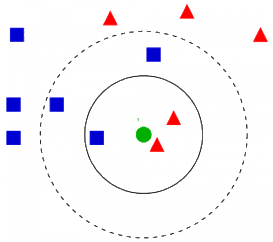
DATA SCIENCE

COMP2200/6200

08 – K-Nearest Neighbors Classifier



Xuyun Zhang (Department of Computing)

Part II Scheduling

MACQUARIE University

COMP2200/6200 Part II

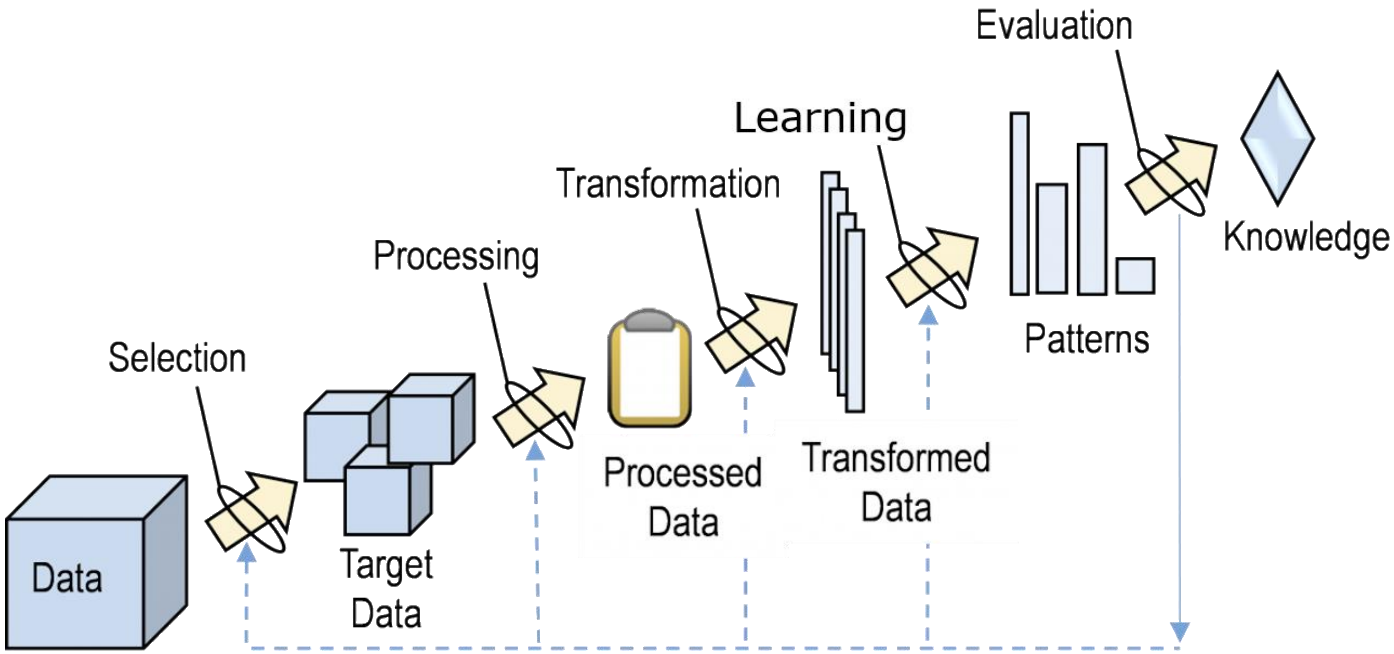
- K-Nearest Neighbors Classifier
- Naïve Bayes Classifier
- Artificial Neural Networks
- Decision Tree Classifier
- Guest Lecture

Lecture Outline

- Machine Learning for Data Science
- K-Nearest Neighbors Classifier
- Model Selection

Data Mining and ML

- Data Mining ≈ Database + Machine Learning
  - Database: data management and pre-processing
  - Machine learning: data analysis & knowledge discovery



What is Learning?

