

# Machine Learning

(Học máy – IT3190E)

**Khoat Than**

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

2022

2

## Contents

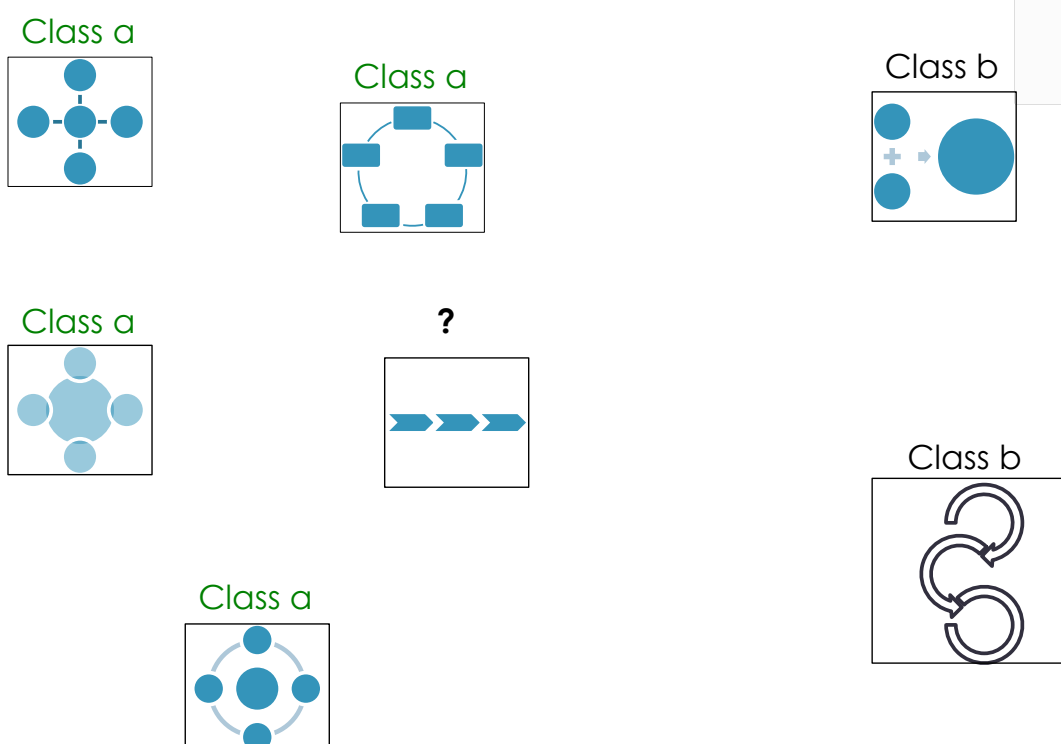
---

- Introduction to Machine Learning
- **Supervised learning**
  - **K-nearest neighbors**
- Unsupervised learning
- Reinforcement learning
- Practical advice

## Classification problem

- **Supervised learning:** learn a function  $y = f(x)$  from a given training set  $\{\{x_1, x_2, \dots, x_N\}; \{y_1, y_2, \dots, y_N\}\}$  so that  $y_i \cong f(x_i)$  for every  $i$ .
  - Each training instance has a label/response.
- **Multiclass classification/categorization:** output  $y$  is only one from a pre-defined set of labels
  - $y$  in {normal, spam}
  - $y$  in {fake, real}

