ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Lesson 2: Machine Learning

Content

1. Basic concepts

- The data is described by the attributes in set $A = \{A_1, A_2, ..., A_{|A|}\}$
- Class attribute $C = \{c_1, c_2, ..., c_{|C|}\}$ ($|C| \geq 2$), c_i is a class label. Each learning dataset includes examples containing information about "experience".
- Given a dataset D, the goal of learning is to build a classifier function /predictor associates attribute values in A with classes in C.
- Functions can be used to classify/predict unseen data
- The function is also known as a classifier/prediction model or classifier

Table 1

ID	Age	Have job	Have home	Credit	Class
1	Young	FALSE	FALSE	Normal	No
2	Young	FALSE	FALSE	Good	No
3	Young	TRUE	FALSE	Normal	Yes
4	Young	TRUE	TRUE	Normal	Yes
5	Young	FALSE	FALSE	Normal	No
6	Middle-age	FALSE	FALSE	Normal	No
7	Middle-age	FALSE	FALSE	Good	No
8	Middle-age	TRUE	TRUE	Good	Yes
9	Middle-age	FALSE	TRUE	Excellent	Yes
10	Middle-age	FALSE	TRUE	Excellent	Yes
11	Old	FALSE	TRUE	Excellent	Yes
12	Old	FALSE	TRUE	Good	Yes
13	Old	TRUE	FALSE	Good	Yes
14	Old	TRUE	FALSE	Excellent	Yes
15	Old	FALSE	FALSE	Normal	No

- Supervised Learning: Class labels are provided in the dataset
- The data used for learning is called training data
- After the model is learned through a learning algorithm, it is evaluated on a test dataset to measure its accuracy.
- Do not use test data to learn the model.
- The labeled data set is usually divided into two independent sets for training and testing.

$$\text{Accuracy} = \frac{\text{Number of correctly classified samples}}{\text{Size of test dataset}}$$

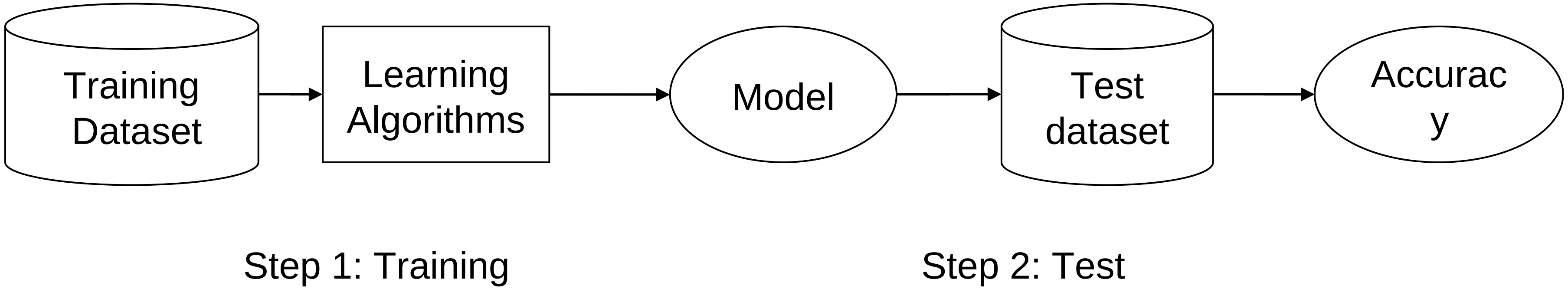
What is machine learning?

- Given a dataset representing past "experience", a task T and a performance metric M. A computer system is capable of learning from the data to perform task T if after learning performance of the machine on task T (measured by M) is improved.
- The learned model or knowledge helps the system perform the task better than no learning at all.
- The learning process is the process of building models or knowledge distillation.

- In **Table1**, if there is no learning process, assume that the test dataset has the same class distribution as the training data.
  - Make random predictions → *Accuracy* = 50%
  - Make predictions according to the most popular class (class *Yes*) → *Accuracy* = 9/15 = 60%
- The model is capable of learning if the accuracy is improved

Relationship between training and test dataset

- Assumption: The distribution of training dataset and test dataset is the same.



2. Evaluation methods

2.1 Evaluation methods

- Split the data into two independent training and test sets (usually using a 50-50 or 70-30 ratio)
  - Random sampling to generate training set; the rest as test set
  - If data is built over time, use past data as training data
- If dataset is small, perform sampling and evaluation n times then average
- Cross-validation: data is divided into n equal independent parts. Each time, one part is used as test data and n-1 remainder as training data. The results are averaged.
- Leave-one-out: If the data is too small, each set contains only 1 element, number of parts = number of elements in the data set.
- Validation set: Used to select the model's hyperparameters (parameters not learned)

2.2 Evaluation Metrics

Confusion matrix

	Predicted Positive	Predicted Negative
Actually Positive	TP	FN
Actually Negative	FP	TN

TP: actual positive, predicted positive (true positive)
TN: actual negative, predicted negative (true negative)
FP: actual negative, predicted positive (false positive)
FN: actual positive, predicted negative (false negative)

A positive example is an example with a class label of interest
A negative example is an example with a class label of disinterest

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \qquad F = \frac{2pr}{p + r}$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
+	+	+	-	+	-	+	-	+	+	-	-	+	-	-	-	+	-	-	+

Assume: 10 positive texts

Rank 1: p = 1/1 = 100% r = 1/10 = 10%

Rank 2: p = 2/2 = 100% r = 2/10 = 20%

…

Rank 9: p = 6/9 = 66.7% r = 6/10 = 60%

Break-even point

Rank 10: p = 7/10 = 70% r = 7/10 = 70%

$$TPR = \frac{TP}{TP + FN}$$

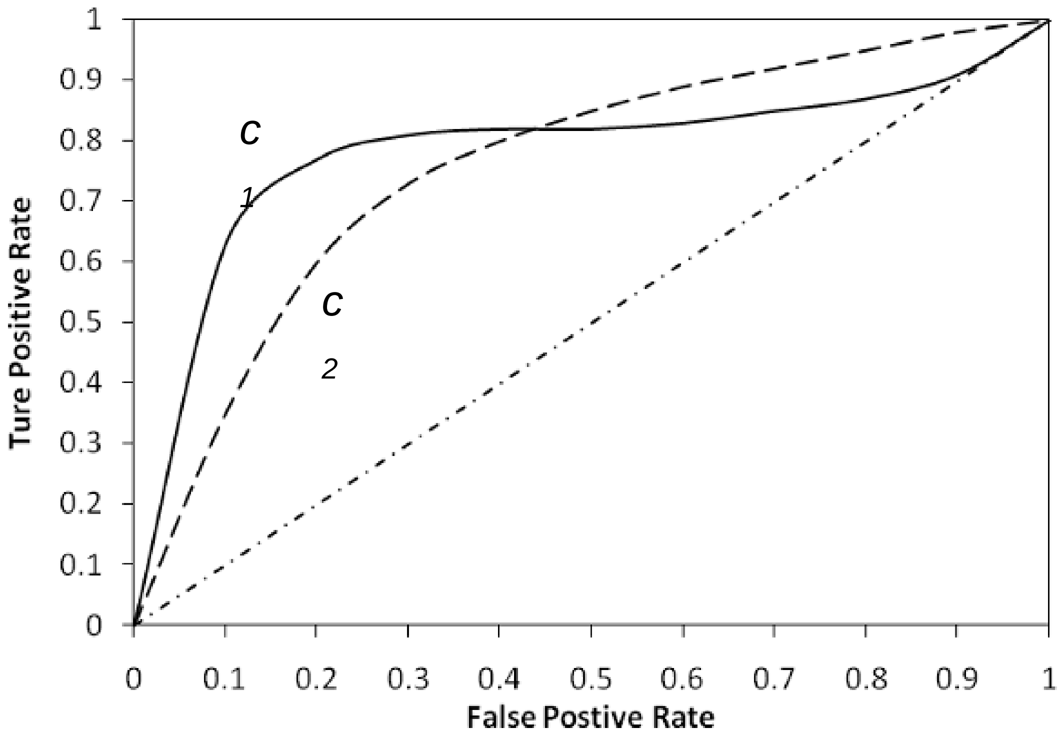
(sensitivity)

$$FPR = \frac{FP}{TN + FP}$$

= (1- specificity)