- Given
  - **Task** T
  - **Performance** P of a computer program on task T
  - **Experience** E
- If a computer program gains performance improvement on T by leveraging experience E, it can be said that the computer program learns from E with respect to T and P. [Mitchell, 1997]
- Experience is often embedded in the form of data
- Compared to no learning

Learning Example

- Task: predict if loan requests can be approved or not
- Performance: accuracy
- No learning: to randomly approve a future request
  - Accuracy (expectation): 50% (training accuracy)
- Experience (data): loan application data

ID	Age	Has_job	Own_house	Approved
1	young	yes	no	yes
2	middle	yes	yes	yes
3	old	no	yes	no

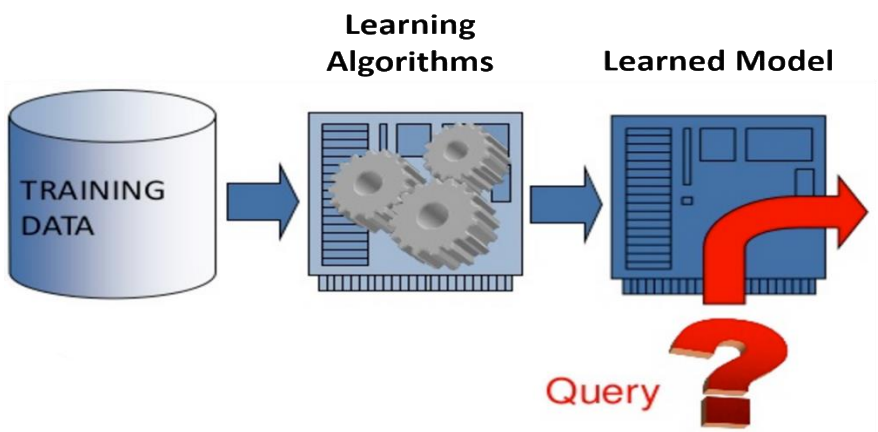
- A simple learning: approve a future request with 'yes'
  - Accuracy: 66.7% (training accuracy) > 50%

What Is Machine Learning?

- A subset of artificial intelligence in the field of computer science
  - Gives computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data
  - Without being explicitly programmed
- Ingredients
  - **Model**: to represent the form of learning result
  - **Learning algorithm**: how to generate a model from data
  - **Data**: input to ML algorithms
- Aims
  - **to predict unseen data** or **to interpret existing data**

Two Stages of Machine Learning

- Training stage
  - Training: to generate a model from observed data
- Testing stage
  - Testing: to use the learned model to predict unseen data



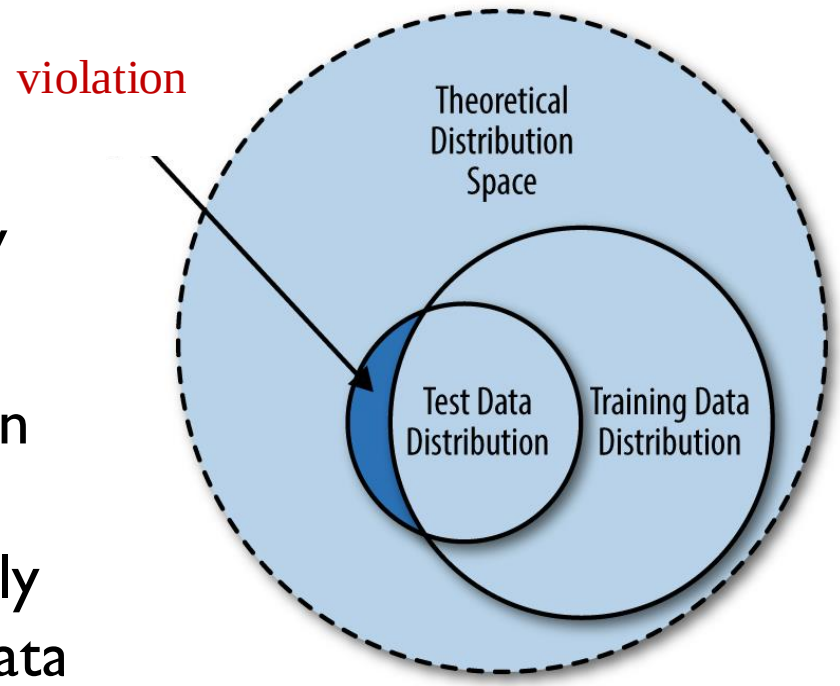
E.g., does a patient with coughing, running nose and fever suffer from COVID-19 or flu?

Data

- **Data set** D: a set of observed data instances
  - $D = \{d_1, d_2, \cdots, d_{|D|}\}$
  - Many types: numerical vectors, an image, a graph, …
  - Assumption: i.i.d. (independent & identical distributed)
    - Instances in D follow a (unknown) distribution, from which each instance is independently sampled
- **Training data**: data used in the training stage
- **Testing data**: data used in the testing stage
- **Validation data**: used to select learning models
  - Part of training data

Data (Cont'd)

- Assumption: the distribution of training examples is identical to the distribution of test examples (including future unseen examples)
  - In practice, this is often violated to certain degree
  - Strong violations will clearly result in poor performance
  - To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data