## Which class does the object belong to?



## Neighbor-based learning (1)

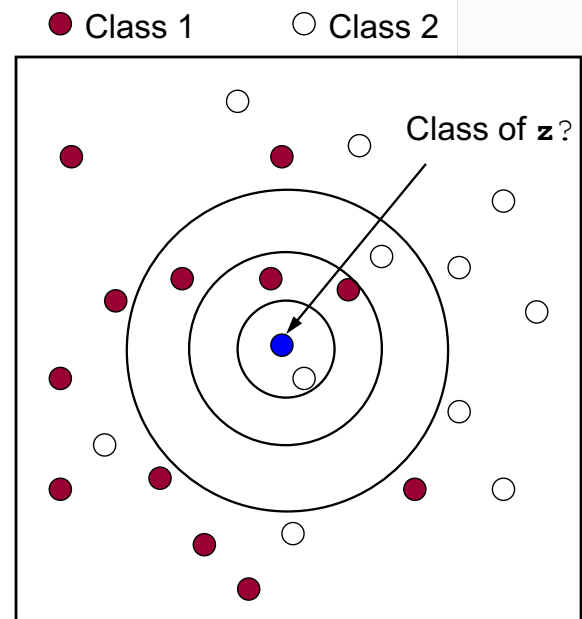
- *K-nearest neighbors* (KNN) is one of the most simple methods in ML. Some other names:
  - Instance-based learning
  - Lazy learning
  - Memory-based learning
- *Main ideas:*
  - There is no specific assumption on the function to be learned.
  - Learning phase just stores all the training data.
  -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

## Neighbor-based learning (2)

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

## KNN: example

- Take 1 nearest neighbor?
  - Assign  $\mathbf{z}$  to class 2.
- Take 3 nearest neighbors?
  - Assign  $\mathbf{z}$  to class 1.
- Take 5 nearest neighbors?
  - Assign  $\mathbf{z}$  to class 1.



## KNN for classification

- Data representation:
  - Each observation is represented by a vector in an  $n$ -dimensional space, e.g.,  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$ . Each dimension represents an attribute/feature/variable.
  - There is a set  $C$  of predefined labels.
- Learning phase:
  - Simply save all the training data  $\mathbf{D}$ , with their labels.
- Prediction: for a new instance  $\mathbf{z}$ .
  - For each instance  $\mathbf{x}$  in  $\mathbf{D}$ , compute the distance/similarity between  $\mathbf{x}$  and  $\mathbf{z}$ .
  - Determine a set  $NB(\mathbf{z})$  of the nearest neighbors of  $\mathbf{z}$ .
  - Using majority of the labels in  $NB(\mathbf{z})$  to predict the label for  $\mathbf{z}$ .

## KNN for regression

### ■ Data representation:

- Each observation is represented by a vector in an  $n$ -dimensional space, e.g.,  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$ . Each dimension represents an attribute/feature/variable.
- The output  $y$  is a real number.

### ■ Learning phase:

- Simply save all the training data  $\mathbf{D}$ , with their labels.

### ■ Prediction: for a new instance $\mathbf{z}$ .

- For each instance  $\mathbf{x}$  in  $\mathbf{D}$ , compute the distance/similarity between  $\mathbf{x}$  and  $\mathbf{z}$ .
- Determine a set  $NB(\mathbf{z})$  of the nearest neighbors of  $\mathbf{z}$ , with  $|NB(\mathbf{z})| = k$ .
- Predict the label for  $\mathbf{z}$  by  $y_z = \frac{1}{k} \sum_{x \in NB(\mathbf{z})} y_x$