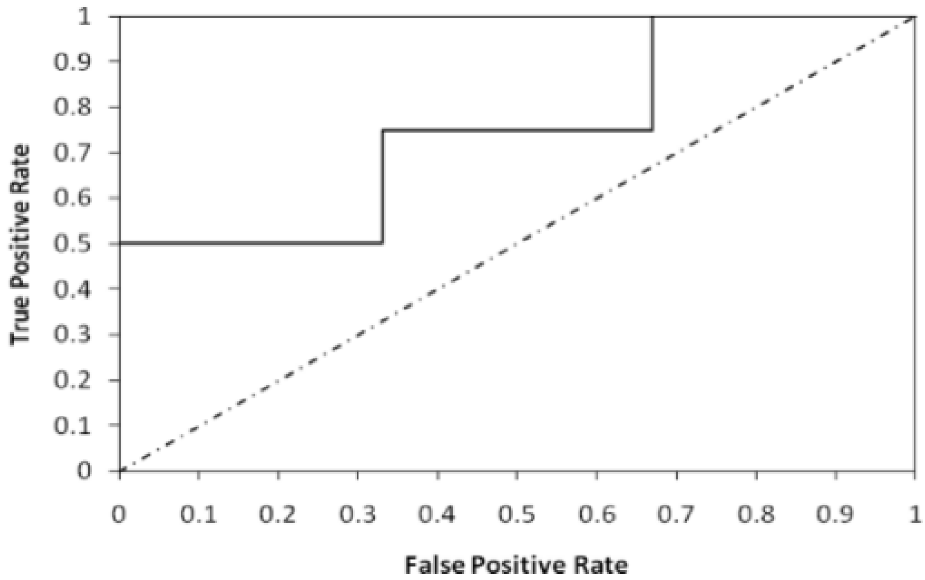
$$TNR = \frac{TN}{TN + FP}$$

(specificity)

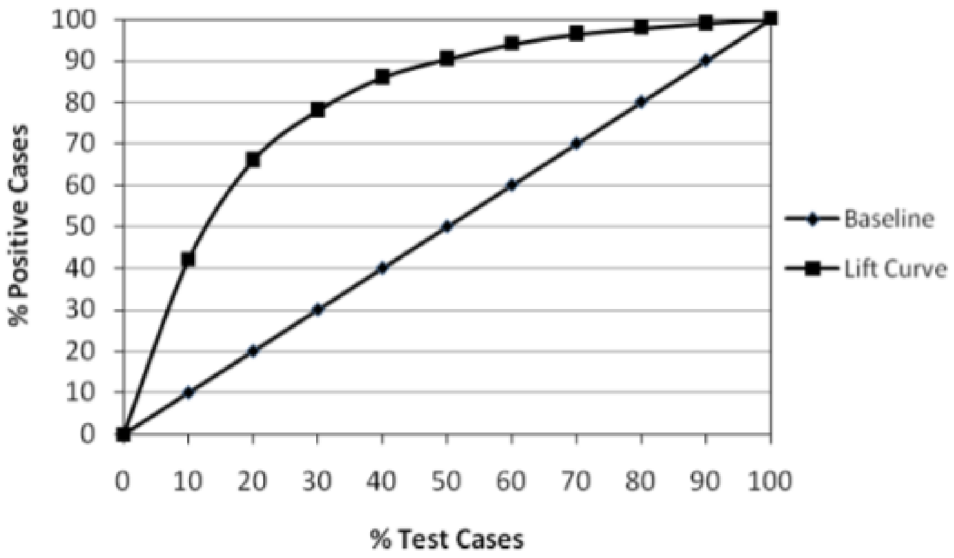


ROC curve of two classifiers c1 and c2 on the same set of dataset

Rank		1	2	3	4	5	6	7	8	9	10
Actual class		+	+	–	–	+	–	–	+	–	–
TP	0	1	2	2	2	3	3	3	4	4	4
FP	0	0	0	1	2	2	3	4	4	5	6
TN	6	6	6	5	4	4	3	2	2	1	0
FN	4	3	2	2	2	1	1	1	0	0	0
TPR	0	0.25	0.5	0.5	0.5	0.75	0.75	0.75	1	1	1
FPR	0	0	0	0.17	0.33	0.33	0.50	0.67	0.67	0.83	1



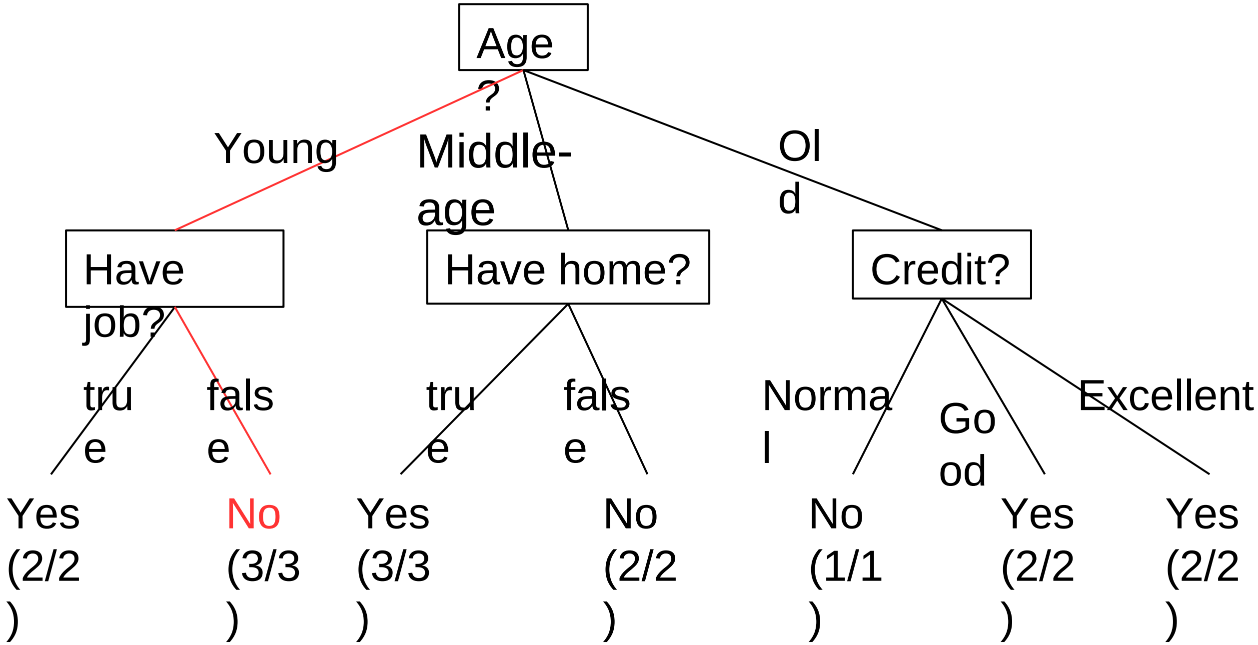
Bin	1	2	3	4	5	6	7	8	9	10
# of test cases	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
# of positive cases	210	120	60	40	22	18	12	7	6	5
% of positive cases	42.0%	24.0%	12%	8%	4.4%	3.6%	2.4%	1.4%	1.2%	1.0%
% cumulative	42.0%	66.0%	78.0%	86.0%	90.4%	94.0%	96.4%	97.8%	99.0%	100.0%