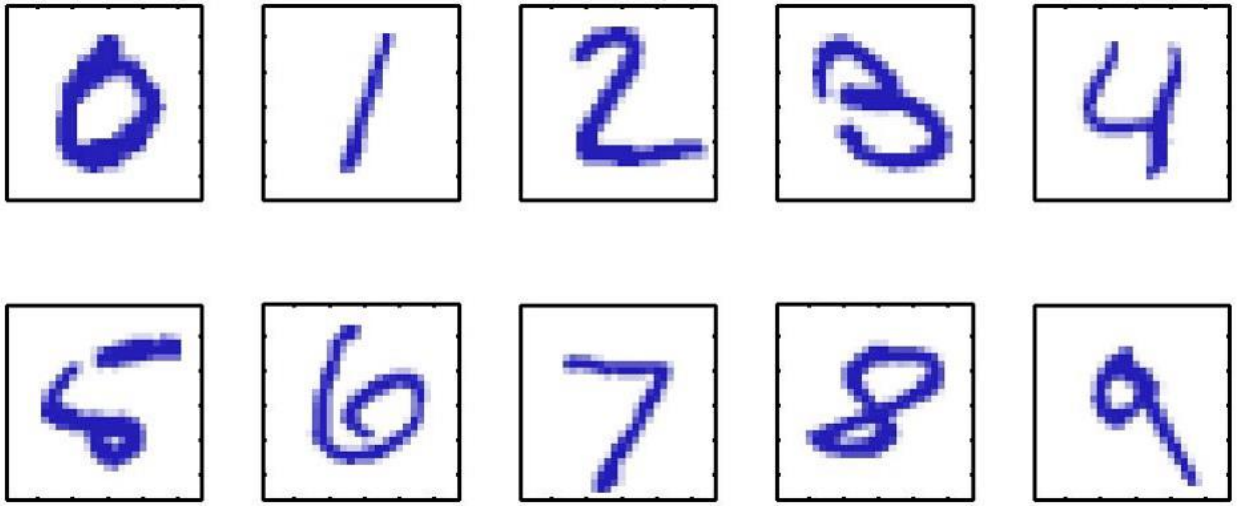


Data (Cont'd)

- Label/target of data
  - The interesting attribute(s) for prediction
  - Further, $d_i \equiv \langle x_i, y_i \rangle$
  - **Supervised learning models**
    - **Regression**: y is continuous
    - **Classification**: y is discrete (binary or multi-class)
- No explicit label/target information
  - Then, $d_i \equiv \langle x_i, \cdot \rangle$
  - **Unsupervised learning models**
    - **Clustering**
- **Semi-supervised learning** (labeling is often costly)

Data Examples

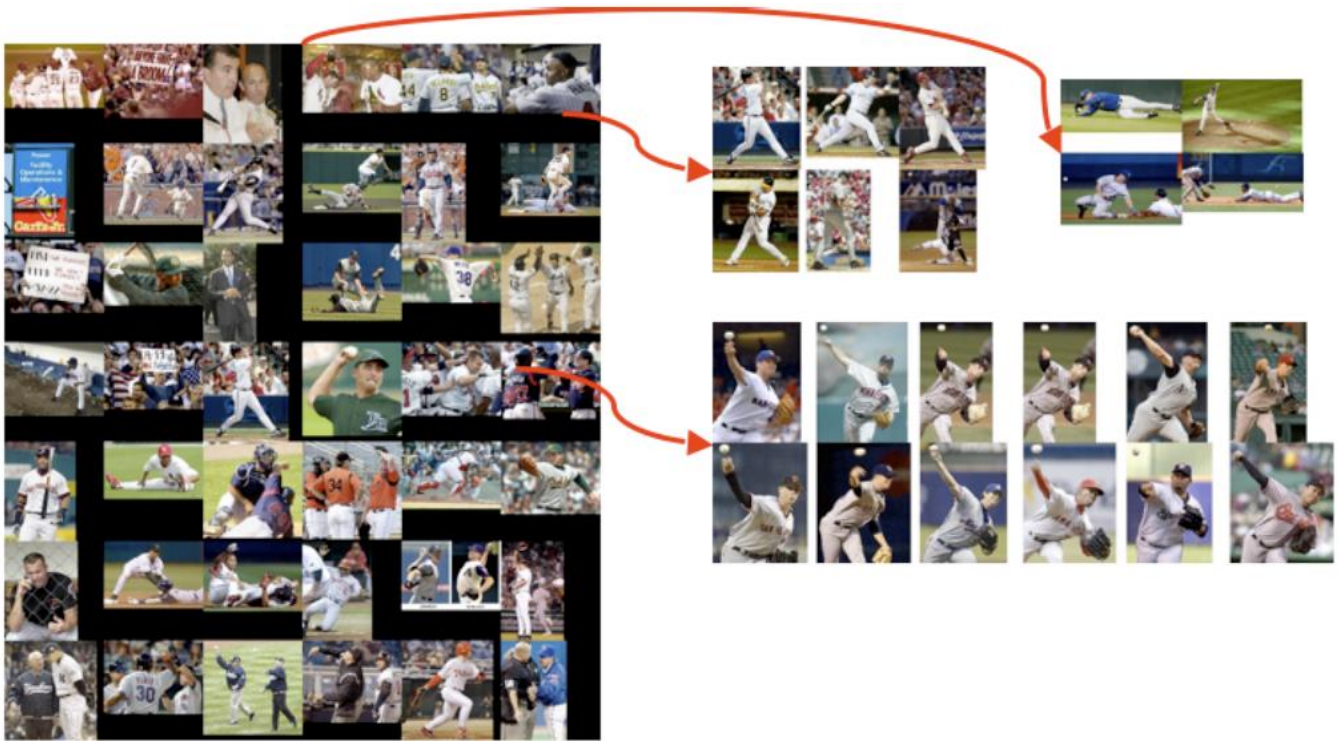
- MINST data: handwritten digit recognition
  - http://yann.lecun.com/exdb/mnist/



