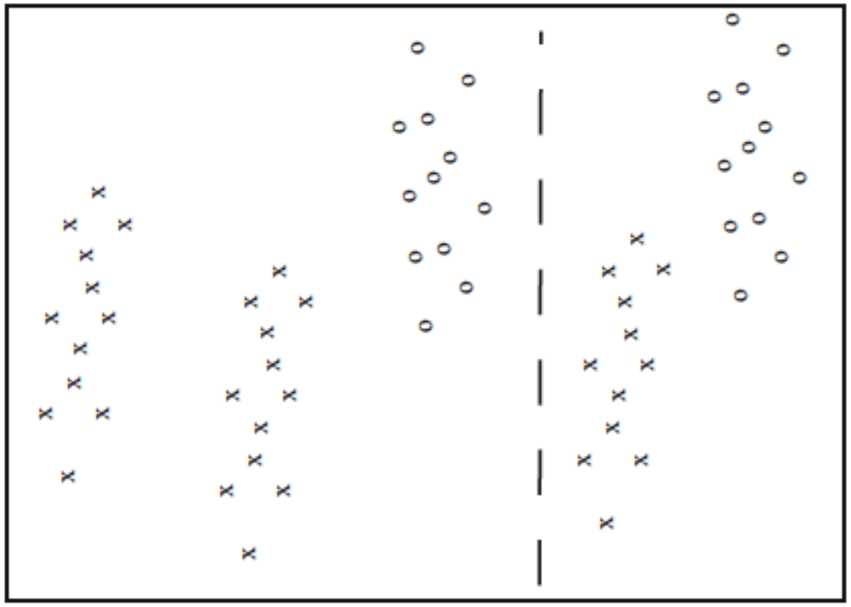
Choosing a split

- An informative split makes the subsets more concentrated and reduces uncertainty about class labels



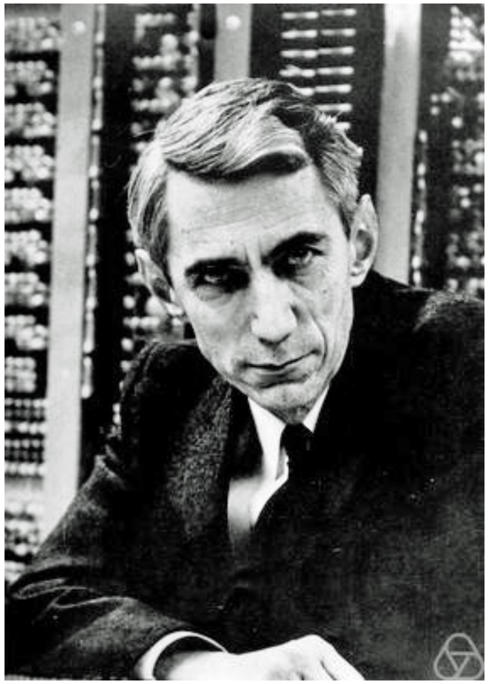
Which is more informative?





Quantifying uncertainty using entropy

- We can measure uncertainty as the number of bits of information needed to distinguish between classes in a dataset (first introduced by Claude Shannon)
  - We need $\log_2 2 = 1$ bit to distinguish 2 equal classes
  - We need $\log_2 4 = 2$ bit to distinguish 4 equal classes



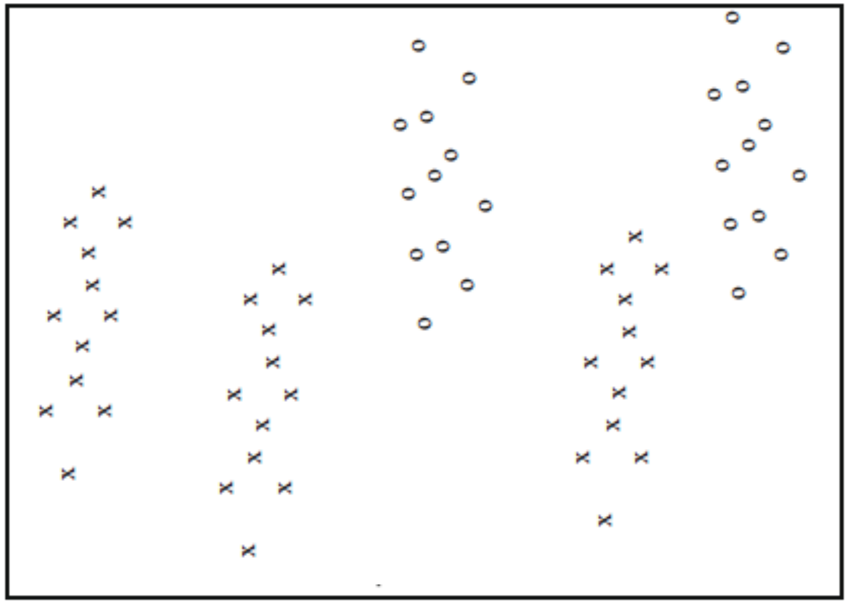
Claude Shannon (1916-2001)