The corresponding lift curve of the data in the table

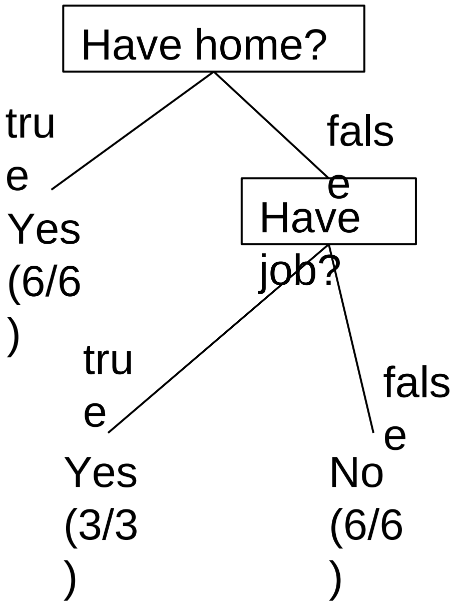
3. Decision Tree

- Decision node: leaf node.
- For prediction, traverse the tree from the root by the values of the attributes until a leaf node is encountered.



Age	Have job	Have home	Credit	Class
Young	FALSE	FALSE	GOOD	?

- Decision trees are built by dividing data into homogeneous subsets. The subset is said to be homogeneous if the examples have the same class.
- Small trees are generally more general and more precise; easier to understand for humans.
- The tree is not the only one born.
- Finding the best tree is an NP-complete problem → using heuristic algorithms.



Have home= true → Class =Yes [sup=6/15, conf=6/6]
Have home = false, Have job = true → Class = Yes [sup=3/15, conf=3/3]
Have home = false, Have job = false → Class = No [sup=6/15, conf=6/6].
Age = Young, Have job = false → Class = No [sup=3/15, conf=3/3]

3.1 Learning algorithm

- Use divide-and-conquer to divide data recursively
- Stop condition: all examples have the same class or all attributes are already used (line 1-4)
- At each recursive step, choose the best attribute to split the data by the attribute's value based on the heterogeneity function (line 7-11)
- Greedy Algorithm

Algorithm decisionTree(D, A , T)

if D contains only the training example of class $c_j \in C$ **then**

create leaf node T with class label c_j;

elseif $A = \emptyset$ **then**

create leaf node T with class label c_j being the most common class in D

else // D contains examples with multiple classes. Select an attribute

// to split D into subsets so that each subset is more homogeneous.

p_0 = impurityEval-1(D);

for each attribute $A_i \in A$ (=$\{A_1, A_2, ..., A_k\}$) **do**

p_i = impurityEval-2(A_i, D)

endfor

Chose $A_g \in \{A_1, A_2, ..., A_k\}$ minimize heterogeneity by $p_0 - p_i$;

if $p_0 - p_g < threshold$ **then** // A_g does not significantly reduce heterogeneity p_0

create leaf node T with class label c_j being the most common class in D

else // A_g reduces heterogeneity p_0

Create a decision node T according to A_g;

Let the values of A_g be $v_1, v_2, ..., v_m$.Split D into m non-intersecting subsets $D_1, D_2, ..., D_m$ based on m values of A_g.

for each D_j in $\{D_1, D_2, ..., D_m\}$ **do**

if $D_j \neq \emptyset$ **then**

create a branch (edge) corresponding to node T_j for v_j that is a child of T;

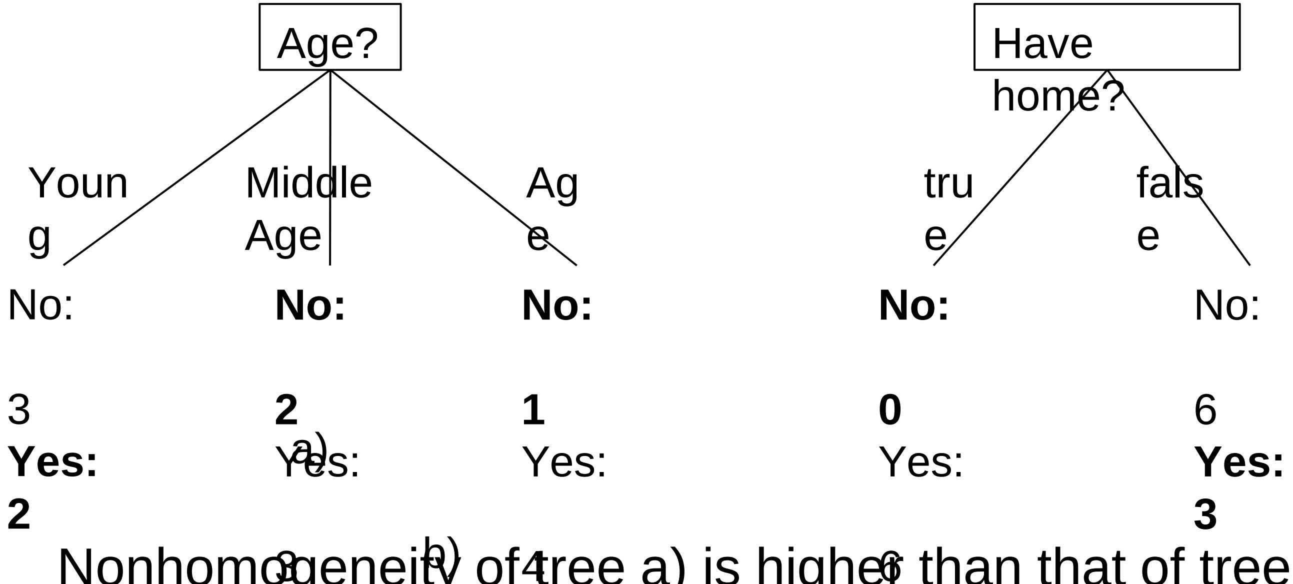
decisionTree{D_j, $A - \{A_g\}$, T_j) // delete A_g

endif

endfor

endif

3.2 Nonhomogenous function



Nonhomogeneity of tree a) is higher than that of tree b)

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j)\log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1$$

- $\Pr(c_j)$ is the probability that the data belongs to class c_j
- The unit of entropy is bit
- Convention $0\log_2 0 = 0$
- The more homogeneous the data, the smaller the entropy and vice versa.

Example 6:

Dataset D has two class positive (pos) và tiêu cực (neg)

1. Pr(pos) = Pr(neg) = 0.5 → entropy(D) = $-0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$
2. Pr(pos) = 0.2, Pr(neg) = 0.8 → entropy(D) = $-0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$
3. Pr(pos) = 1, Pr(neg) = 0 → entropy(D) = $-1 \times \log_2 1 - 0 \times \log_2 0 = 0$

Information gain

1. Calculate entropy(D) (line 7)
2. Attribute selection: For each attribute A_i, assuming there are v values, split D into non-intersecting subsets D_1, D_2, …, D_v (line 9)

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

3. Information gain A_i

$$\text{gain}(D, A_i) = \text{entropy}(D) - \text{entropy}_{A_i}(D)$$

$entropy(D) = -6/15 \times \log_2 6/15 – 9/15 \log_2 9/15 = 0.971$

$entropy_{Age}(D) = 5/15 \times entropy(D_{Age=Young}) + 5/15 \times entropy(D_{Age=Middle\text{-}Age})$

$+ 5/15 \times entropy(D_{Age=Old})$

$= 5/15 \times 0.971 + 5/15 \times 0.971 + 5/15 \times 0.722 = 0.888$

$Entropy_{have\ home}(D) = 6/15 \times entropy(D_{have\ home=true})$

$+ 9/15 \times entropy(D_{have\ home=false})$

$= 6/15 \times 0 + 9/15 \times 0.918 = 0.551$

$Entropy_{have\ job}(D) = 0.647$

$entropy_{credit}(D) = 0.608$

gain(D, Age) = 0.971 - 0.888 = 0.083

gain(D, Have home) = 0.971 - 0.551 = 0.420

gain(D, Have job) = 0.971 - 0.647 = 0.324

gain(D, Credit) = 0.971 - 0.608 = 0.363.

