Artificial Intelligence
(IT3160E)

Than Quang Khoat

khoattq@soict.hust.edu.vn

School of Information and Communication Technology
Hanoi University of Science and Technology

2022

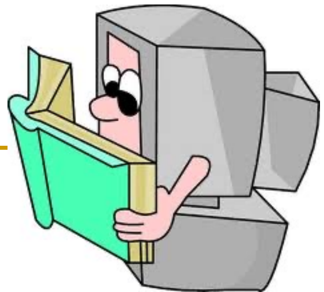
Content:

- Introduction of Artificial Intelligence
- Intelligent agent
- Problem solving: Search, Constraint satisfaction
- Logic and reasoning
- Knowledge representation
- **Machine learning**

Introduction to Machine Learning

- Machine Learning (ML) is an active subfield of Artificial Intelligence.
- ML seeks to answer the question:
  - → *"How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?"* [Mitchell, 2006]

- Some other views on ML:
  - → Build systems that automatically improve their performance [Simon, 1983]
  - → Program computers to optimize a performance objective at some task, based on data and past experience [Alpaydin, 2010]



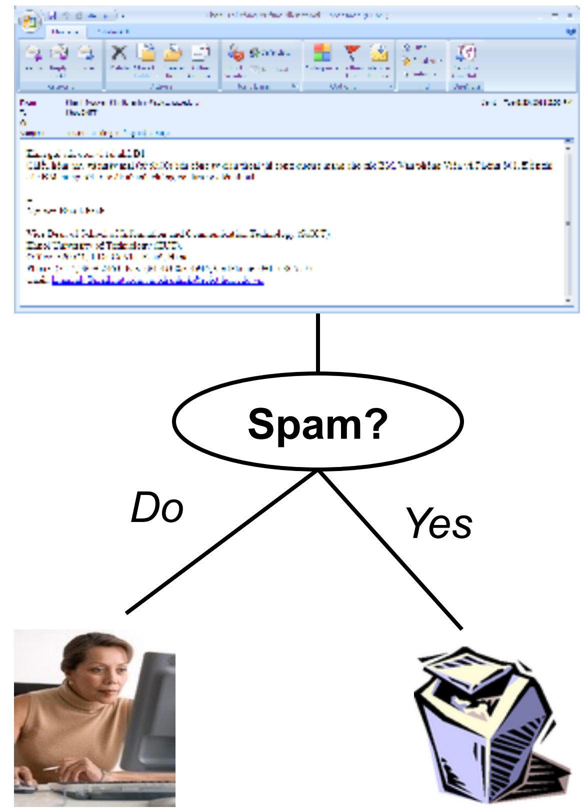
A learning machine

- We say that a machine learns if the system reliably improves its performance P at task T, following experience E.
- A learning problem can be described as a triple (***T, P, E***)
  - ***T:*** a task
  - ***P:*** the evaluation criteria of performance
  - ***E:*** experience

Example of ML problem (1)

Email spam filtering:

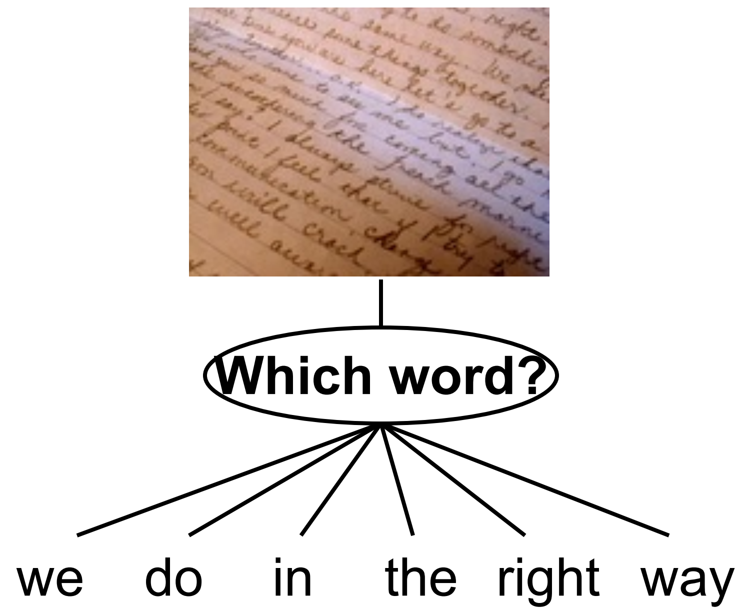
- ***T*** : To predict (i.e., to filter) spam emails
- ***P***: % of correctly classified (i.e., predicted) incoming emails
- ***E***: A set of sample emails, where each email is represented by a set of attributes (e.g., a set of keywords) and its corresponding label (i.e., normal or spam)



Example of ML problem (2)

Handwritten characters recognition

- ***T***: To recognize the words that appear in a captured image of a handwritten document
- ***P***: % of correctly recognized words
- ***E***: A set of captured images of handwritten words, where each image associates with a word's label (ID)



Example of ML problem (3)

Image tagging

- ***T***: give some words that describe the meaning of a picture
- ***P***: ?
- ***E***: set of pictures, each has been labelled with a set of words.

FISH WATER OCEAN TREE CORAL

PEOPLE MARKET PATTERN TEXTILE DISPLAY

BIRDS NEST TREE BRANCH LEAVES

Learning machine (1)

- A mapping (function):

$$f: x \longmapsto y$$

  - x: observations (data), past experience
  - y: prediction, new knowledge, new experience,…
- Regression: if y is a real number
- Classification: if y only belongs to a discrete set

Learning machine (2)

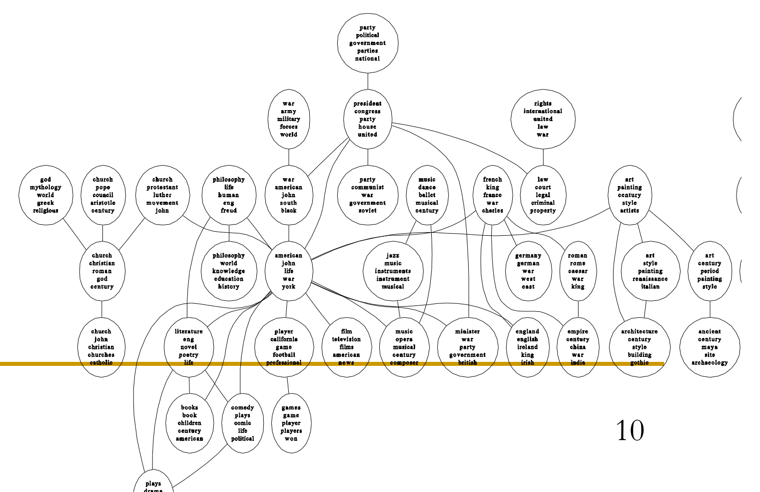
- Where to learn?
  - Learn from a set of training examples (training set).
    $\{x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_M\}$