## KNN: two key ingredients (1)



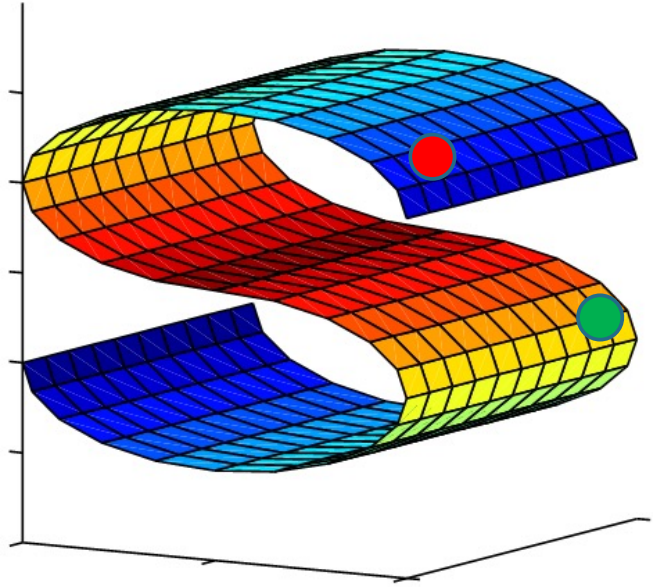
Different  
thoughts,

Different  
views

Different  
measures

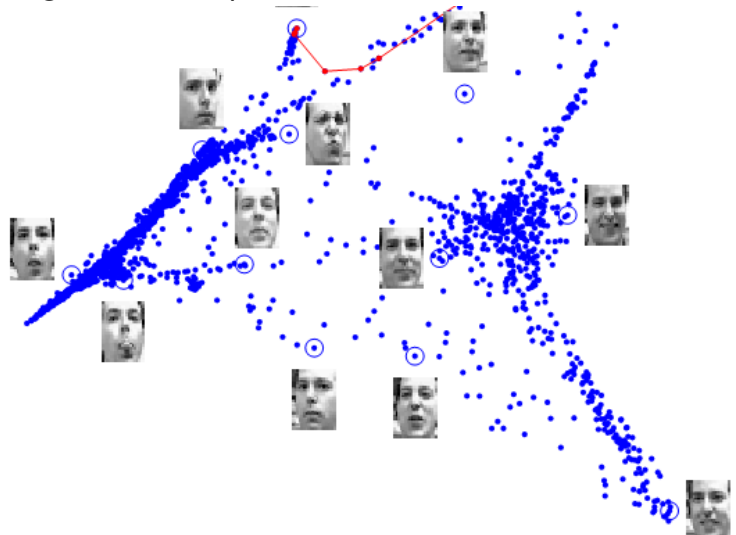
## KNN: two key ingredients (2)

- The **distance/similarity** measure
  - Each measure implies a view on data
  - Infinite many measures !!!
  - What measure should be?



## KNN: two key ingredients (3)

- The set  $NB(\mathbf{z})$  of **nearest neighbors**.
  - How many neighbors are enough?
  - How can we select  $NB(\mathbf{z})$ ?  
(by choosing  $k$  or restricting the area?)



## KNN: 1 or more neighbors?

- In theory, *1-NN can be among the best methods under some conditions*. [Kontorovich & Weiss, AISTATS 2015]
  - KNN is Bayes optimal under some conditions:  $Y$  bounded, large training size  $M$ , and the true regression function being continuous, and
 
$$k \rightarrow \infty, (k/M) \rightarrow 0, (k/\log M) \rightarrow +\infty$$
- 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:
  - Plays a very important role in KNN.
  - Indicates *how we assume/suppose the distribution of our data*.
  - Be determined once, and does not change in all prediction later.
- Some common distance measures:
  - *Geometric distance*: usable for problems with real inputs
  - *Hamming distance*: usable for problems with binary inputs, such as  $x$  in  $\{0; 1\}$

## Distance/similarity measure (2)

### ■ Some geometric distances:

- Minkowski ( $L_p$ -norm): 
$$d(x, z) = \left( \sum_{i=1}^n |x_i - z_i|^p \right)^{1/p}$$
- Manhattan ( $L_1$ -norm): 
$$d(x, z) = \sum_{i=1}^n |x_i - z_i|$$
- Euclid ( $L_2$ -norm): 
$$d(x, z) = \sqrt{\sum_{i=1}^n (x_i - z_i)^2}$$
- Chebyshev (max norm): 
$$d(x, z) = \lim_{p \rightarrow \infty} \left( \sum_{i=1}^n |x_i - z_i|^p \right)^{1/p}$$
  

$$= \max_i |x_i - z_i|$$

## Distance/similarity measure (3)

### ■ Hamming distance:

- for problems with binary inputs
- such as  $\mathbf{x} = (1, 0, 0, 1, 1)$

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

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

### ■ Cosine measure:

- Suitable for some problems with textual inputs.

$$d(x, z) = \frac{x^T z}{\|x\| \cdot \|z\|}$$



## KNN: attribute normalization

- Normalizing the attributes is sometimes important to get good predictiveness in KNN.
  - No normalization implies that the magnitude of an attribute might play a heavy role, and artificially overwhelms the other attributes. Ex.:

$$d(x, z) = \sqrt{\sum_{i=1}^n (x_i - z_i)^2}$$

- $x = (\text{Age}=20, \text{Income}=12000, \text{Height}=1.68)$
  - $z = (\text{Age}=40, \text{Income}=1300, \text{Height}=1.75)$
  - $d(x, z) = [(20-40)^2 + (12000-1300)^2 + (1.68-1.75)^2]^{0.5}$
- This is unrealistic and unexpected in some applications.
- Some common normalizations:
  - Make all values of  $x_j$  in  $[-1; 1]$ ;
  - Make all values of  $x_j$  to have empirical mean 0 and variance 1.