Information gain ratio

- Information gain often favors attributes with multiple values
- Gain ratio normalized entropy over attribute values

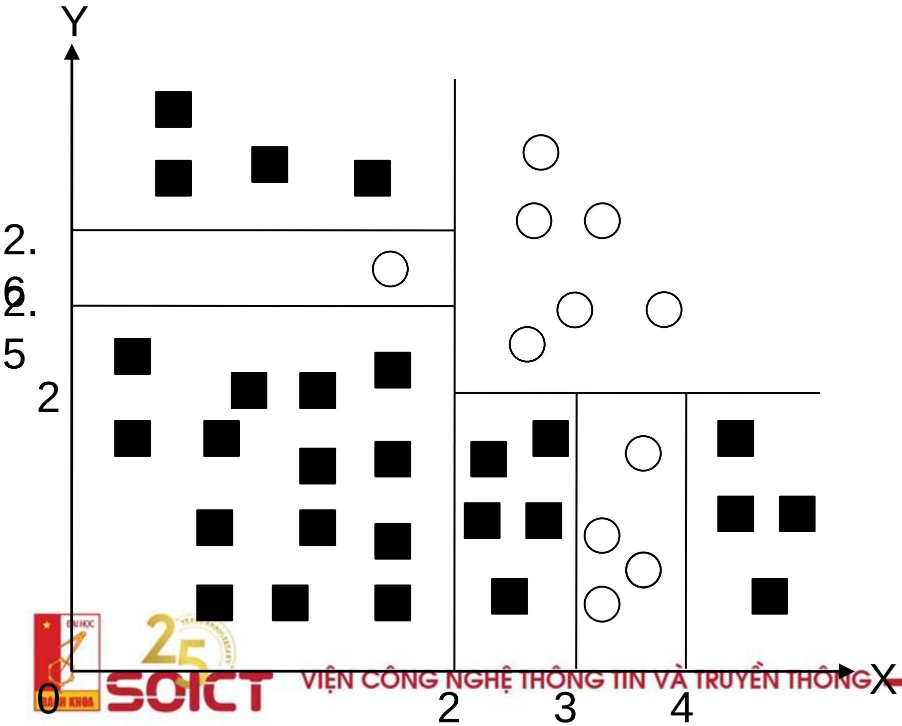
s: number of different values of A_i

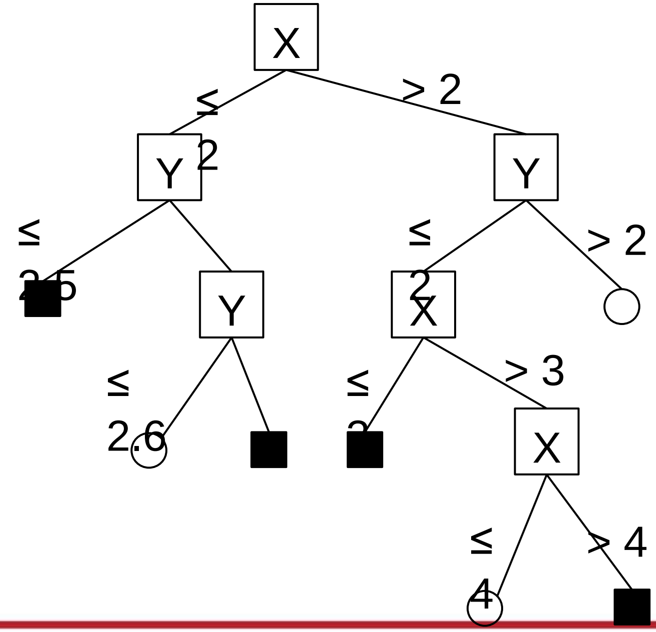
D_j is a subset of D whose attribute A_i has value j

$$\text{GainRatio}(D, Ai) = \frac{\text{Gain}(D, Ai)}{-\sum_{j=1}^{s} \left(\frac{D_j}{D_j} + \frac{D_j}{D_j} \right)}$$

3.3 Handle persistent attributes

- Split the attribute into two intervals (binary split)
- The dividing threshold is chosen to maximize information gain (ratio)
- During tree creation, the attribute is not deleted (line 20)