- After learning:
  - We obtain a model, new knowledge, or new experience.
  - We can use that model/function to do prediction or inference for future observations.

$$y = f(x)$$

Two basic learning problems

- Supervised learning (học có giám sát): learn a function $y = f(x)$ from a given training set $\{(x_1, y_1), \ldots, (x_M, y_M)\}$ so that $y_i \cong f(x_i)$ for every i.
  - Classification (categorization): if y only belongs to a discrete set, for example {fish, plant, fruit, cat}
  - Regression*:* if y is a real number
- Unsupervised learning (học không giám sát): learn a function $y = f(x)$ from a given training set $\{x_1, x_2, \ldots, x_{\mathrm{M}}\}$.
  - Y can be a data cluster
  - Y can be a hidden structure

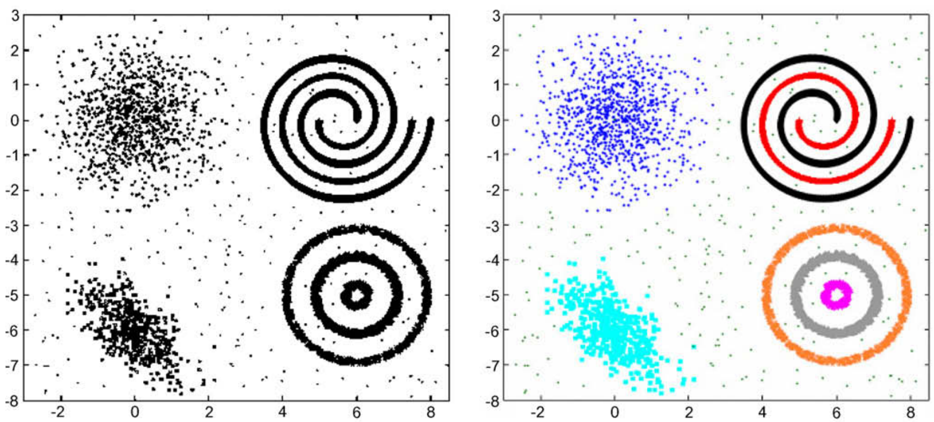


Supervised learning: Examples

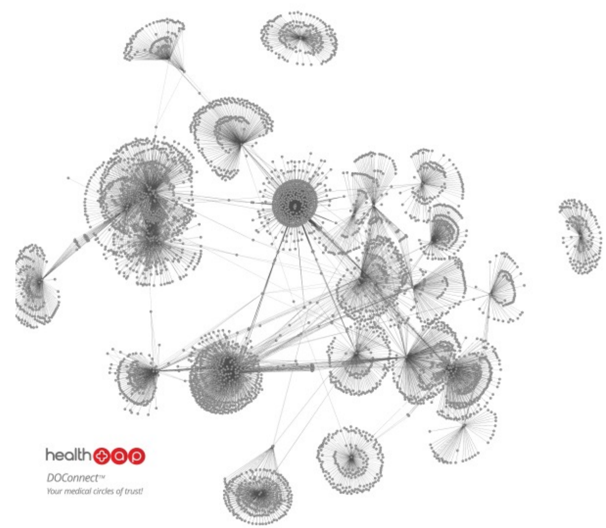
- Email spam filtering
- Web page categorization
- Risk estimation of loan application
- Prediction of stock indices
- Discovery of network attacks

Unsupervised learning: Examples (1)

- Clustering data into clusters
  - Discover the data groups/clusters

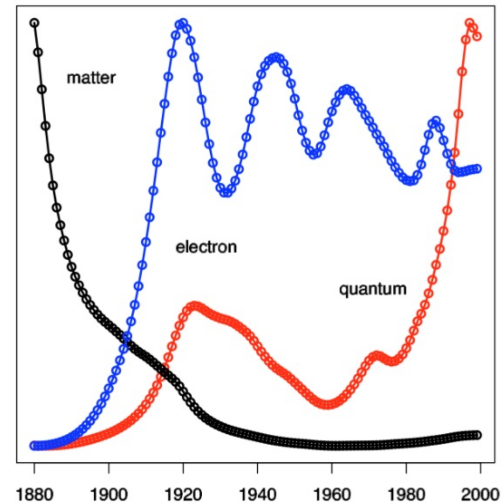


- Community detection
  - Detect communities in online social networks

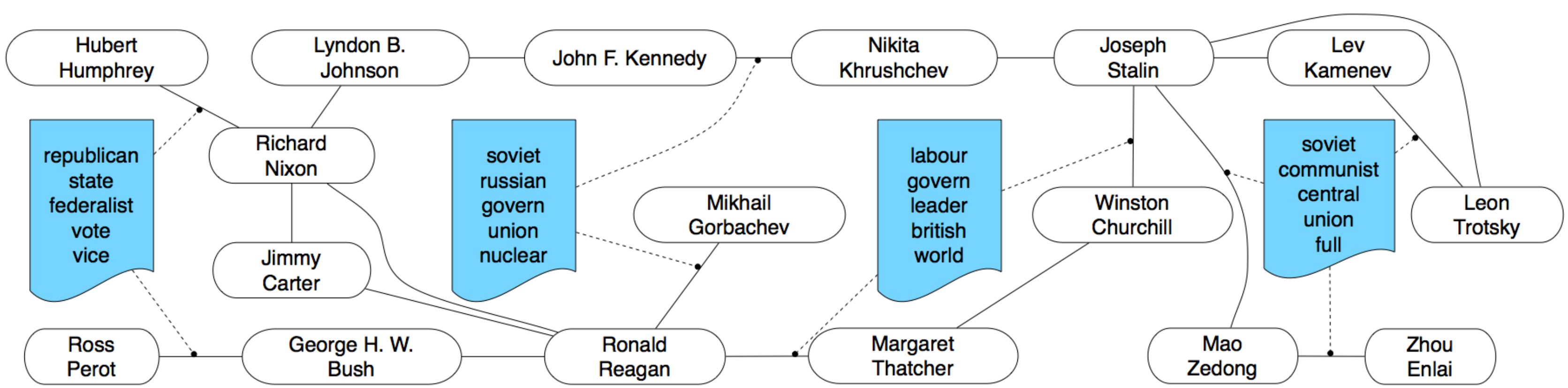


Unsupervised learning: Examples (2)