Data Examples (Cont'd)

- Represent input image as a vector $x_i \in \mathbb{R}^{784}$
  - Feature extraction, a pre-processing step
  -  : $x_i = \langle 0, 0, \cdots, 253, 255, \cdots, 0 \rangle$
- Label/target vector t_i
  - $t_i \in \{0, 1\}^{10}$, a discrete, finite label set
  - $t_i = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0)$
- Learning model $y : \mathbb{R}^{784} \to \mathbb{R}^{10}$
  - Dataset: $\mathcal{D} = \{\langle x_1, t_1 \rangle, \cdots, \langle x_N, t_N \rangle\}$
  - Supervised learning
  - Classification problem

Data Examples (Cont'd)



- Finding similar images, image clustering, etc.
  - Unsupervised learning
  - Only $\mathcal{D} = \{x_i\}$ is available, no label information

Model

- **Model**: a map from input space to output space
  - i.e., $f : \mathcal{X} \to \mathcal{Y}$
  - An infinite number of such maps
  - Input space: space spanned by feature attributes
  - Output space: space spanned by label/target attributes
- **Hypothesis space** H: space of all possible maps
  - Functional space: $\mathcal{H} \equiv \{f \mid \mathcal{X} \to \mathcal{Y}\}$
- **Ground truth**: underlying true mechanisms of generating the observed data
  - But never known in reality
  - The purpose of learning: to approximate the ground truth

Parameter Space

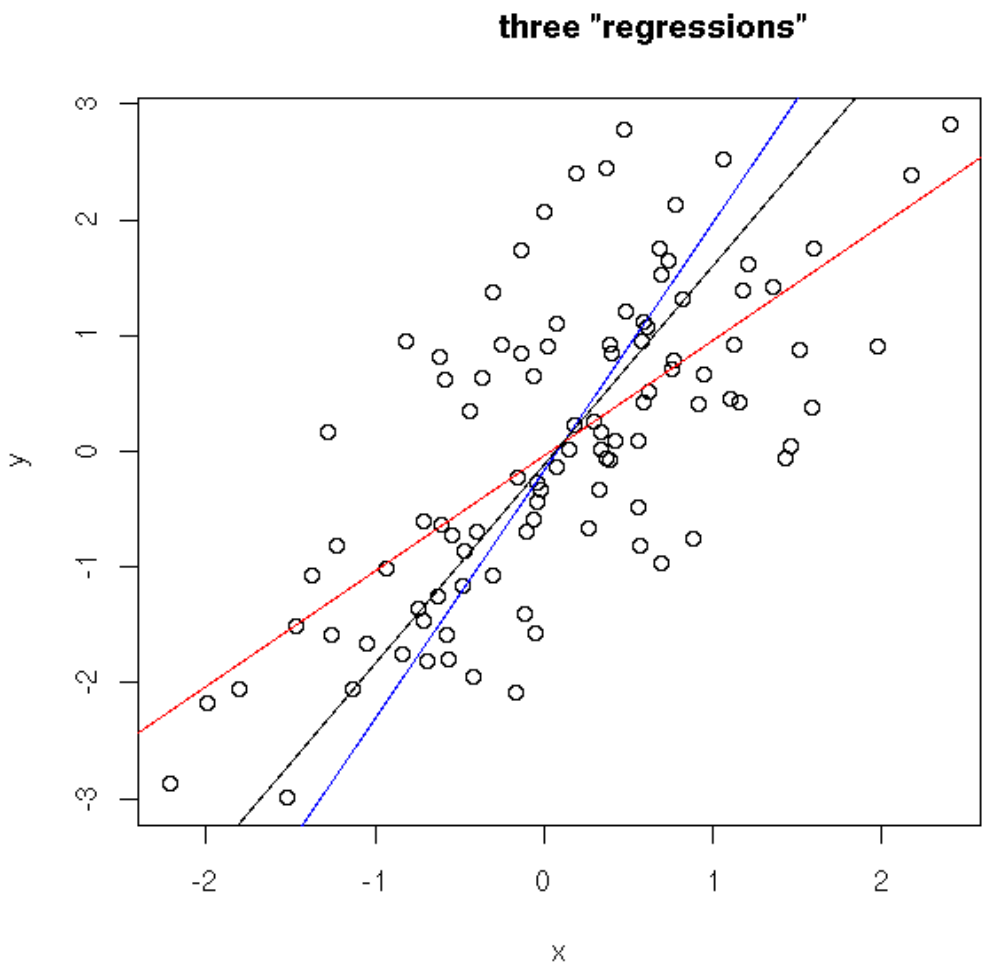
- What do we really learn from data for a model?
  - Hypothesis space is usually pre-specified in terms of problem domains
  - Different models are determined by parameters
  - Let θ denote the parameter vector, we have