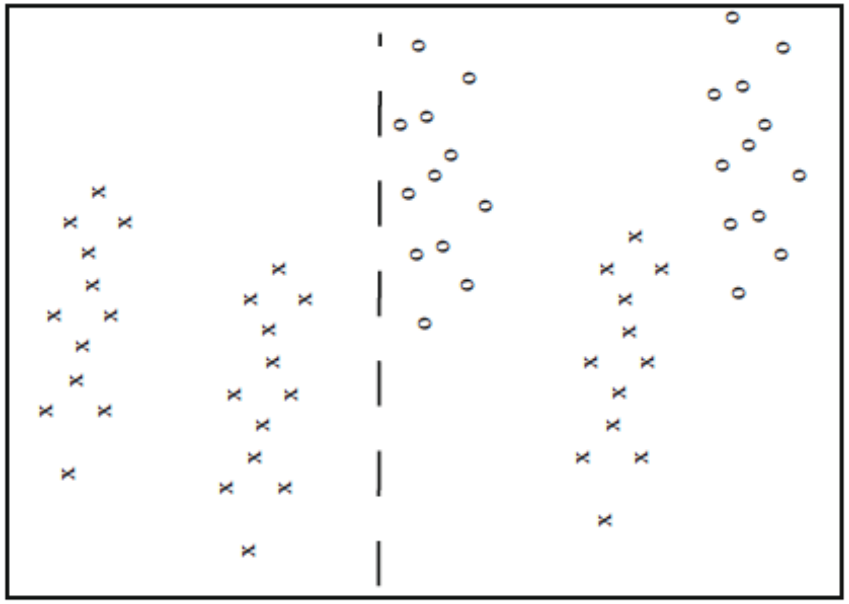
Quantifying uncertainty using entropy

- **Entropy** (Shannon entropy) is the measure of uncertainty for a general distribution
  - If class ***i*** contains a fraction ***P***(***i***) of the data, we need $log_2 \frac{1}{P(i)}$ bits for that class
  - The entropy H(D) of a dataset is defined as the **weighted mean** of entropy for every class:

$$H(D) = \sum_{i=1}^{c} P(i) log_2 \frac{1}{P(i)}$$

Entropy: before the split

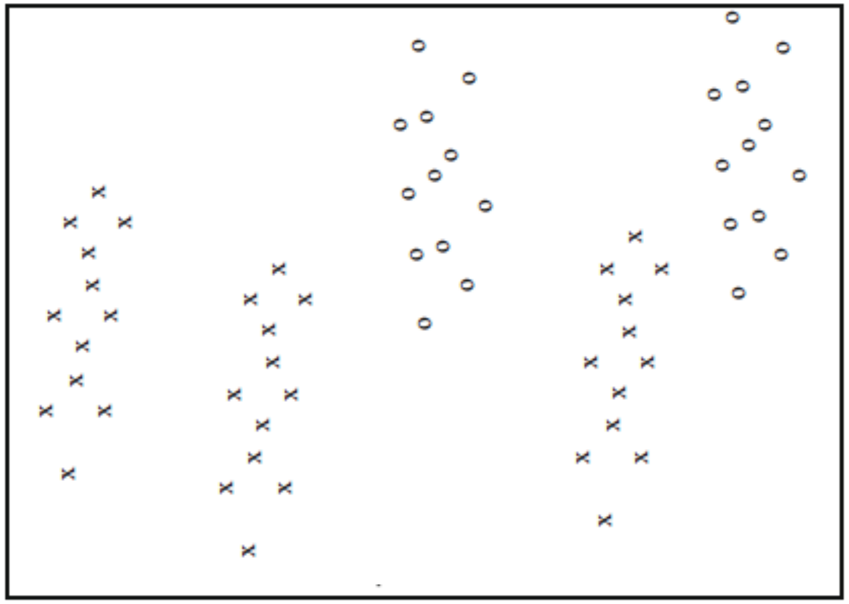


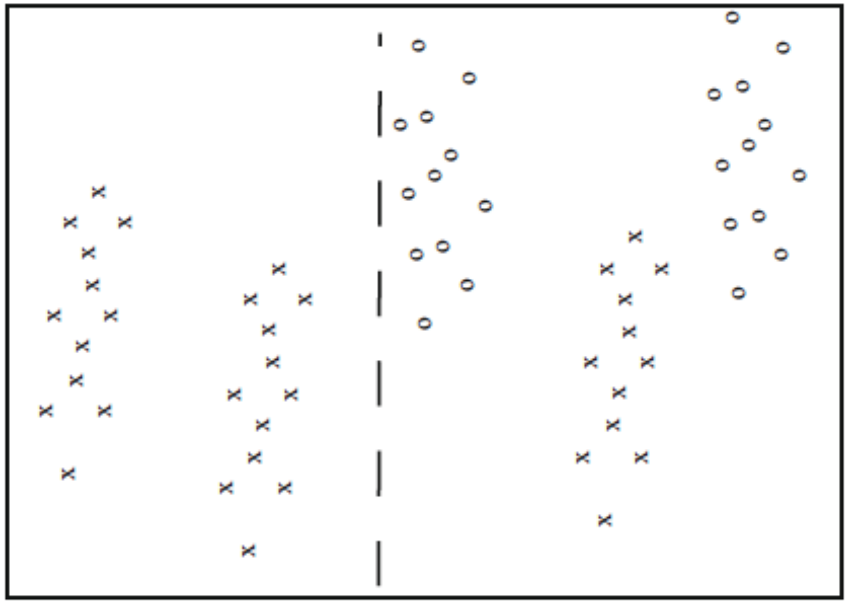


$$H(D) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}$$

$$= 0.971\ bits$$

Entropy: examples



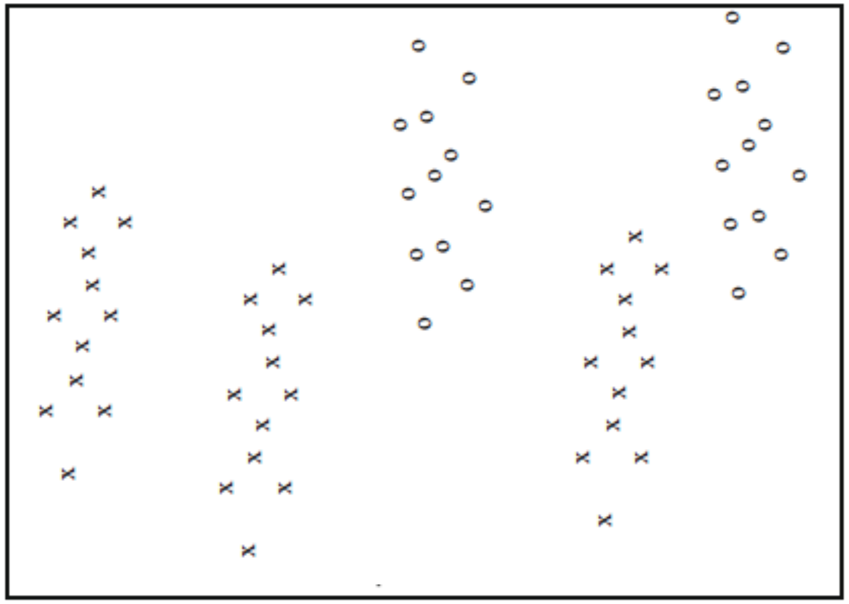


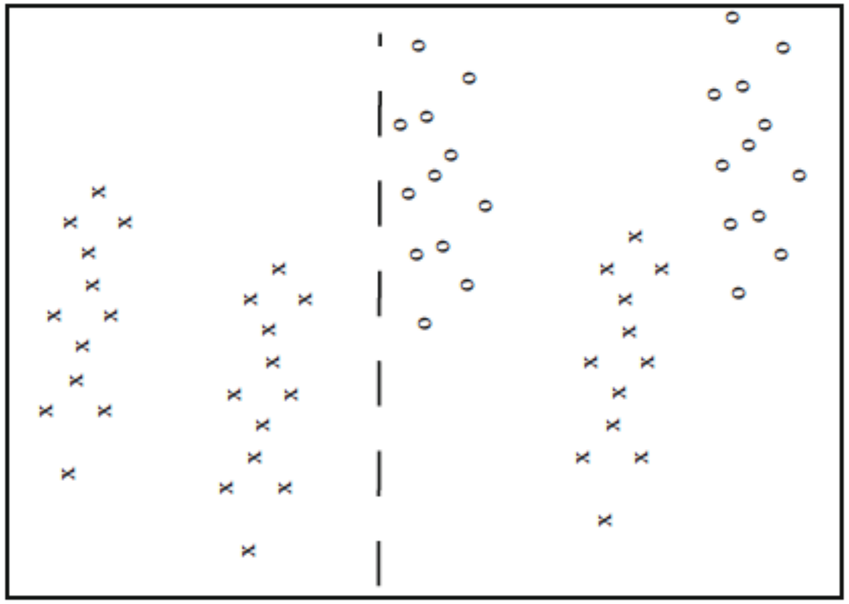
$$H(D) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}$$

$$= 0.971\ bits$$

$$H(D_l) = -1\ log_2 1 = 0\ bits$$

Entropy: examples





$$H(D) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}$$

$$= 0.971\ bits$$

$$H(D_l) = -1\ log_2 1 = 0\ bits$$

$$H(D_r) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}$$
$$= 0.918\ bits$$