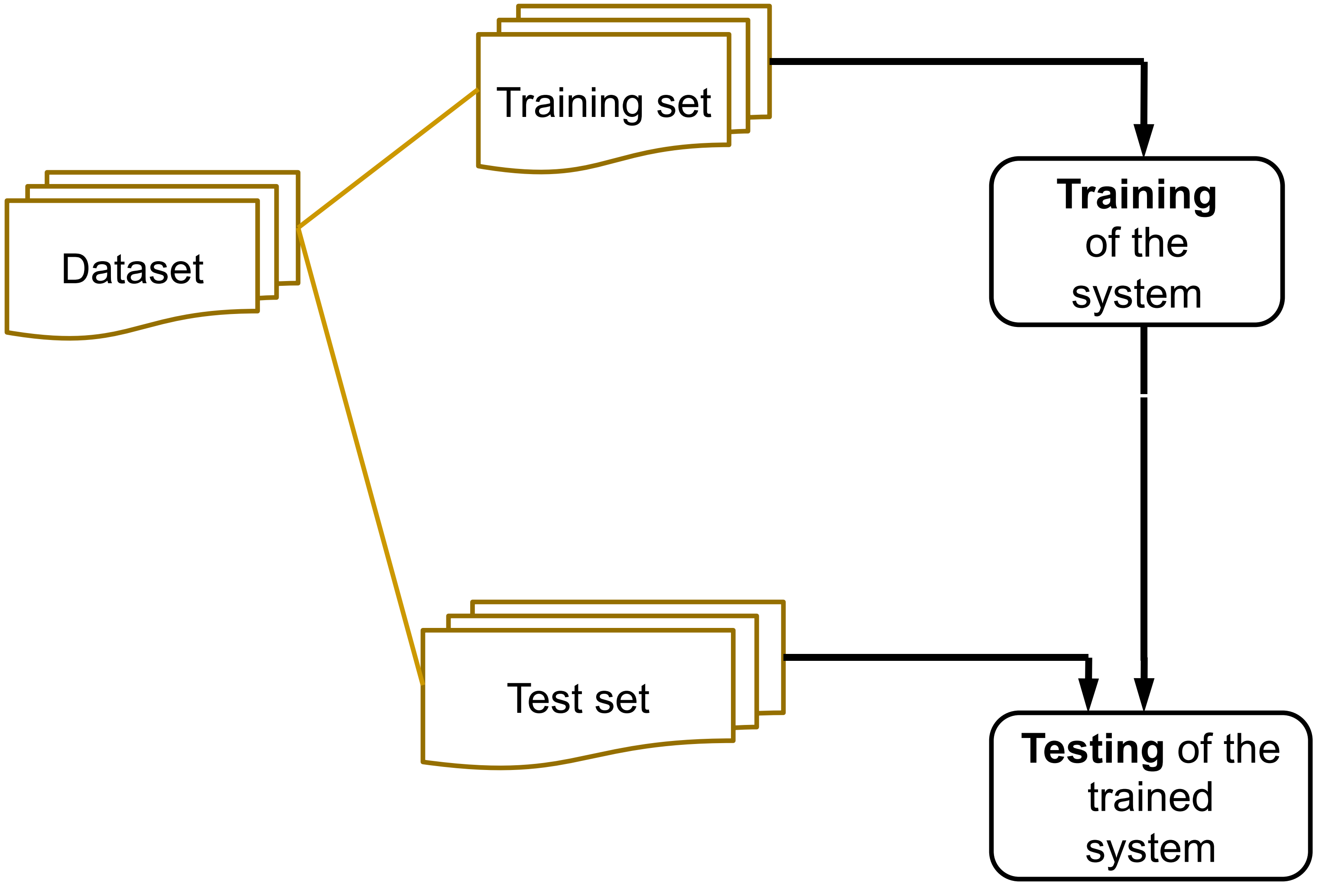
- Trends detection
  - Discover the trends, demands, future needs of online users