$$\mathcal{H} \equiv \{f \mid y = f_\theta(x)\}$$

- Parameter space
  - Can be real value spaces, e.g., $\mathbb{R}^n$
  - Structure of a model (more implicit), e.g., tree or graph structures, as well as partitions of the input space

Parameter Space (Cont'd)

- E.g., linear regression models $y = w_0 + w_1 x$



Lecture Outline

- Machine Learning for Data Science

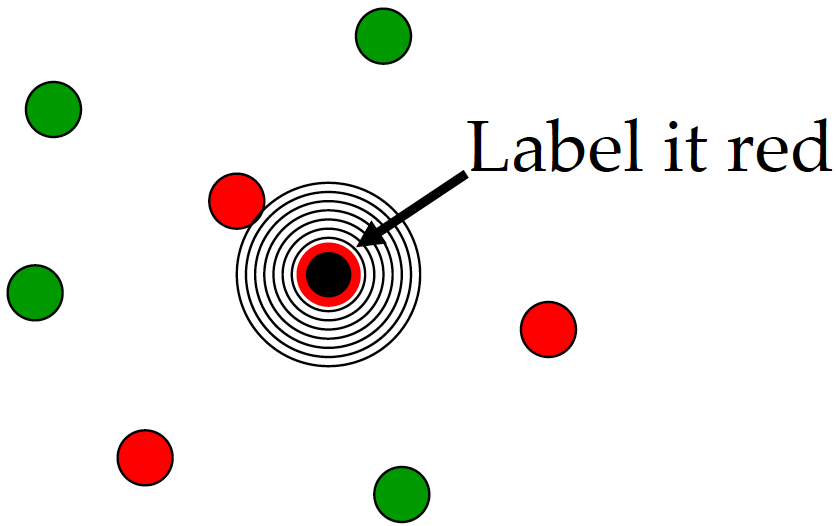
- K-Nearest Neighbors Classifier

Nearest Neighbors Classification

- Idea: instance-based learning
  - Similar examples have similar labels
  - Classify new examples like similar training examples
- Algorithm
  - Given some new example $\boldsymbol{x}$ for which we need to predict its class y
  - Find the most similar training examples
  - Classify $\boldsymbol{x}$ "like" these most similar examples
- Questions:
  - How to determine similarity?
  - How many similar training examples to consider?

1-Nearest Neighbor

- One of the simplest classifiers
- Basic idea: label a new point the same as the closest known data instance
- E.g.,