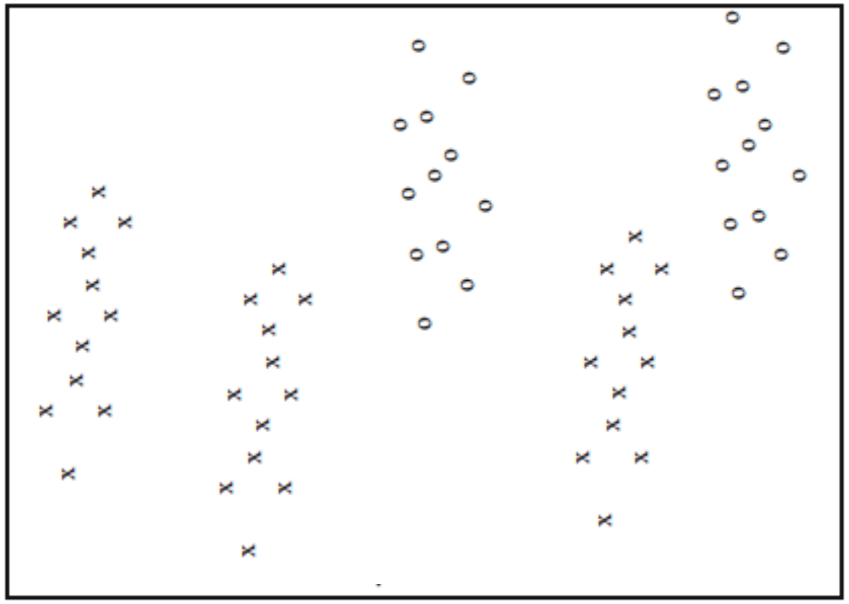
Information gain of a split

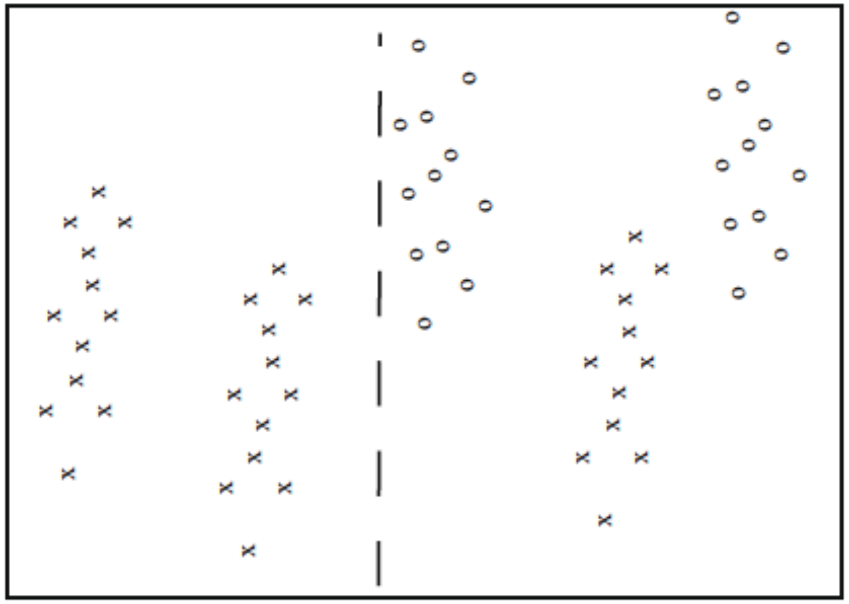
- The information gain of a split is the amount of entropy that was reduced on average after the split

$$I = H(D) - (\frac{N_{Dl}}{N_D} H(D_l) + \frac{N_{Dr}}{N_D} H(D_r))$$

- where
  - N_D is the number of items in the dataset D
  - N_{Dl} is the number of items in the left-child dataset D_l
  - N_{Dr} is the number of items in the left-child dataset D_r

Information gain: examples





$$I = H(D) - (\frac{N_{Dl}}{N_D} H(D_l) + \frac{N_{Dr}}{N_D} H(D_r))$$
$$= 0.971 - (\frac{24}{60} \times 0 + \frac{36}{60} \times 0.918)$$
$$= 0.420 \ bits$$

Q. Is the splitting method global optimum?

A. Yes
B. No

How to choose a dimension and split

- If there are **d** dimensions, choose approximately $\sqrt{d}$ of them as candidates at random
- For each candidate, find the split that maximizes the information gain
- Choose the best overall dimension and split
- Note that splitting can be generalized to categorical features for which there is no natural ordering of the data

When to stop growing the decision tree?