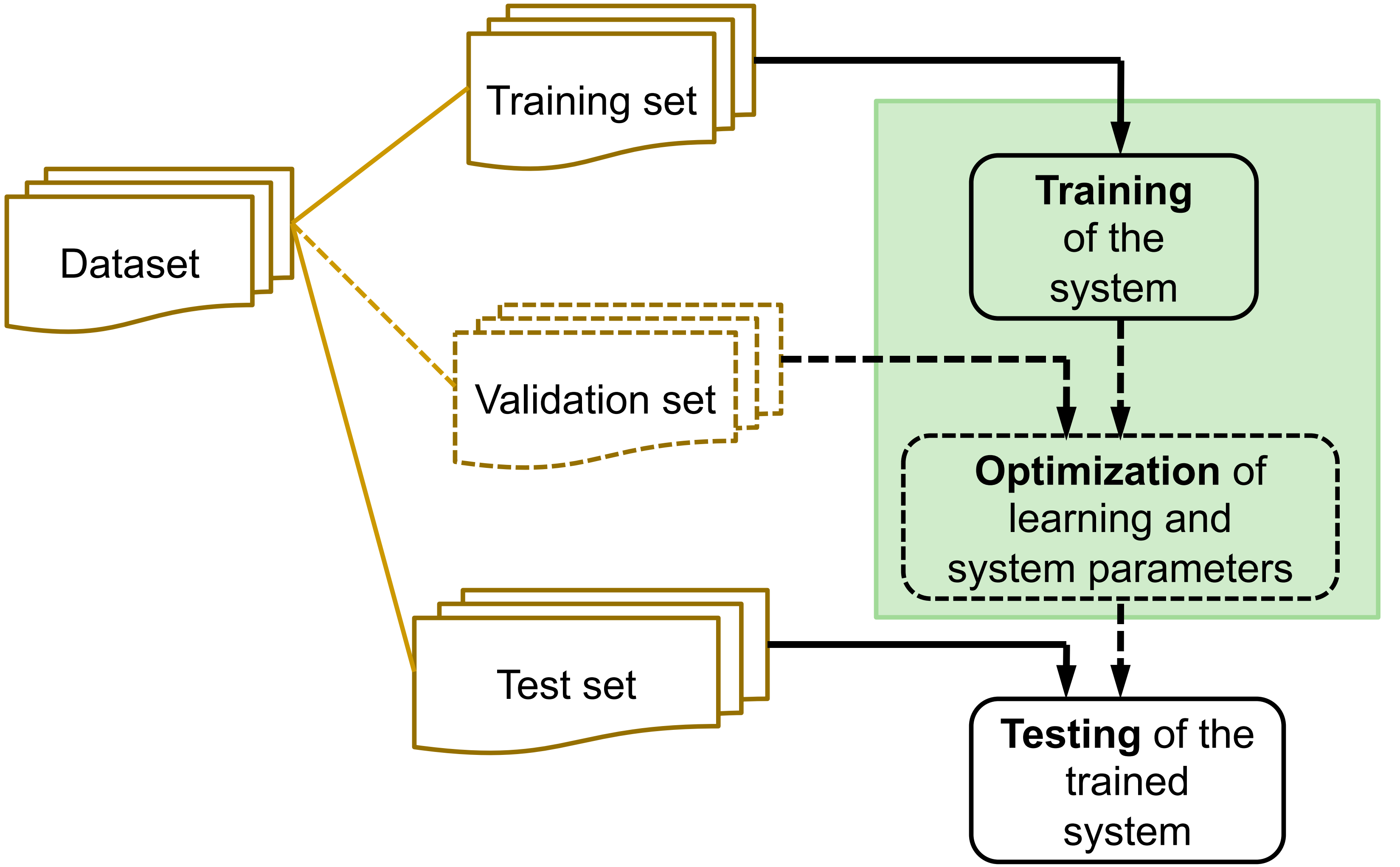
- Entity-interaction analysis



ML processes: basic



ML processes: careful



Designing a ML system (1)

- Training (learning) examples
  - The training feedback is included in training examples or indirectly provided (e.g., from the working environment)
  - They are supervised or unsupervised training examples
  - The training examples should be compatible with (i.e., representative for) the future test examples

- The target function to be learned
  - F: $X \rightarrow \{0,1\}$
  - F: $X \rightarrow$ A set of class labels
  - F: $X \rightarrow R^+$ (i.e., a domain of positive real values)
  - …

Designing a ML system (2)

- Representation of the target function to be learned
  - A polynomial function
  - A set of rules
  - A decision tree
  - An artificial neural network
  - …
- ML algorithm that can learn approximately the target function
  - Regression-based
  - Rule induction
  - ID3 or C4.5
  - Back-propagation
  - …

Challenges in ML (1)

- Learning algorithm
  - Which learning algorithms can learn approximately a given target function?
  - Under which conditions, a selected learning algorithm converges (approximately) the target function?
  - For a specific application problem and a specific example (object) representation, which learning algorithm performs best?

Challenges in ML (2)

- Training examples
  - How many training examples are enough for the training?
  - How does the size of the training set (i.e., the number of training examples) affect the accuracy of the learned function?
  - How do error (noise) and/or missing-value examples affect the accuracy?

Challenges in ML (3)

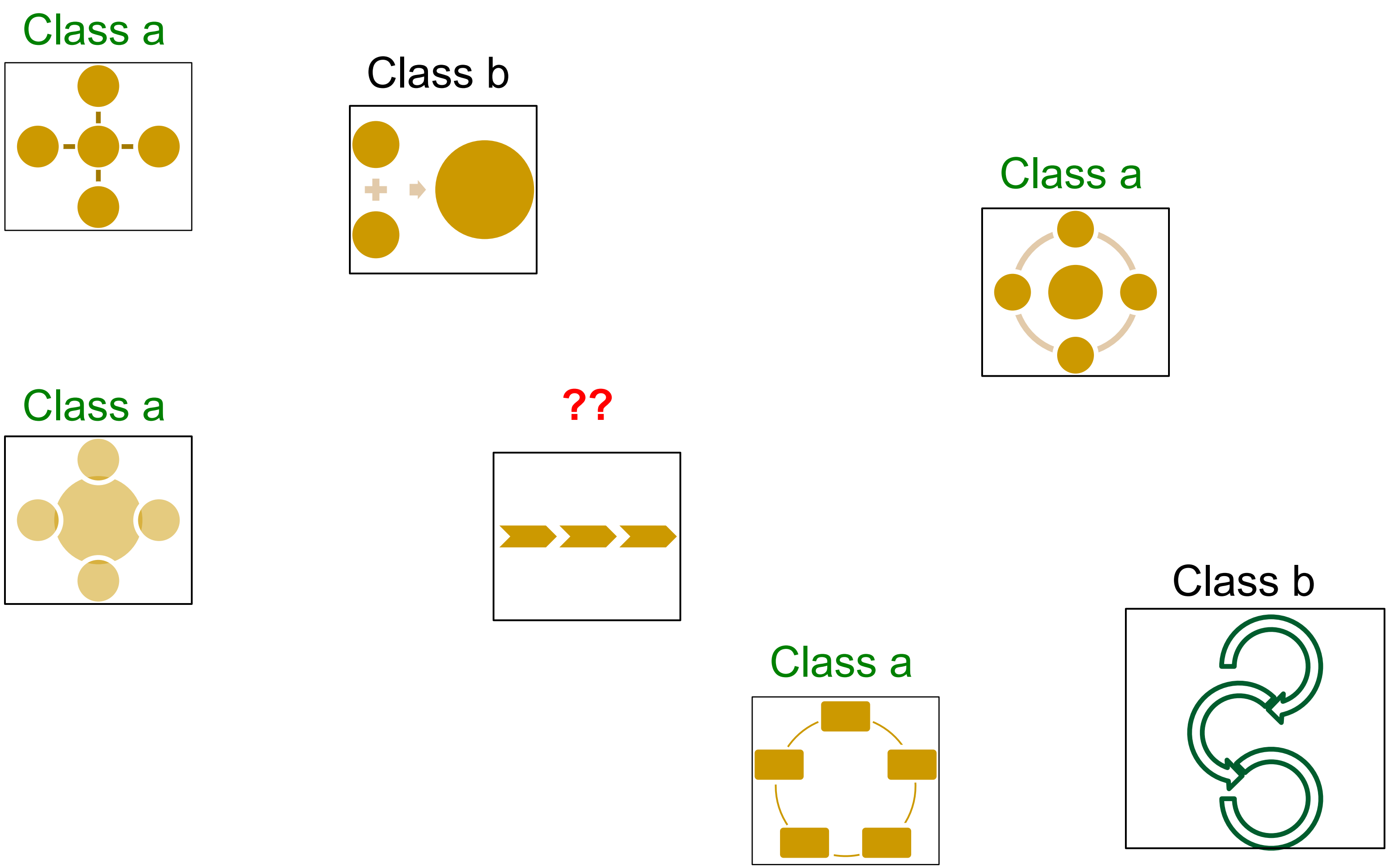
- Learning process
  - What is the best ways of use order of training examples?
  - How does the domain knowledge (apart from the training examples) contribute to the machine learning process?