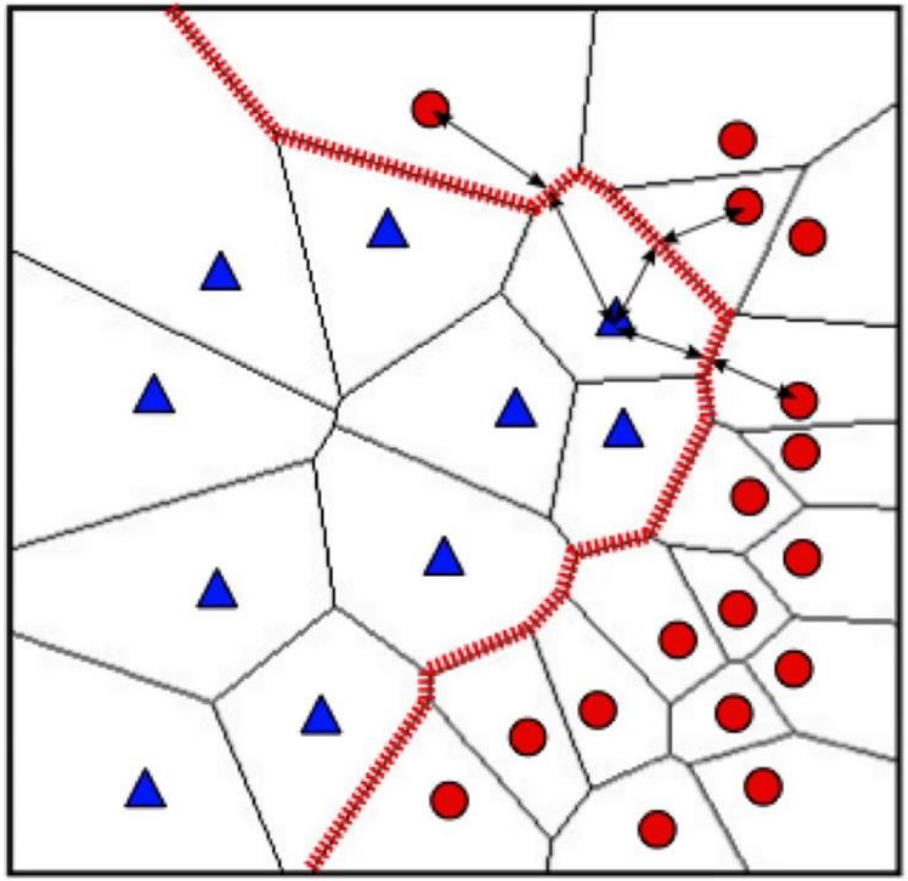


1-Nearest Neighbor (Cont'd)

- A distance metric (to measure similarity)
  - Euclidean distance is commonly-used
  - When different units are used for each dimension
    - Standardization can apply
  - For categorical data, we can use hamming distance
    - $d(\boldsymbol{x}_1, \boldsymbol{x}_2) =$ number of features on which $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ differ
  - Others (e.g., cosine, Manhattan)
- How many nearby neighbors to look at?
  - Only one
- How to fit with training data instances?
  - Just predict the same output as the nearest neighbor

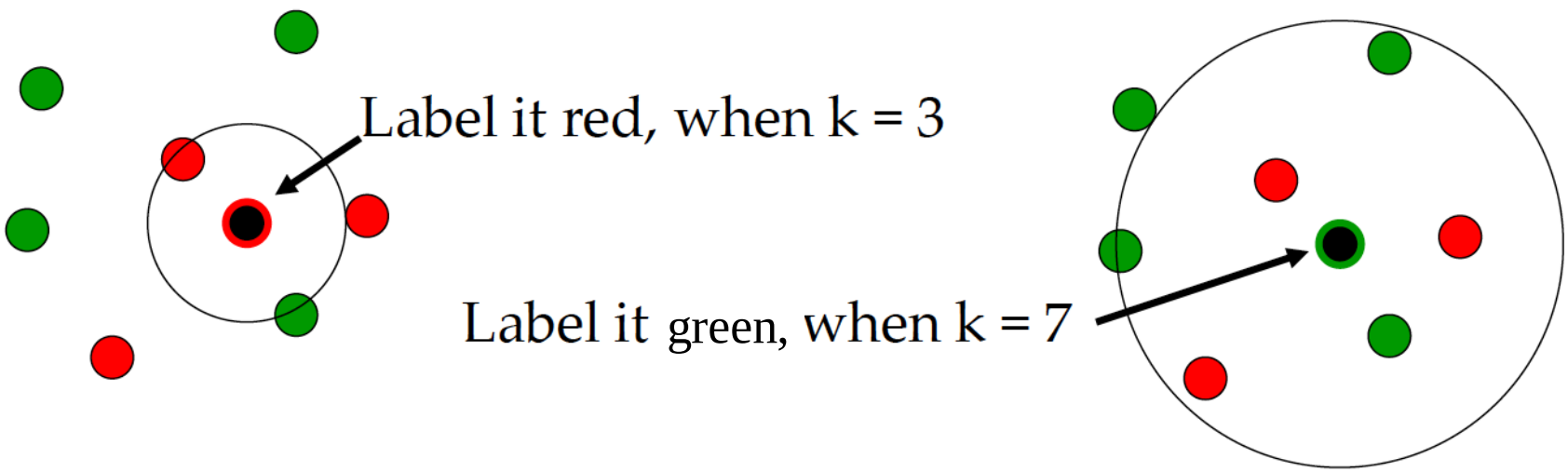
1-Nearest Neighbor (Cont'd)