- Growing the tree too deep can lead to overfitting to the training data
- Stop recursion on a data subset if any of the following occurs:
  - All items in the data subset are in the same class
  - The data subset becomes smaller than a predetermined size
  - A predetermined maximum tree depth has been reached.

How to label the leaves of a decision tree

- A leaf will usually have a data subset containing many class labels
- Choose the class that has the most items in the subset
- Alternatively, label the leaf with the number it contains in each class for a probabilistic “soft” classification.

Pros and Cons of a decision tree

- Pros:
  - Easy to interpret.
  - Handles both discrete and continuous inputs
  - Insensitive to scaling
  - Fast running time
- Cons:
  - Accuracy is not great, due to the greedy algorithm
  - Tends to be unstable (high variance)

From decision trees to random forest

- Decision trees have some drawbacks
  - May not perform well on training data because its simplistic random training
  - May not perform well on test data because of overfitting
- A random forest is a randomly generated ensemble of decision trees that avoids both of the above problems by merging the classifications of the individual trees.

Training, evaluation and classification

- Build the random forest by training each decision tree on a random subset with replacement from the training data and subset of features are also randomly selected--- “Bagging”
- Evaluate the random forest by testing on its out-of-bag items
- Classify by merging the classifications of individual decision trees
    - By simple vote
    - Or by adding soft classifications together and then take a vote

An example of bagging

Drawing random samples from our training set with replacement. E.g., if our training set consists of 7 training samples, our bootstrap samples (here: n=7) can look as follows, where C_1, C_2, … C_m shall symbolize the decision tree classifiers.

Sample indices	Bagging Round 1	Bagging Round 2	…	Bagging Round M
1	2	7		
2	2	3		
3	1	2		
4	3	1		
5	4	1		
6	7	7		
7	2	1		
	$\downarrow$	$\downarrow$		
	C_1	C_2		

Pros and Cons of Random forest

- Pros:
  - Multiple trees are de-correlated, so ensemble prediction is more accurate
  - Less prone to overfitting
  - Fast running time
- Cons:
  - Difficult to interpret

Q2. Do you think random forest will always outperform simple decision tree?

A. Yes
B. No

Considerations in choosing a classifier

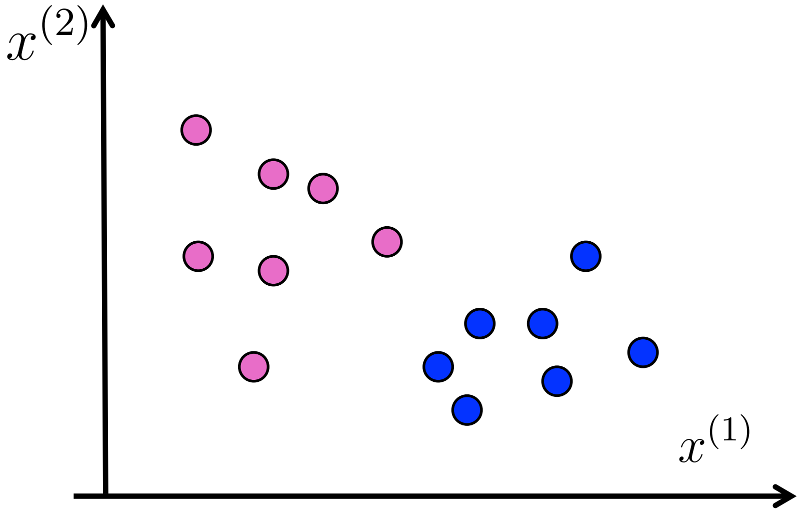
- When solving a classification problem, it is good to try several techniques.
- Criteria to consider in choosing the classifier include
  - Accuracy
  - Training speed
  - Classification speed
  - Performance with small training set
  - Interpretability

Support Vector Machine (SVM) overview

- The Decision boundary and function of a Support Vector Machine
- Loss function (cost function in the book)
- Training
- Validation
- Extension to multiclass classification

SVM problem formulation

- At first we assume a binary classification problem
- The training set consists of N items
  - Feature vectors x_i of dimension d
  - Corresponding class labels $y_i \in \{\pm 1\}$
- We can picture the training data as a d-dimensional scatter plot with colored labels