Challenges in ML (4)

- Learning capability
  - Which target function the system should learn?
    - Representation of the target function: Representation capability (e.g., linear / non-linear function) vs. Complexity of the learning algorithm and learning process
  - Limits for the learning capability of learning algorithms?
  - The system's capability of **generalization** from the training examples?
    - The ultimate goal of ML systems
    - Avoid *Overfitting problem* (high accuracy on the training set, but low accuracy on the validation and test sets)
  - Adaptability of a system with continual changes in the environment?

Classification problem

Which class does the object belong to?



Nearest neighbors learning

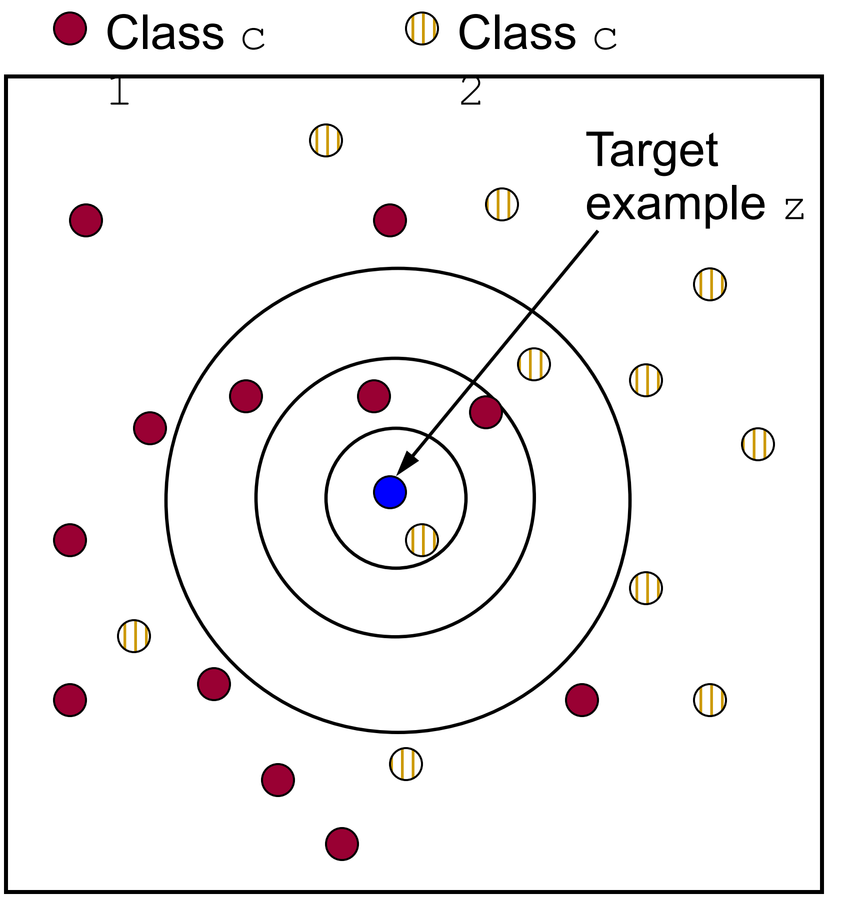
- ***K-nearest neighbors*** (k-NN) is one of the most simple methods in ML. Some other names:
  - Instance-based learning
  - Lazy learning
  - Memory-based learning
- Main ideas
  - There is no specific assumption on the function to be learned.
  - Learning phase just stores all the training data.
  - Prediction for a new instance is based on its nearest neighbors in the training data.

k-NN

- Two main ingredients :
  - The similarity measure (distance) between instances/objects.
  - The neighbors to be taken in prediction.
- Under some conditions, k-NN can achieve the Bayes optimal error which is the desired performance of any methods. [Gyuader and Hengartner, JMLR 2013]
  - Even 1-NN (with some simple modifications) can achieve this performance. [Kontorovich & Weiss, AISTATS 2015]

k-NN example: Classification problem

- Take 1 nearest neighbor?
  - → Assign z to class c2
- Take 3 nearest neighbors
  - → Assign z to class c1
- Take 5 nearest neighbors
  - → Assign z to class c1



k-NN for classification

- Data representation:
  - The description: $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, where $x_i \in R$
  - The class label: $c \in C$, where C is a pre-defined set of class labels.
- Learning phase
  - Simply save all the training data $\boldsymbol{D}$, with their labels.
- Prediction: to classify a new instance $\boldsymbol{z}$
  - For each training instance $\boldsymbol{x} \in \boldsymbol{D}$, compute the distance/similarity between $\boldsymbol{x}$ and $\boldsymbol{z}$
  - Determine a set $NB(\boldsymbol{z})$ of the nearest neighbors of $\boldsymbol{z}$
  - Using majority of the labels in $NB(\boldsymbol{z})$ to predict the label for z.

k-NN for regression

- Data representation:
  - Each observation is represented by $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, where $x_i \in R$
  - The output $y_x \in R$ is a real number.
- Learning phase
  - Simply save all the training data $\boldsymbol{D}$, with their labels
- Prediction: for a new instance $\boldsymbol{z}$
  - For each instance $\boldsymbol{x} \in \boldsymbol{D}$, compute the distance/similarity between $\boldsymbol{x}$ and $\boldsymbol{z}$.
  - Determine a set $NB(\boldsymbol{z})$ of the *k* nearest neighbors of $\boldsymbol{z}$, with
  - Predict the label for $\boldsymbol{z}$: $y_z = \frac{1}{k} \sum_{\boldsymbol{x} \in NB(\boldsymbol{z})} y_x$