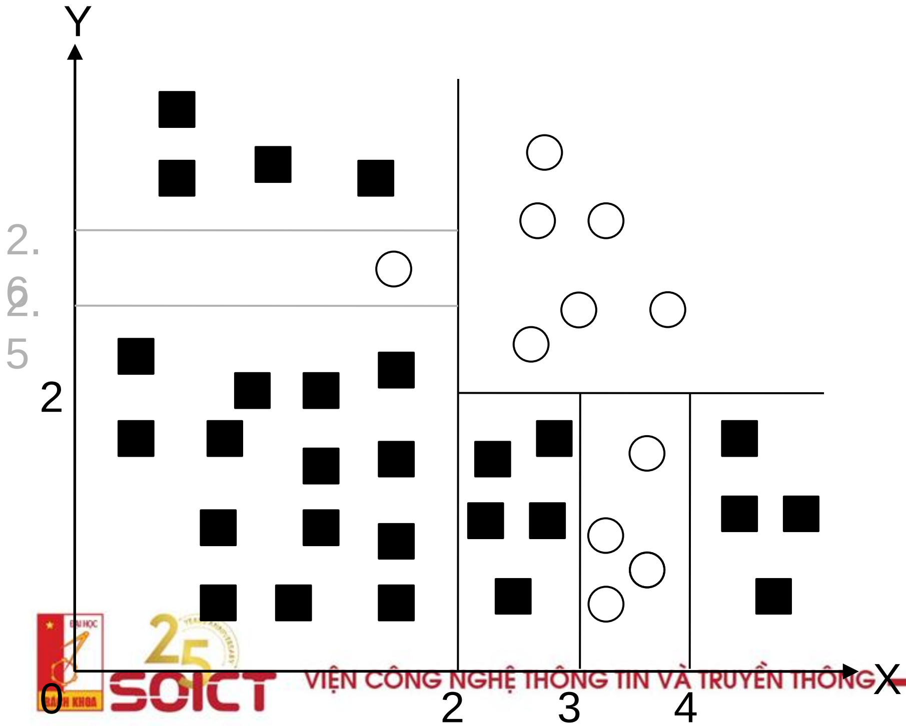
3.4 Some advanced issues

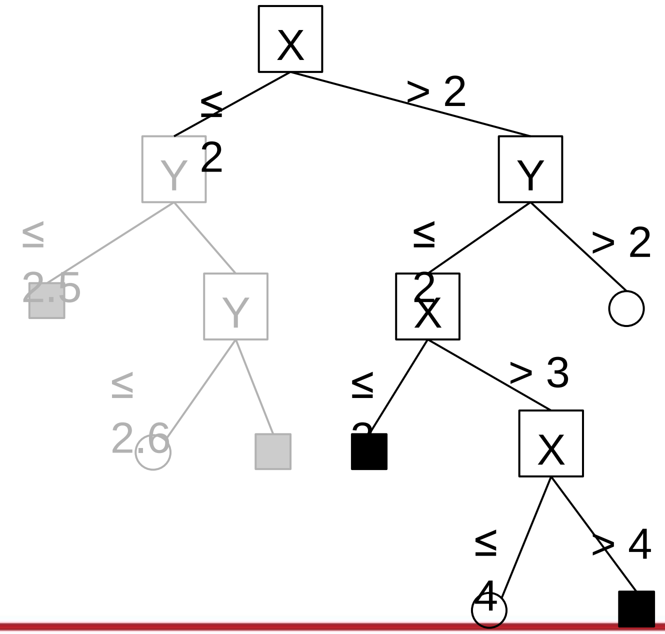
- **Overfitting**: A classifier *f* is said to be overfit if there exists a classifier *f'* whose accuracy *f* > *f'* trên DL on the training set but < on test set
  - Cause: data contains noise (wrong class label or wrong attribute value) or complex classification problem or contains randomness.
  - Pruning: decision tree is too deep → prune the tree by estimating the error at each branch, if the error of the subtree is larger then pruning. An independent data set (validation set) can be used for pruning. Alternatively, pre-pruning or law pruning can be applied
- Missing value: Use "unknown" or most common value (or average with continuous attribute)
- Class imbalance: Over sampling, ranking

Rule 1: X ≤ 2, Y > 2.5, Y > 2.6 → ■
Rule 2: X ≤ 2, Y > 2.5, Y ≤ 2.6 → ○
Rule 3: X ≤ 2, Y ≤ 2.5 → ■

X ≤ 2, Y > 2.6 → ■

X ≤ 2 → ■





4. Naive Bayes Algorithm

4.1 Naive Bayes Algorithm

- Given a attribute set A_1, A_2, ..., $A|A|$, C is a class attribute with values c_1, c_2, …, $c_{|C|}$, example test$d$ = $<A_1=a_1, \ldots, A_{|A|}=a_{|A|}>$
- Assumption MAP (maximum a posteriori): find class c_j such that $\Pr(C=c_j|A_1=a_1, \ldots, A_{|A|}=a_{|A|})$ Given an attribute set A1, A2, ..., A|A|, C is a class attribute with values c1, c2, …, c|C|, for example test d = <A1=a1, …, A|A|=a|A|>Assumption MAP (maximum a posteriori): find class cj such that Pr(C=cj|A1=a1, ..., A|A|=a|A|) is maximal

$$\Pr(C=c_j| A_1=a_1,\ldots, A_{|A|}=a_{|A|}) = \frac{\Pr(A_1=a_1, \ldots, A_{|A|}=a_{|A|}| C=c_j) \times \Pr(C=c_j)}{\Pr(A_1=a_1, \ldots, A_{|A|}=a_{|A|})}$$

$$\Pr(C=c_j| A_1=a_1,\ldots, A_{|A|}=a_{|A|}) \propto \Pr(A_1=a_1, \ldots, A_{|A|}=a_{|A|}| C=c_j) \times \Pr(C=c_j)$$

$$\Pr(A_1=a_1, ..., A_{|A|}=a_{|A|}| C=c_j) = \Pr(A_1=a_1| A_2=a_2,..., A_{|A|}=a_{|A|}, C=c_j)$$
$$\times \Pr(A_2=a_2,..., A_{|A|}=a_{|A|}| C=c_j)$$
$$= \Pr(A_1=a_1| A_2=a_2,..., A_{|A|}=a_{|A|}, C=c_j)$$
$$\times \Pr(A_2=a_2| A_3=a_3,..., A_{|A|}=a_{|A|},C=c_j)$$
$$\times \Pr(A_3=a_3,..., A_{|A|}=a_{|A|}| C=c_j)$$
$$= ...$$

Conditional Independence Assumption:

$$\Pr(A_1=a_1| A_2=a_2,..., A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1| C=c_j)$$
$$\Pr(A_2=a_2| A_3=a_3,..., A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_2=a_2| C=c_j)$$
...

$$\Pr(A_1=a_1, ..., A_{|A|}=a_{|A|}| C=c_j) = \Pr(A_1=a_1| C=c_j) \times \Pr(A_2=a_2| C=c_j) \times \dots \times \Pr(A_n=a_n| C=c_j)$$

$$\boxed{\Pr(C=c_j| A_1=a_1,..., A_{|A|}=a_{|A|}) \propto \Pr(A_1=a_1| C=c_j) \times \Pr(A_2=a_2| C=c_j) \times \dots \times \Pr(A_n=a_n| C=c_j) \times \Pr(C=c_j)}$$

$$\Pr(C=c_j) = \frac{\text{number of examples with class } c_j}{\text{total number of examples in the dataset}}$$

$$\Pr(A_i=a_i \mid C=c_j) = \frac{\text{number of examples with } A_i = a_i \text{ and class } c_j}{\text{number of examples } c_j}$$

A

B

C

m

b t

m

s t

g

g t

$\Pr(C = t) = 1/2$ $\Pr(C = f) = 1/2$

$\Pr(A = m \mid C = t) = 2/5$ $\Pr(A = g \mid C = t) = 2/5$ $\Pr(A = h \mid C = t) = 1/5$

$\Pr(A = m \mid C = f) = 1/5$ $\Pr(A = g \mid C = f) = 2/5$ $\Pr(A = h \mid C = f) = 2/5$

$\Pr(B = b \mid C = t) = 1/5$ $\Pr(B = s \mid C = t) = 2/5$ $\Pr(B = q \mid C = t) = 2/5$

$\Pr(B = b \mid C = f) = 2/5$ $\Pr(B = s \mid C = f) = 1/5$ $\Pr(B = q \mid C = f) = 2/5$

A = m, B = q, C = ?

$\Pr(C = t \mid A = m, B = q) \propto \Pr(C = t) \times \Pr(A = m \mid C = t) \times \Pr(B = q \mid C = t)$

$\propto 1/2 \times 2/5 \times 2/5 =$

$\Pr(C = f \mid A = m, B = q) \propto \Pr(C = f) \times \Pr(A = m \mid C = f) \times \Pr(B = q \mid C = f)$

- Probability – 0: The attribute value a_i is not x/h of the same class c_j in the training data makes $\Pr(A_i = a_i|C = c_j) = 0$

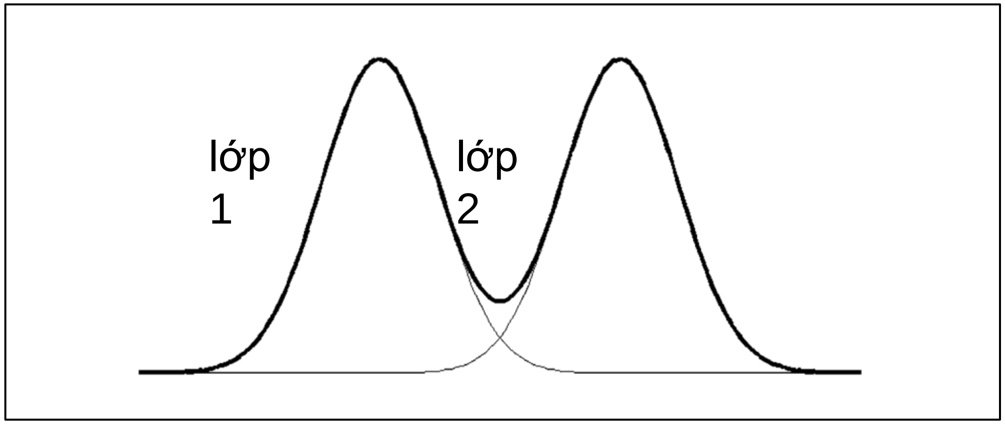
$$\Pr(A_i = a_i|\ C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda \times m_i}$$

where n_{ij} is the example number with $A_i = a_i$ và $C = c_j$; m_i is the number of different values of the attribute A_i

- $\lambda = 1/n$ where n is the number of training examples
- $\lambda = 1$: Laplace smoothing

4.2 Text classification based on NB

- Probabilistic generation model: Assume that each document is generated by a distribution according to the hidden parameters. These parameters are estimated based on the training dataset. The parameters are used to classify the test text by using Bayes' law to calculate the posterior probability of the class that is likely to produce the text.
- Two assumptions: i) dataset is generated by a mixing model ii) Each mixing component corresponds to a class.