Decision boundary of SVM

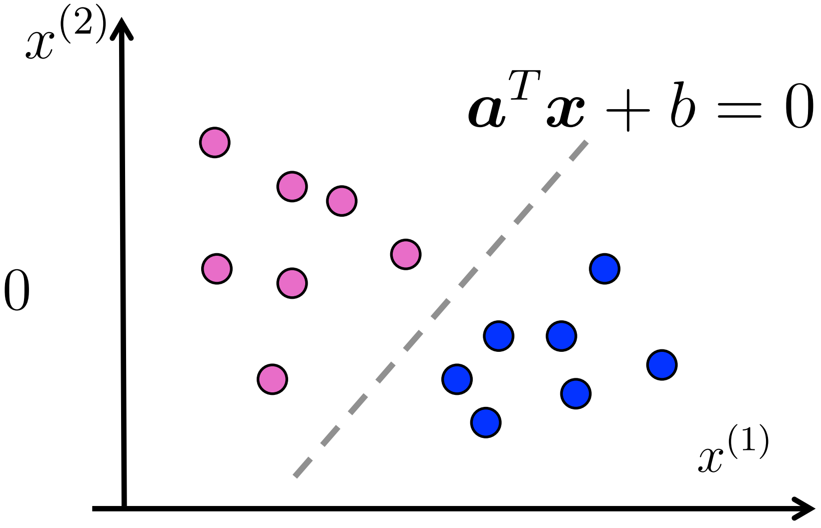
- SVM uses a hyperplane as its **decision boundary**

- The decision boundary is:

$$a_1 x^{(1)} + a_2 x^{(2)} + \ldots + a_d x^{(d)} + b = 0$$

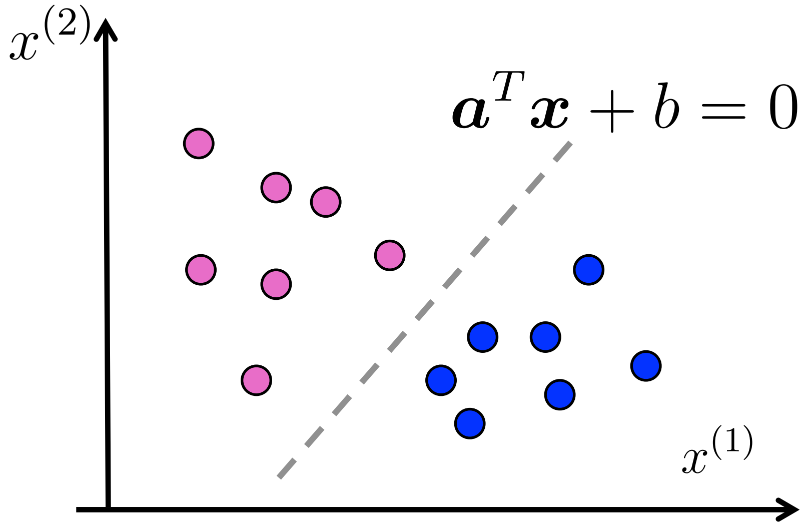
- In vector notation, the hyperplane can be written as:

$$\boldsymbol{a}^T \boldsymbol{x} + b = 0$$



Q3. How many solutions can we have for the decision boundary?

A. One
B. Several
C. Infinite

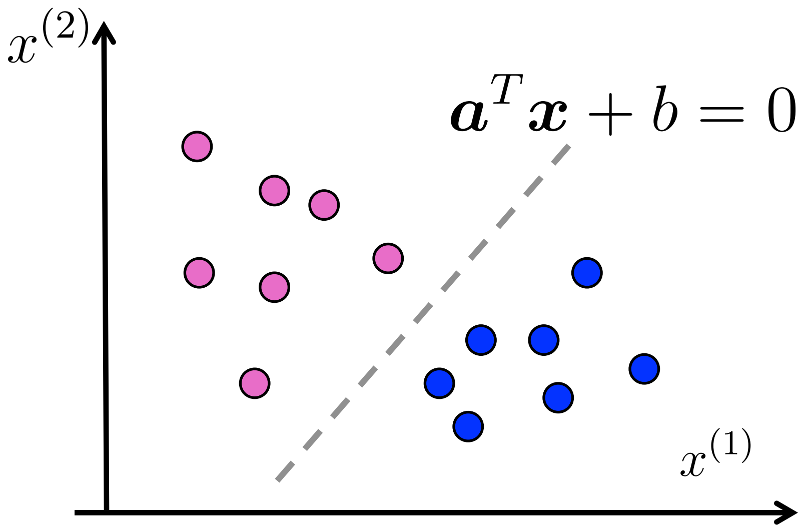


Classification function of SVM

- SVM assigns a class label to a feature vector according to the following rule:

  +1 if $\boldsymbol{a}^T \boldsymbol{x}_i + b \geq 0$
  -1 if $\boldsymbol{a}^T \boldsymbol{x}_i + b < 0$