k-NN: two key ingredients

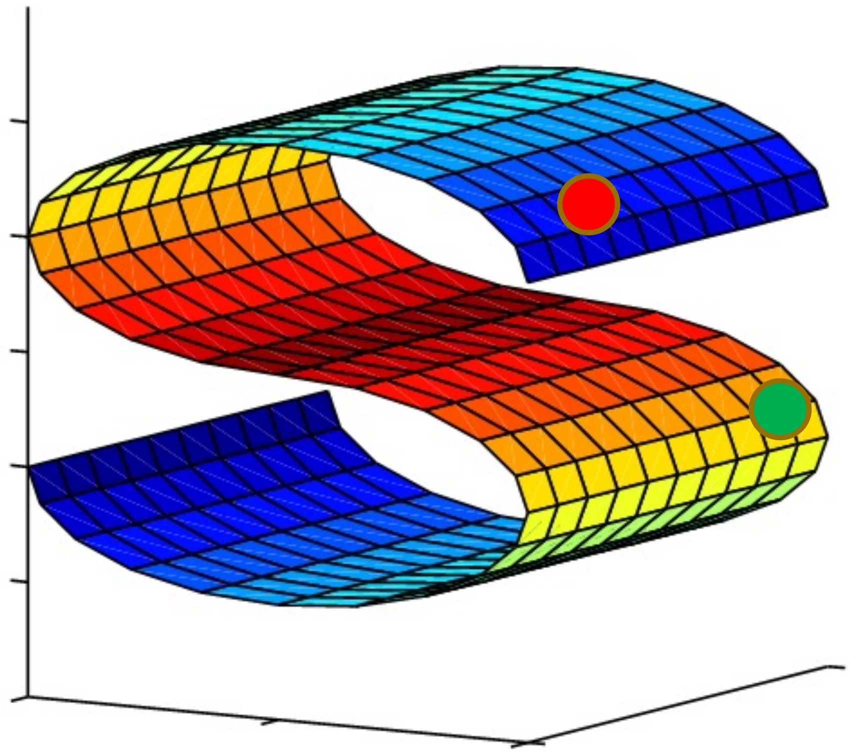
Different thoughts,

Different views

Different measures

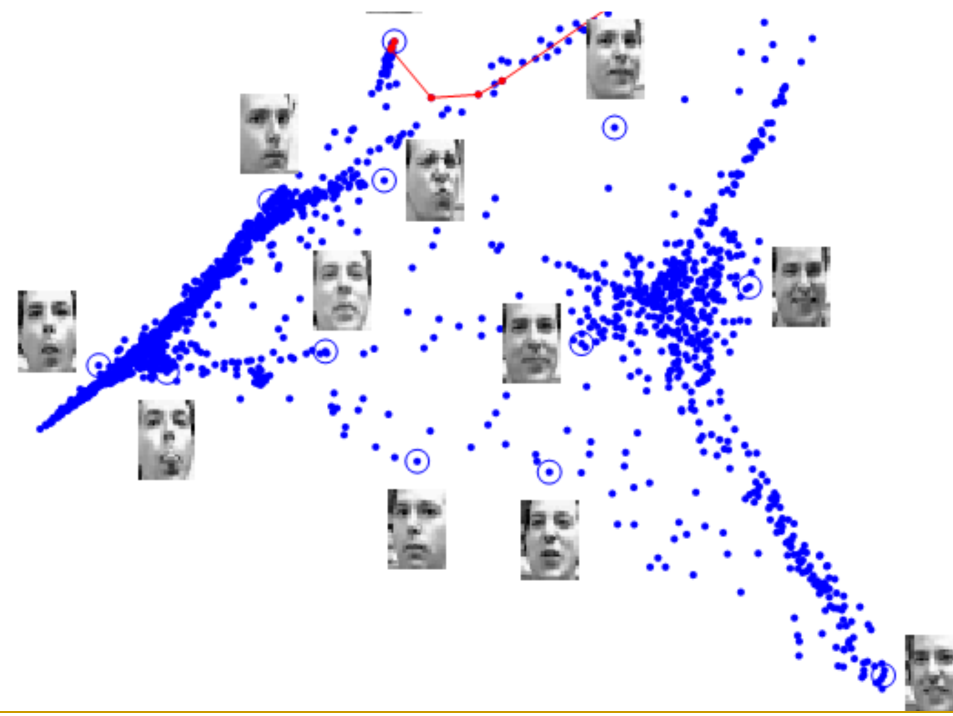
k-NN: two key ingredients

- The **distance/similarity** measure
  - Each measure corresponds to a view on data.
  - Infinitely many measures!!!
  - Which measure to use?



k-NN: two key ingredients

- The set NB(z) of nearest neighbors
  - How many neighbors are enough?
  - How can we select NB(z)? (by choosing k or restricting the area?)



k-NN: 1 or more neighbors?

- *In theory, 1-NN can be among the best methods under some conditions.*
- k-NN is Bayes optimal under some conditions: Y bounded, large training size M, and the true regression function being continuous, and

$$k \to \infty, (k/M) \to 0, (k/\log M) \to +\infty$$

- In practice, we should use more neighbors for prediction (k>1), but not too many:
    - To avoid noises/errors in only one nearest neighbor.
    - Too many neighbors might break the inherent structure of the data manifold, and thus prediction might be bad.

Distance/similarity measure (1)

- The distance measure ***d***
  - Plays a very important role in k-NN methods.
  - Be determined once, and does not change in all prediction later
- Some common distance measures ***d***
  - *Geometric distance*: usable for problems with real inputs ($x_i \in R$)
  - *Hamming distance*: usable for problems with binary inputs ($x_i \in \{0, 1\}$)

Distance/similarity measure (2)

- Some geometric distances:
  - Minkowski (p-norm): $$d(x,z)=\left(\sum_{i=1}^{n}|x_i - z_i|^p\right)^{1/p}$$
  - Manhattan ($p = 1$): $$d(x,z)=\sum_{i=1}^{n}|x_i - z_i|$$
  - Euclid ($p = 2$): $$d(x,z)=\sqrt{\sum_{i=1}^{n}(x_i - z_i)^2}$$
  - Chebyshev ($p = \infty$): $$d(x,z)=\lim_{p\to\infty}\left(\sum_{i=1}^{n}|x_i - z_i|^p\right)^{1/p} = \max_i |x_i - z_i|$$

Distance/similarity measure (3)

- Hamming distance
  - For problems with binary inputs

$$d(x,z) = \sum_{i=1}^{n} Difference(x_i, z_i)$$

$$Difference(a,b) = \begin{cases} 1, if\ (a \neq b) \\ 0, if\ (a = b) \end{cases}$$

k-NN: limitations/advantages

- Advantages
  - Low cost for the training phase (Only needs to store training examples)
  - Works well for multi-class classification problems
    - Doesn't require to learn *c* classifiers for *c* classes.
  - k-NN is able to reduce some bad effects from noises when k > 1.
    - Prediction/classification is made based on *k* nearest neighbors.
  - Very flexible in choosing the distance/similarity measure:
    - We can use similarity measure: cosine similarity
    - We can use dissimilarity measure, such as Kullback Leibler divergence, Bregman divergence.
- Limitations
  - Requires a suitable distance/similarity measure for your problem
  - Requires intensive computation at inference time.