The probability distribution function of two Gaussian with parameters $\theta_1 = (\mu_1, \sigma_1)$ và $\theta_1 = (\mu_1, \sigma_1)$

- Suppose there are K mixed components, component j has parameter θ_j, the parameter of the whole model includes $\Theta = (\varphi_1, \varphi_2, \ldots, \varphi_k, \theta_1, \theta_2, \ldots, \theta_k)$ where φ_j is the weight of component j ($\Sigma\varphi_j = 1$)
- Assuming there are classes $c_1, c_2, \ldots, c_{|C|}$, we have $|C| = K$, $\varphi_j = \Pr(c_j|\Theta)$, the text generation process d_i:

1. Select a mixing component j based on the a *priori probabilities* of the classes, $\varphi_j = \Pr(c_j|\Theta)$
2. Generate d_i based on the distribution $\Pr(d_i| c_j; \theta_j)$

- Probability of generating d_i based on the whole model:

$$\Pr(d_j| \Theta) = \sum_{j=1}^{|C|} \Pr(c_j| \Theta) \times \Pr(d_i| c_j; \theta_j)$$

- Text is represented as a bag of words

1. Words are generated independently, independent of the context (other words in the text)

2. Probability of the word does not depend on the position in the text

3. Length and class of text are independent of each other

- Each text is generated by a polynomial distribution of words with n independent trials where n is the length of the text.

- Polynomial test is a process that generates k values ($k \geq 2$) with probabilities $p_1, p_2, \ldots, p_k$

Example: The dice produce 6 values 1, 2, …, 6 with fair probability $p_1 = p_2 = \ldots = p_6 = 1/6$)

- Suppose there are n independent trials, let X_t be the number of times the value t is generated, then $X_1, X_2, \ldots, X_k$ are discrete random variables.

($X_1, X_2, \ldots, X_k$) follows a polynomial distribution with parameters $n, p_1, p_2, \ldots, p_k$

- n: document length $|d_i|$
- $k = |V|$: dictionary length
- p_t: Probability of word w_t x/h in document, $\Pr(w_t| c_j; \Theta)$
- X_t: Random variable representing the number of times w_t x/h in document
- N_{ti}: Number of times w_t x/h in d_i
- Probability Function:

$$\Pr(d_i \mid c_j;\Theta) = \Pr(|d_i|)\,|d_i|!\prod_{t=1}^{|V|}\frac{\Pr(w_t \mid c_j;\Theta)^{N_{ti}}}{N_{ti}!}$$

$$\sum_{t=1}^{|V|} N_{ti} = |d_i| \qquad \sum_{t=1}^{|V|}\Pr(w_t \mid c_j;\Theta) = 1$$

- Parameter estimation:

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)}.$$

- *Lidstone* smoothing (λ < 1)

(λ = 1: *Laplace* smoothing)

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{\lambda + \sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{\lambda |V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)}$$

$$\Pr(c_j \mid \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|}$$

- Classify:

$$\Pr(c_j \mid d_i; \hat{\Theta}) = \frac{\Pr(c_j \mid \hat{\Theta}) \Pr(d_i \mid c_j; \hat{\Theta})}{\Pr(d_i \mid \hat{\Theta})}$$

$$= \frac{\Pr(c_j \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j; \hat{\Theta})}{\sum_{r=1}^{|C|} \Pr(c_r \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_r; \hat{\Theta})}$$

$$\arg\max_{c_j \in C} \Pr(c_j \mid d_i; \hat{\Theta}).$$

5. SVM Algorithm

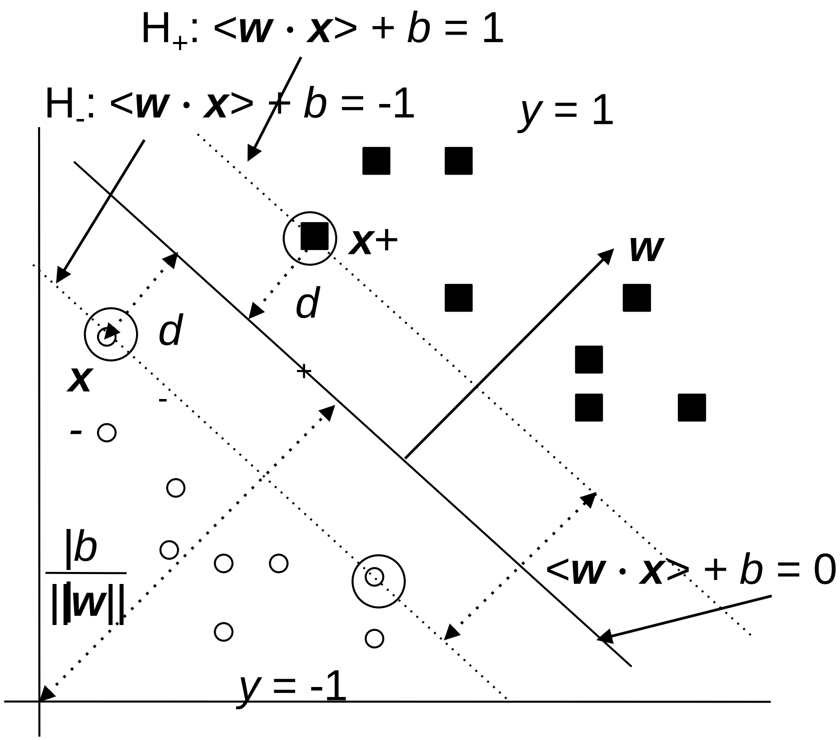
- Support vector machine (SVM) is a linear learning system for building 2-class classifiers.
- Example set D: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots(\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ input vector r-dimension in space $X \subseteq R^r$, y_i is class label, $y_i \in \{1, -1\}$
- SVM constructs a linear function f: $X \quad R^r \to R$ với $\mathbf{w} = (w_1, w_2, \ldots, w_r) \in R^r$

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

$$y_i = \begin{cases} 1 & \text{if } f(\mathbf{x}_i) \geq 0 \\ -1 & \text{if } f(\mathbf{x}_i) < 0 \end{cases}$$