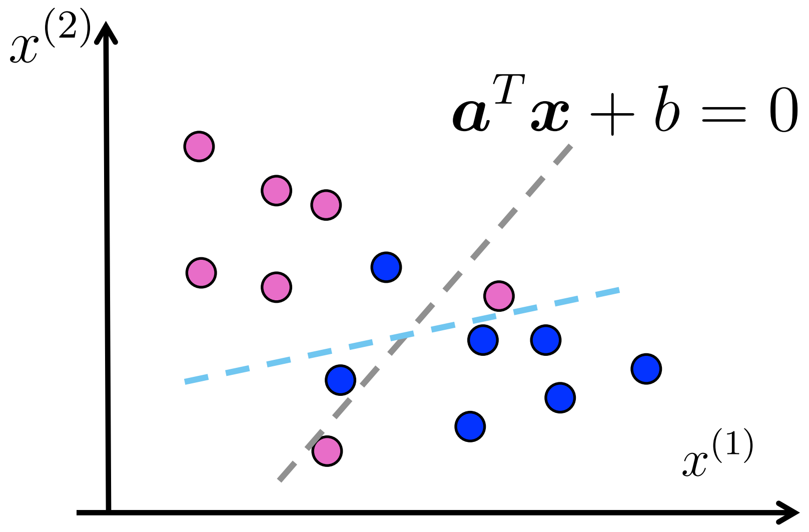
- In other words, the classification function is: $sign(\boldsymbol{a}^T \boldsymbol{x}_i + b)$



- Note that
  - If $|\boldsymbol{a}^T \boldsymbol{x}_i + b|$ is small, then $\boldsymbol{x}_i$ was close to the decision boundary
  - If $|\boldsymbol{a}^T \boldsymbol{x}_i + b|$ is large, then $\boldsymbol{x}_i$ was far from the decision boundary

What if there is no clean cut boundary?

- Some boundaries are better than others for the training data
- Some boundaries are likely more robust for run-time data
- We need to a quantitative measure to decide about the boundary
- The **loss function** can help decide if one boundary is better than others

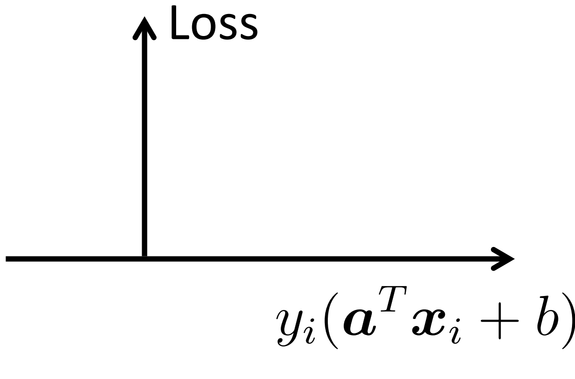


Loss function 1

- For any given feature vector $\boldsymbol{x}_i$ with class label $y_i \in \{\pm 1\}$, we want
  - Zero loss if $\boldsymbol{x}_i$ is classified correctly $sign(\boldsymbol{a}^T\boldsymbol{x}_i + b) = y_i$
  - Positive loss if $\boldsymbol{x}_i$ is misclassified $sign(\boldsymbol{a}^T\boldsymbol{x}_i + b) \neq y_i$
  - If $\boldsymbol{x}_i$ is misclassified, more loss is assigned if it's further away from the boundary

- This loss function 1 meets the criteria above:

$$max(0, -y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$



- Training error cost

$$S(\boldsymbol{a}, b) = \frac{1}{N}\sum_{i=1}^{N} max(0, -y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$

Q4. What's the value of this function ?

$$max(0, -y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b)) \quad \text{if} \quad sign(\boldsymbol{a}^T\boldsymbol{x}_i + b) = y_i$$

A. 0.
B. others.

Q5. What's the value of this function ?

$$max(0, -y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b)) \quad \text{if} \quad sign(\boldsymbol{a}^T\boldsymbol{x}_i + b) \neq y_i$$

A. 0.
B. A value greater than or equal to 0.

The problem with loss function 1

- Loss function1 does not distinguish between the following decision boundaries if they both classify $\boldsymbol{x}_i$ correctly.
  - One passes the two classes closely
  - One that passes with a wider margin

- But leaving a larger margin gives robustness for run-time data- **the large margin principle**