Naïve Bayes

- A classification method based on Bayes theorem
- Using a probability model (function)
- Classification based on the probability values of possible outcomes of the hypotheses

Bayes theorem

$$P(h \mid D) = \frac{P(D \mid h).P(h)}{P(D)}$$

- `P(h)` : **Prior** probability of hypothesis `h`
- `P(D)` : Prior probability that the data `D` is observed
- `P(D|h)` : (Conditional) probability of observing the data `D` given hypothesis `h`. (**likelihood**)
- `P(h|D)` : (**Posterior**) probability of hypothesis `h` given the observed data `D`
  - ➢ **Probabilistic classification methods use this posterior probability!**

Bayes theorem – Example (1)

Assume that we have the following data (of a person):

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes

[Mitchell, 1997]

Bayes theorem – Example (2)

- Dataset `D`. The data of the days when the outlook is sunny and the wind is strong
- Hypothesis `h`. The person plays tennis
- Prior probability `P(h)`. Probability that the person plays tennis (i.e., regardless of the outlook and the wind)
- Prior probability `P(D)`. Probability that the outlook is sunny and the wind is strong
- `P(D|h)`. Probability that the outlook is sunny and the wind is strong, given knowing that the person plays tennis
- `P(h|D)`. Probability that the person plays tennis, given knowing that the outlook is sunny and the wind is strong

Maximum a posteriori (MAP)

- Given a set H of possible hypotheses (e.g., possible classifications), the learner finds the most probable hypothesis `h(∈H)` given the observed data `D`

- Such a maximally probable hypothesis is called a **maximum a posteriori (MAP)** hypothesis

$$h_{MAP} = \arg\max_{h \in H} P(h \mid D)$$

$$h_{MAP} = \arg\max_{h \in H} \frac{P(D \mid h).P(h)}{P(D)}$$ (by Bayes theorem)

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h).P(h)$$ (`P(D)` is a constant, independent of `h`)

MAP: Example

- The set H contains two hypotheses
  - h_1: The person will play tennis
  - h_2: The person will not play tennis
- Compute the two posteriori probabilities: $P(h_1|D)$, $P(h_2|D)$
- The MAP hypothesis: $h_{MAP}=h_1$ if $P(h_1|D) \geq P(h_2|D)$; otherwise $h_{MAP}=h_2$
- So, we compute the two formulae: $P(D|h_1).P(h_1)$ and $P(D|h_2).P(h_2)$, and make the conclusion:
  - Dếu $P(D|h_1).P(h_1) \geq P(D|h_2).P(h_2)$, the person will play tennis
  - Otherwise, the person will not play tennis

Maximum likelihood estimation (MLE)

- MAP: Given a set of possible hypotheses H, find a hypothesis that maximizes the probability: $P(D|h).P(h)$
- Assumption of **Maximum likelihood estimation – MLE** method: All hypotheses have the same prior probability: $P(h_i)=P(h_j), \forall h_i, h_j \in H$
- MLE method finds a hypothesis that maximizes the probability $P(D|h)$, where $P(D|h)$ is called *likelihood* of the data D given hypothesis h
- Maximum likelihood hypothesis:

$$h_{ML} = \arg\max_{h \in H} P(D|h)$$

MLE: Example

- The set H contains two hypotheses
  - h_1: The person will play tennis
  - h_2: The person will not play tennis

  D: The data of the dates when the outlook is sunny and the wind is strong

- Compute the two likelihood values of the data D given the two hypotheses: $P(D|h_1)$ và $P(D|h_2)$
  - $P(Outlook=Sunny, Wind=Weak|h_1)$ = 1/8
  - $P(Outlook=Sunny, Wind=Weak|h_2)$ = 1/4

- The MLE hypothesis $h_{MLE}=h_1$ nếu $P(D|h_1) \geq P(D|h_2)$; otherwise $h_{MLE}=h_2$

  → Because $P(Outlook=Sunny, Wind=Weak|h_1) <$ $P(Outlook=Sunny, Wind=Weak|h_2)$, we arrive at the conclusion: The person will not play tennis

Naïve Bayes classifier (1)

- Classification problem
  - A training set D, where each training instance x is represented as an n-dimensional attribute vector: $(x_1, x_2, ..., x_n)$
  - A pre-defined set of classes: $C=\{c_1, c_2, ..., c_m\}$
  - Given a new instance z, which class should z be classified to?
- We want to find the most probable class for instance z

$$c_{MAP} = \arg\max_{c_i \in C} P(c_i \mid z)$$

$$c_{MAP} = \arg\max_{c_i \in C} P(c_i \mid z_1, z_2, ..., z_n)$$

$$c_{MAP} = \arg\max_{c_i \in C} \frac{P(z_1, z_2, ..., z_n \mid c_i).P(c_i)}{P(z_1, z_2, ..., z_n)}$$ (by Bayes theorem)

Naïve Bayes classifier (2)

- To find the most probable class for z…

$$c_{MAP} = \arg\max_{c_i \in C} P(z_1, z_2, \ldots, z_n \mid c_i).P(c_i)$$

(P(z1, z2, ..., zn) is the same for all classes)

- **Assumption in Naïve Bayes classifier.** The attributes are *conditionally independent* given class labels

$$P(z_1, z_2, \ldots, z_n \mid c_i) = \prod_{j=1}^{n} P(z_j \mid c_i)$$

- Naïve Bayes classifier finds the most probable class for z

$$c_{NB} = \arg\max_{c_i \in C} P(c_i).\prod_{j=1}^{n} P(z_j \mid c_i)$$

Naïve Bayes classifier: Algorithm

- The learning (training) phase (given a training set)

  For each class $c_i \in C$

  - Estimate the priori probability: $P(c_i)$
  - For each attribute value x_j, estimate the probability of that attribute value given class c_i: $P(x_j | c_i)$

- The classification phase (given a new instance)
  - For each class $c_i \in C$, compute:

$$P(c_i).\prod_{j=1}^{n} P(x_j | c_i)$$

  - Select the most probable class c^* by

$$c^* = \arg\max_{c_i \in C} P(c_i).\prod_{j=1}^{n} P(x_j | c_i)$$

Naïve Bayes classifier: Example (1)

Will a young student with medium income and fair credit rating buy a computer?