- The resultant space partition is a voronoi diagram



K-Nearest Neighbor

- Generalizes 1 NN to smooth away noise in the labels
- A new data instance is now assigned the most frequent label of its k nearest neighbors
- E.g., $k = 3$, and $k = 7$



KNN Example

- Training data

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

- Similarity metric: # of matching attributes ($k = 2$)
- New examples
  - Example 1 $(great, no, no, normal, no)$?
  - Example 2 $(mediocre, yes, no, normal, no)$?

KNN Example (Cont'd)

- Pairwise similarity

Index	Instance 1	Instance 2	Instance 3	Instance 4
Example 1	3	4	2	1
Example 2	3	2	4	2

- Prediction ($k = 2$)
  - Example 1 $(great, no, no, normal, no)$? 'yes'
    - Most similar: instance 2 (1 mismatch, 4 match) → yes
    - 2nd most similar: instance 1 (2 mismatch, 3 match) → yes
  - Example 2 $(mediocre, yes, no, normal, no)$? 'yes'/'no'
    - Most similar: instance 3 (1 mismatch, 4 match) → no
    - 2nd most similar: instance 1 (2 mismatch, 3 match) → yes

KNN Limitations

- Each prediction takes $O(n)$ computational complexity
  - Use fancy data structures such as KD-trees to accelerate the search of nearest neighbours
  - Or use locality-sensitive hashing to approximate nearest neighbours with constant computational complexity
- Prediction performance degrades when number of attributes grows
  - Curse of dimensionality: when the number of attributes is big, similarity/distance measures become less reliable
  - Remedy
    - Remove irrelevant attributes in pre-processing
    - Weight attributes differently

Lecture Outline

- Machine Learning for Data Science
- K-Nearest Neighbors Classifier
- Model Selection

How to Choose K?

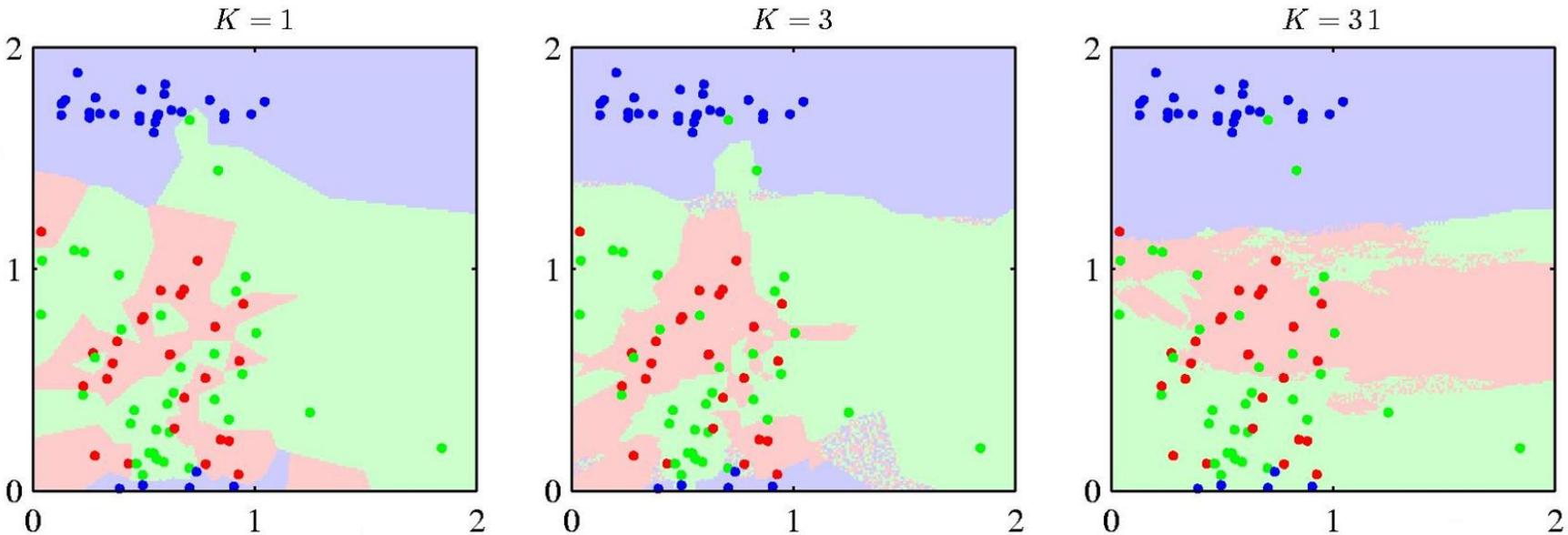
- Question: can we learn K from training data?
  - No! K is a hyperparameter, rather than a model parameter
  - Then, what has been learned in KNN classifier?
    - A partition of the input space
- Usually, K should be determined by model users
  - Different K will produce different classifiers
  - Then, how to choose a value for K?
- **Model Selection**
  - Multiple models generated by different algorithm parameter configurations
  - Multiple models generated by different learning algorithms