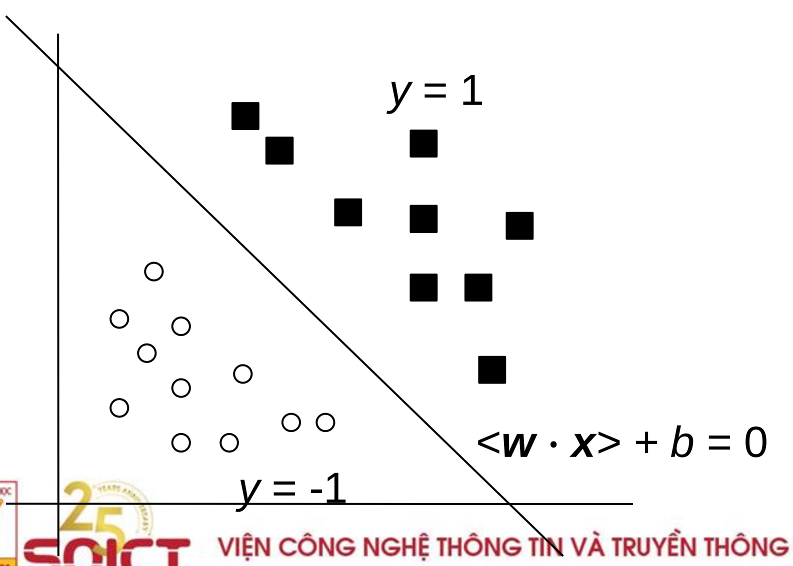
5.1 Linear SVM: divisible data

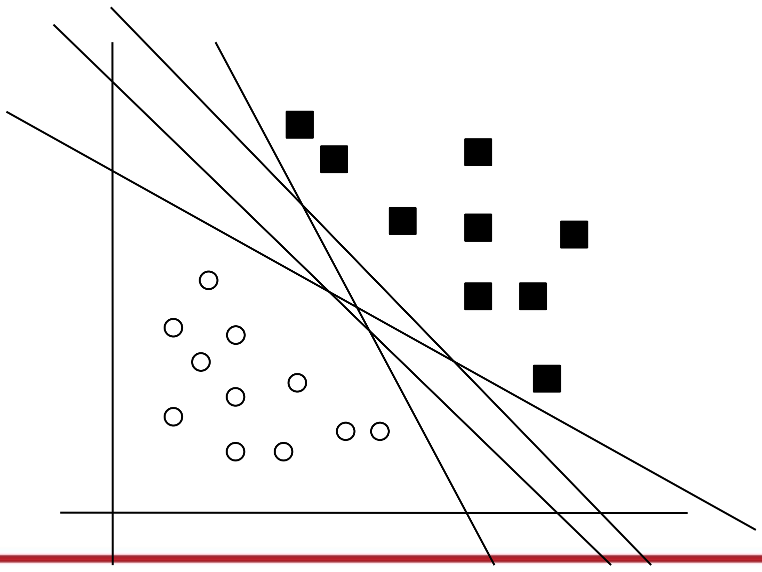
- ***w***: normal vector of hyperplane
- SVM finds hyperplane to maximize margin
- Structural risk minimization principle: Marginal maximization minimizes the upper bound of error (classification)



$$d_+ = d_- = \frac{1}{\|\boldsymbol{w}\|}$$

- Hyper-plane dividing two layers
- There are countless such hyperplanes, how to choose?
- What to do if the data is not linearly divisible?





- Constrained minimization problem

Minimization : $\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle / 2$

Constraint : $y_i(\langle \boldsymbol{w} \cdot \boldsymbol{x}_i \rangle + b) \geq 1 \quad i = 1, 2, \ldots, n$

- Since the squared and convex objective functions, the constraints are linear, we use the Lagrange multiplier method

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1]$$

where α_i is a *Lagrange multiplier*

General problem

Minimization: $f(\boldsymbol{x})$

Constraints: $g_i(\boldsymbol{x}) \le b_i$ $i = 1, 2, \ldots, n$

Lagrangian:

$$L_P = f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i [g_i(\mathbf{x}) - b_i)]$$

Conditions *Kuhn -Tucker*:

$\frac{\partial L}{\partial x_j} = 0,$ $j = 1, 2, \ldots, r$

$g_i(\boldsymbol{x}) - b_i \le 0,$ $i = 1, 2, \ldots, n$

$\alpha_i \ge 0, i = 1, 2, \ldots, n$

$\alpha_i(b_i - g_i(\boldsymbol{x})) = 0, i = 1, 2, \ldots, n$

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i \alpha_i x_{ij} = 0, \ \ j = 1, 2, \ldots, r$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n} y_i \alpha_i = 0$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \ge 0, \ \ i = 1, 2, \ldots, n$$

$$\alpha_i \ge 0, \ \ i = 1, 2, \ldots, n$$

$$\alpha_i(y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1) = 0, \ \ i = 1, 2, \ldots, n$$

$\alpha_i > 0$ corresponds to dataset called ***support vectors***

- Due to the convex objective function and linear constraints *Kuhn - Tucker* conditions are necessary and sufficient, replacing the original problem with a dual problem (*Wolfe duality*)

Maximize $$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

Constraints $$\sum_{i=1}^{n} y_i \alpha_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, n.$$

Hyperplane $$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0$$

Classify examples ***z***: $$sign(\langle \mathbf{w} \cdot \mathbf{z} \rangle + b) = sign\left(\sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{z} \rangle + b \right)$$

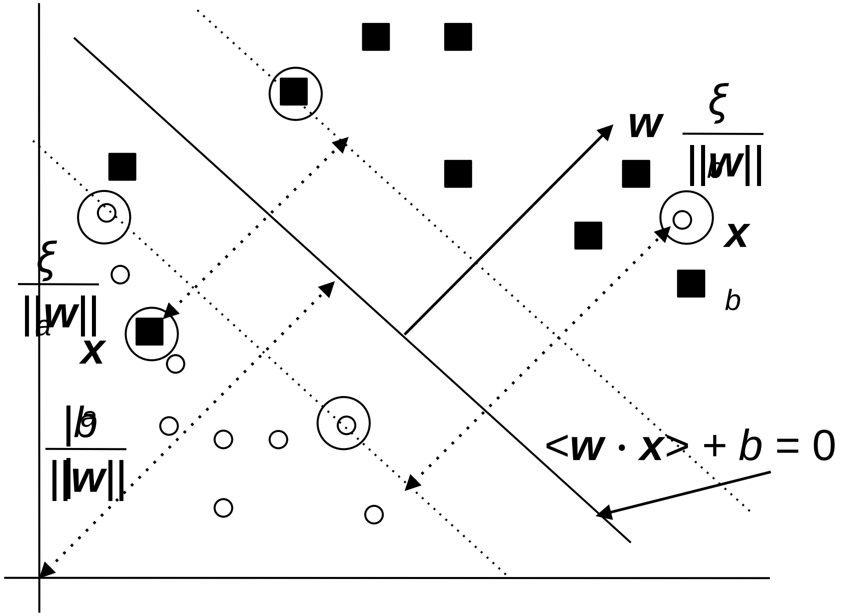
(*sv*: set of support vectors)

5.2 Linear SVM: Indivisible dataset

Minimization: $$\frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i=1}^{n} \xi_i$$

Constraints: $y_i(\langle \boldsymbol{w} \cdot \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i,\ i = 1, 2, \ldots, n$

$\xi_i \geq 0$, i = 1, 2, …, n



Lagrangian :

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i$$

Conditions *Kuhn – Tucker*:

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i \alpha_i x_{ij} = 0, \quad j = 1, 2, ..., r$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n} y_i \alpha_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \quad i = 1, 2, ..., n$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i \geq 0, \quad i = 1, 2, ..., n$$

$$\xi_i \geq 0, \quad i = 1, 2, ..., n$$

$$\alpha_i \geq 0, \quad i = 1, 2, ..., n$$

$$\mu_i \geq 0, \quad i = 1, 2, ..., n$$

$$\alpha_i(y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0, \quad i = 1, 2, ..., n$$

$$\mu_i \xi_i = 0, \quad i = 1, 2, ..., n$$

Dual Problems

Maximize:

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

Constraints:

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, ..., n$$

$$b = \frac{1}{y_i} - \sum_{i=1}^{n} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$$

Hyperplane

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i=1}^{n} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0.$$

Comments:

$\alpha_i = 0 \rightarrow y_i(<\boldsymbol{w} \cdot \boldsymbol{x}> + b) \geq 1$ và $\xi_i = 0$

$0 < \alpha_i < C \rightarrow y_i(<\boldsymbol{w} \cdot \boldsymbol{x}> + b) = 1$ và $\xi_i = 0$