Rec. ID	Age	Income	Student	Credit_Rating	Buy_Computer
1	Young	High	No	Fair	No
2	Young	High	No	Excellent	No
3	Medium	High	No	Fair	Yes
4	Old	Medium	No	Fair	Yes
5	Old	Low	Yes	Fair	Yes
6	Old	Low	Yes	Excellent	No
7	Medium	Low	Yes	Excellent	Yes
8	Young	Medium	No	Fair	No
9	Young	Low	Yes	Fair	Yes
10	Old	Medium	Yes	Fair	Yes
11	Young	Medium	Yes	Excellent	Yes
12	Medium	Medium	No	Excellent	Yes
13	Medium	High	Yes	Fair	Yes
14	Old	Medium	No	Excellent	No

http://www.cs.sunysb.edu/~cse634/lecture_notes/07classification.pdf

Naïve Bayes classifier: Example (2)

- Representation of the problem
  - z = (Age=Young, Income=Medium, Student=Yes, Credit_Rating=Fair)
  - Two classes:: c_1 (buy a computer) and c_2 (not buy a computer)
- Compute the priori probability for each class
  - $P(c_1) = 9/14$
  - $P(c_2) = 5/14$
- Compute the probability of each attribute value given each class
  - P(Age=Young|c_1) = 2/9; P(Age=Young|c_2) = 3/5
  - P(Income=Medium|c_1) = 4/9; P(Income=Medium|c_2) = 2/5
  - P(Student=Yes|c_1) = 6/9; P(Student=Yes|c_2) = 1/5
  - P(Credit_Rating=Fair|c_1) = 6/9; P(Credit_Rating=Fair|c_2) = 2/5

Naïve Bayes classifier: Example (3)

- Compute the likelihood of instance x given each class
  - For class c_1

    $P(z|c_1) = P(Age=Young|c_1).P(Income=Medium|c_1).P(Student=Yes|c_1).P(Credit\_Rating=Fair|c_1) = (2/9).(4/9).(6/9).(6/9) = 0.044$

  - For class c_2

    $P(z|c_2) = P(Age=Young|c_2).P(Income=Medium|c_2).P(Student=Yes|c_2).P(Credit\_Rating=Fair|c_2) = (3/5).(2/5).(1/5).(2/5) = 0.019$

- Find the most probable class
  - For class c_1

    $P(c_1).P(z|c_1) = (9/14).(0.044) = 0.028$

  - For class c_2

    $P(c_2).P(z|c_2) = (5/14).(0.019) = 0.007$

  $\rightarrow$Conclusion: The person z will buy a computer!

Naïve Bayes classifier: Issues (1)

- If there is no example belonging to class c_i with attribute x_j

  $P(x_j|c_i) = 0$, and thus: $$P(c_i).\prod_{j=1}^{n} P(x_j|c_i) = 0$$

- Solution: Use Bayes theorem to approximate $P(x_j|c_i)$

$$P(x_j|c_i) = \frac{n(c_i, x_j) + mp}{n(c_i) + m}$$

  - $n(c_i)$: number of examples belonging to class c_i
  - $n(c_i, x_j)$: number of examples belonging to class c_i with attribute x_j
  - p: approximation of $P(x_j|c_i)$

    $\rightarrow$ Uniform approximation $p=1/k$, if attribute f_j has k values
  - m: a constant

    $\rightarrow$ To complement $n(c_i)$ the number of observations with an additional m examples with an approximate probability p

Naïve Bayes classifier: Issues (2)

- Limitation in the precision of computer
  - $P(x_j|c_i)<1$, for all attribute x_j and class c_i
  - Hence, when the number of attributes becomes too large:

$$\lim_{n\to\infty}\left(\prod_{j=1}^{n} P(x_j \mid c_i)\right) = 0$$

- Solution: apply logarithmic function to the probability

$$c_{NB} = \arg\max_{c_i \in C}\left(\log\left[P(c_i).\prod_{j=1}^{n} P(x_j \mid c_i)\right]\right)$$

$$c_{NB} = \arg\max_{c_i \in C}\left(\log P(c_i) + \sum_{j=1}^{n} \log P(x_j \mid c_i)\right)$$

Document classification using NB (1)

- Problem definition
  - A training set D, where each training example is a document associated with a class label: $D = \{(d_k, c_i)\}$
  - A pre-defined set of class labels: $C = \{c_i\}$
- Training phase
  - From the document set D, extract the set of distinct terms T
  - Let D_{c_i} be the set of document in D with class label c_i
  - For each class label c_i
    - Compute the priori probability of class c_i: $P(c_i) = \dfrac{|D_{c_i}|}{|D|}$
    - For each term t_j, compute the probability of term t_j given class label c_i

$$P(t_j \mid c_i) = \frac{\left(\sum_{d_k \in D_{c_i}} n(d_k, t_j)\right) + 1}{\left(\sum_{d_k \in D_{c_i}} \sum_{t_m \in T} n(d_k, t_m)\right) + |T|}$$

($n(d_k, t_j)$: the number of occurrences of term t_j in document d_k)

Document classification using NB (2)

- Classification phase: for a new document d
  - From document d, extract the set T_d consists of terms (keywords) t_j defined in the set T
  - **Assumption:** The probability of term t_j given class c_i is independent of its position in the document

    $P(t_j \text{ at position } k \mid c_i) = P(t_j \text{ at position } m \mid c_i), \ \forall k,m$
  - For each class c_i, compute the posterior probability of document d given c_i

$$P(c_i).\prod_{t_j \in T_d} P(t_j \mid c_i)$$

  - Classify document d in class c^*

$$c^* = \arg\max_{c_i \in C} P(c_i).\prod_{t_j \in T_d} P(t_j \mid c_i)$$

References

- E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2010.
- T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- T. M. Mitchell. *The discipline of machine learning*. CMU technical report, 2006.
- H. A. Simon. *Why Should Machines Learn?* In R. S. Michalski, J. Carbonell, and T. M. Mitchell (Eds.): Machine learning: An artificial intelligence approach, chapter 2, pp. 25-38. Morgan Kaufmann, 1983.
- A. Kontorovich and Weiss. *A Bayes consistent 1-DD classifier*. Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS). JMLR: W&CP volume 38, 2015.
- A. Guyader, D. Hengartner. *On the Mutual Dearest Deighbors Estimate in Regression*. Journal of Machine Learning Research 14 (2013) 2361-2376.
- L. Gottlieb, A. Kontorovich, and P. Disnevitch. *Dear-optimal sample compression for nearest neighbors*. Advances in Deural Information Processing Systems, 2014.