## KNN: attribute weighting

- Weighting the attributes is sometimes important for KNN.
  - No weight implies that the attributes play an equal role, e.g., due to the use of the Euclidean distance:

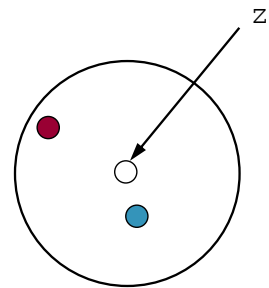
$$d(x, z) = \sqrt{\sum_{i=1}^n (x_i - z_i)^2} \quad \longrightarrow \quad d(x, z) = \sqrt{\sum_{i=1}^n w_i (x_i - z_i)^2}$$

- This is unrealistic in some applications, where an attribute might be more important than the others in prediction.
  - Some weights ( $w_i$ ) on the attributes might be more suitable.
- How to decide the weights?
  - Base on the knowledge domain about your problem.
  - Learn the weights automatically from the training data.

## KNN: weighting neighbors (1)

$$y_z = \frac{1}{k} \sum_{x \in NB(z)} y_x$$

- Prediction of labels miss some information about neighbors.
  - The neighbors in  $NB(\mathbf{z})$  play the same role with respect to the different distances to the new instance.
  - This is unrealistic in some applications, where *closer neighbors should play more important role than the others*.
- Using the distance as weights in prediction might help.
  - Closer neighbors should have more effects.
  - Farther points should have less effects.



## KNN: weighting neighbors (2)

- Let  $v$  be the weights to be used.
  - $v(\mathbf{x}, \mathbf{z})$  can be chosen as the inverse of the distance from  $\mathbf{x}$  to  $\mathbf{z}$ ,  $d(\mathbf{x}, \mathbf{z})$ .

- Some examples:

$$v(x, z) = \frac{1}{\alpha + d(x, z)} \quad v(x, z) = \frac{1}{\alpha + [d(x, z)]^2} \quad v(x, z) = e^{-\frac{d(x, z)^2}{\sigma^2}}$$

- For classification:

$$c_z = \arg \max_{c_j \in C} \sum_{x \in NB(z)} v(x, z) \cdot \text{Identical}(c_j, c_x) \quad \text{Identical}(a, b) = \begin{cases} 1, & \text{if } (a = b) \\ 0, & \text{if } (a \neq b) \end{cases}$$

- For regression:

$$y_z = \frac{\sum_{x \in NB(z)} v(x, z) \cdot y_x}{\sum_{x \in NB(z)} v(x, z)}$$

## KNN: limitations/advantages

### ■ Advantages:

- Low cost for the training phase.
- Very flexible in choosing the distance/similarity measure: we can use many other measures, such as Kullback-Leibler divergence, Bregman divergence,...
- KNN is able to reduce some bad effects from noises when  $k > 1$ .
- In theory, KNN can reach the best performance among all regression methods, under some conditions.  
(this might not be true for other methods)

### ■ Limitations:

- Have to find a suitable distance/similarity measure for your problem.
- Prediction requires intensive computation.

## References

- A. Kontorovich and Weiss. A Bayes consistent 1-NN classifier. *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR: W&CP volume 38, 2015.
- A. Guyader, N. Hengartner. On the Mutual Nearest Neighbors Estimate in Regression. *Journal of Machine Learning Research* 14 (2013) 2361-2376.
- L. Gottlieb, A. Kontorovich, and P. Nisnevitch. Near-optimal sample compression for nearest neighbors. *Advances in Neural Information Processing Systems*, 2014.

## Exercises

---

- What is the different between KNN and OLS?
- Is KNN prone to overfitting?
- How to make KNN work with sequence data?  
(each instance is a sequence)