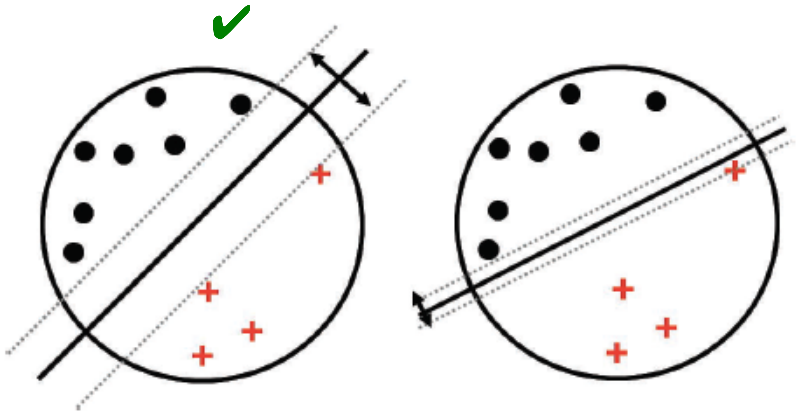


Figure 14.11 Illustration of the large margin principle. Left: a separating hyper-plane with large margin. Right: a separating hyper-plane with small margin.

Credit: Kelvin Murphy

Q6. Wondering what does "support vector" mean?

A. Yes.
B. No.

Support vectors are those data points in the training data that uniquely define the decision boundary

Q7. SVM classification is faster than decision tree in terms of time complexity

A. TRUE.
B. FALSE.

Loss function 2: the hinge loss

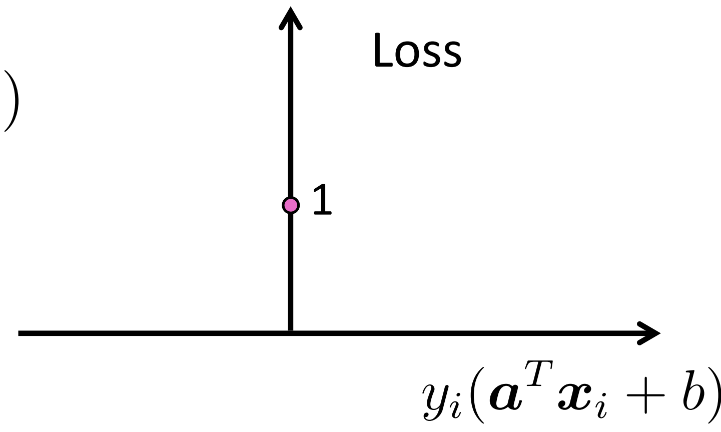
- We want to impose a small positive loss if $\boldsymbol{x}_i$ is correctly classified but close to the boundary

- The **hinge loss** function meets the criteria above:

$$max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$

- Training error cost

$$S(\boldsymbol{a}, b) = \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$



Loss function 2: the hinge loss

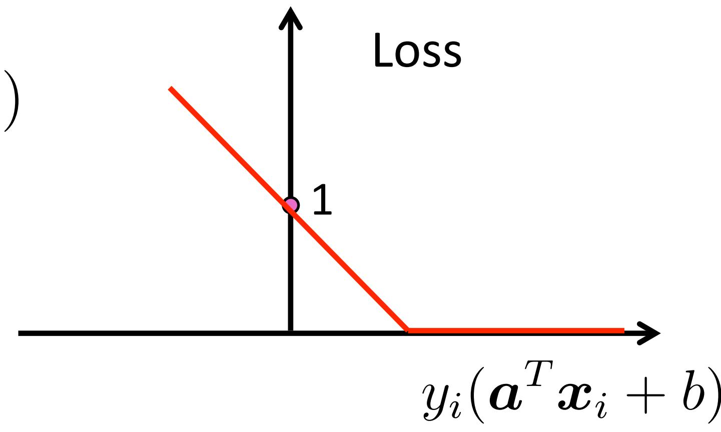
- We want to impose a small positive loss if $\boldsymbol{x}_i$ is correctly classified but close to the boundary

- The **hinge loss** function meets the criteria above:

$$max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$

- Training error cost

$$S(\boldsymbol{a}, b) = \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$



The problem with loss function 2

- Loss function 2 favors decision boundaries that have large $\|\boldsymbol{a}\|$ because increasing $\|\boldsymbol{a}\|$ can zero out the loss for a correctly classified $\boldsymbol{x}_i$ near the boundary.

- But large $\|\boldsymbol{a}\|$ makes the classification function $sign(\boldsymbol{a}^T\boldsymbol{x}_i + b)$ extremely sensitive to small changes in $\boldsymbol{x}_i$ and make it less robust to run-time data.

- So small $\|\boldsymbol{a}\|$ is better.

Assignments

- Read Chapter 11 of the textbook
- Next time: SVM-regularization, Stochastic descent

Additional References

- Robert V. Hogg, Elliot A. Tanis and Dale L. Zimmerman. “Probability and Statistical Inference”
- Morris H. Degroot and Mark J. Schervish "Probability and Statistics”
- Kelvin Murphy, “Machine learning, A Probabilistic perspective”

See you next time

See You!