$\alpha_i = C \rightarrow y_i(<\boldsymbol{w} \cdot \boldsymbol{x}> + b) \leq 1$ và $\xi_i \geq 0$

5.3 Nonlinear SVM: Kernel Function

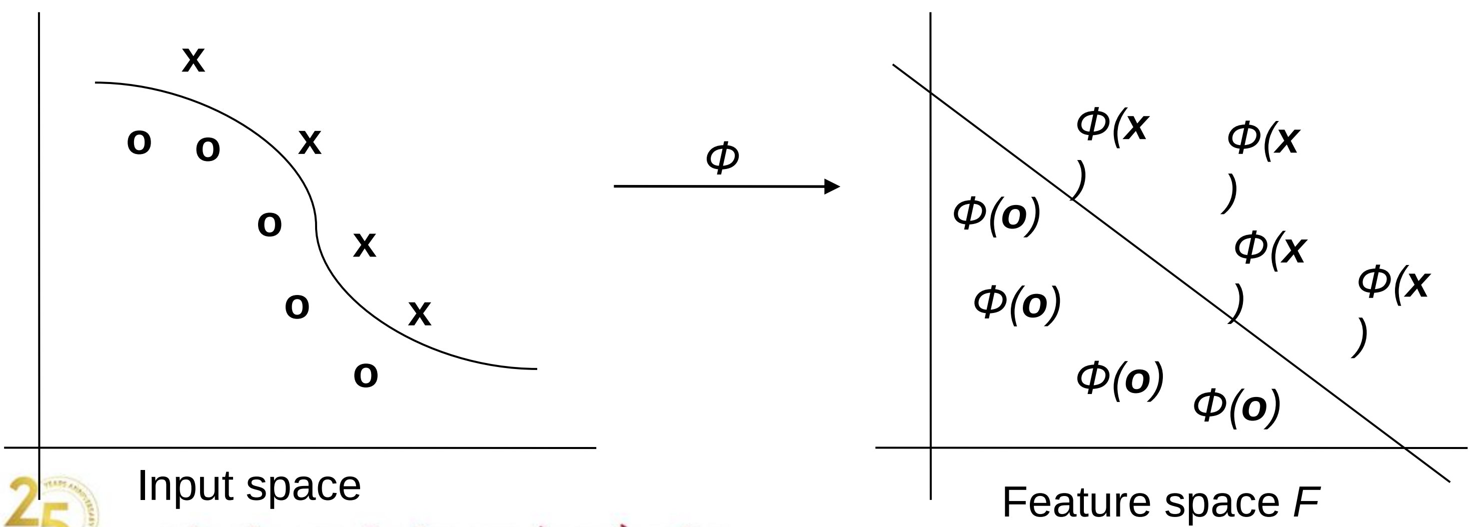
$\{(\Phi(\mathbf{x_1}), y_1), (\Phi(\mathbf{x_2}), y_2), \ldots, (\Phi(\mathbf{x_n}), y_n)\}$

$\Phi: X \rightarrow F$
$\mathbf{x} \mapsto \Phi(\mathbf{x})$

Minimization $$\frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i=1}^{n} \xi_i$$

Constraints $$y_i(<\mathbf{w} \cdot \Phi(\mathbf{x}_i)> + b) \geq 1 - \xi_i,\ i = 1, 2, \ldots, n$$

$$\xi_i \geq 0,\ \mathrm{i} = 1, 2, \ldots, n$$



Dual Problems

Maximize $$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$$

Constraints $$\sum_{i=1}^{n} y_i \alpha_i = 0$$

$$0 \le \alpha_i \le C, \quad i = 1, 2, \ldots, n$$

Classify

$$\sum_{i=1}^{n} y_i \alpha_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b$$

Eg: $(x_1, x_2) \mapsto (x_1^2, x_2^2, 2^{1/2} x_1 x_2)$

$(2,3) \mapsto (4, 9, 8.5)$

- Kernel function: a vector multiplication on the input space corresponds to a vector multiplication on a multidimensional feature space

$$K(\boldsymbol{x}, \boldsymbol{z}) = <\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{z})>$$

- Polynomial kernel: $K(\boldsymbol{x}, \boldsymbol{z}) = <\boldsymbol{x} \cdot \boldsymbol{z}>^d$

Eg:

$\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{z} = (z_1, z_2)$

$<\boldsymbol{x} \cdot \boldsymbol{z}>^2 = (x_1z_1 + x_2z_2)^2$

$= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$

$= <(x_1^2, x_2^2, 2^{1/2}x_1x_2) \cdot (z_1^2, z_2^2, 2^{1/2}z_1z_2)>$

$= <\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{z})>$

- Feature space of the kernel polynomial of degree *d* has C_d^{r+d-1}p dimension
- Mercer theorem defines kernel functions
- Polynomial kernel:

$$K(\boldsymbol{x}, \boldsymbol{z}) = (<\boldsymbol{x} \cdot \boldsymbol{z}> + \theta)^{d}$$

- Gaussian RBF:

$$K(\mathbf{x},\mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2/2\sigma}$$

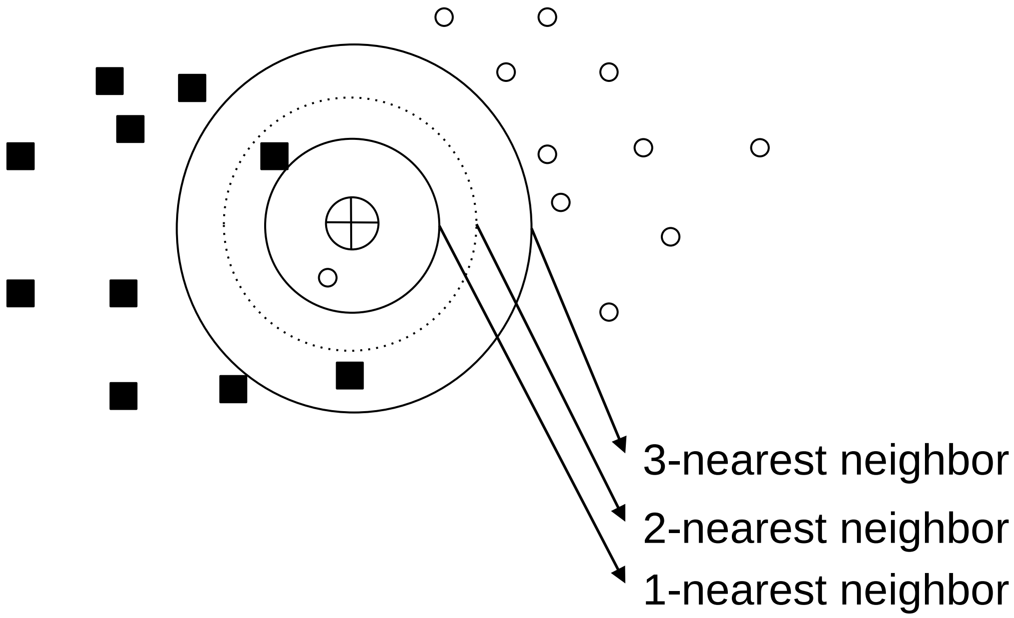
- In order for SVM to work with discrete properties, it is possible to convert to binary
- For multiclass classification, strategies such as one-vs-all or one-vs-one can be used
- Hyperplane is confusing for users, so SVM is often used in applications that don't require explanation

6. kNN

- Given training dataset D, a test example d

1. Calculate the similarity (distance) of d with all training examples.
2. Determine k closest examples based on similarity
3. Classify d based on the labels of k examples above

- Defect:
  - Long sorting time
  - Difficult to explain


3-nearest neighbor
2-nearest neighbor
1-nearest neighbor

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!