Model Complexity

- **Model complexity**
  - Very generally, it refers to the number of degrees of freedom in a learned model
  - Often measured as the number of adjustable weights or parameters in the architecture, e.g., weights in regression
  - High complexity → stronger capability of capturing information from training data
- KNN classifier's complexity?
  - No explicit model parameters
  - Actually, the decision boundary formed from the input space partition is relevant to the model complexity
  - The smoother the boundary is, the complexity is lower

Model Complexity (Cont'd)

- K acts as a smoother and controls model complexity



- $K = 1$ leads to the roughest decision boundaries
  - As 1-NN classifier has the highest complexity, can we just simply select this model as the best for model selection?

Training/Testing Errors

- **Training error** (or empirical error) of a trained model $\hat{f}$ on a training data set of size N

$$E_{emp}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

  - Note that the loss function $L(\cdot,\cdot)$ requires instantiation for a specific model, e.g., the squared error in linear regression

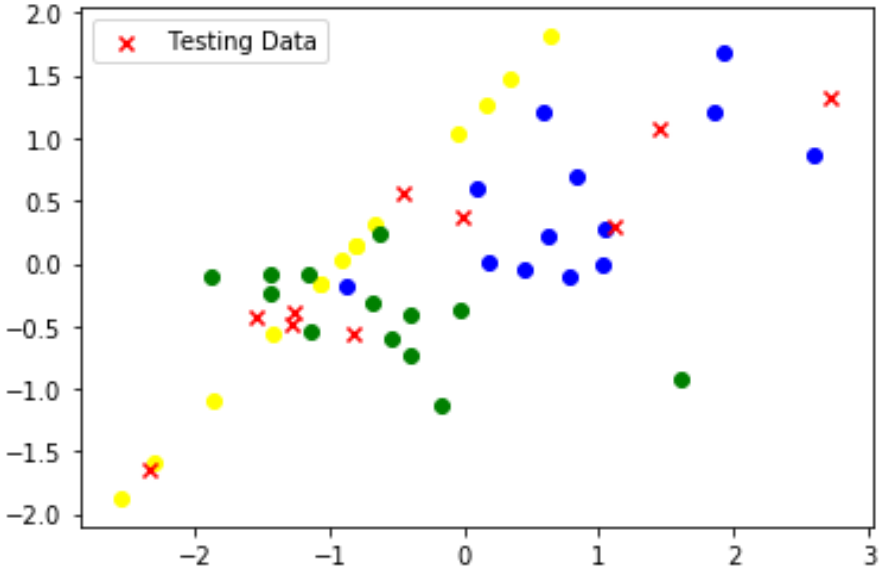
- **Testing error** on a test data set with size N'

$$E_{test}(\hat{f}) = \frac{1}{N'}\sum_{i=1}^{N'} L(y_i, \hat{f}(x_i))$$

  - Indicating the generalisation capability of a learned model

Training/Testing Errors (Cont'd)

- Empirical study on KNN classifiers
  - 40 points for training, 10 points for testing
  - 3 classes
  - $Error$ is calculated by $1.0 - accuracy$



Training/Testing Errors (Cont'd)

- Observations
  - Training error keeps going up when K increases
    - 0 when $K = 1$
  - Testing error does not take lowest value at $K = 1$
    - Testing error is minimized around $K = 9$

- So, a paradox for $K = 1$ case! Why?
  - Overfitting: a model is too strong and captures the very details of training samples, lacking generalization capability
    - Data is noisy, and the model is fitted to noise
    - Fighting against overfitting is important in machine learning!

Training/Testing Errors (Cont'd)

- Observations
  - Both errors are maximized when $K = 40$
    - All testing sample will be predicted as the same label. Why?

- This is the underfitting issue
  - It's easy to address by just increasing model complexity
- Insights
  - Model selection is a non-trivial job!
  - We need to use testing error as performance indicator to guide model selection for the generalization capability

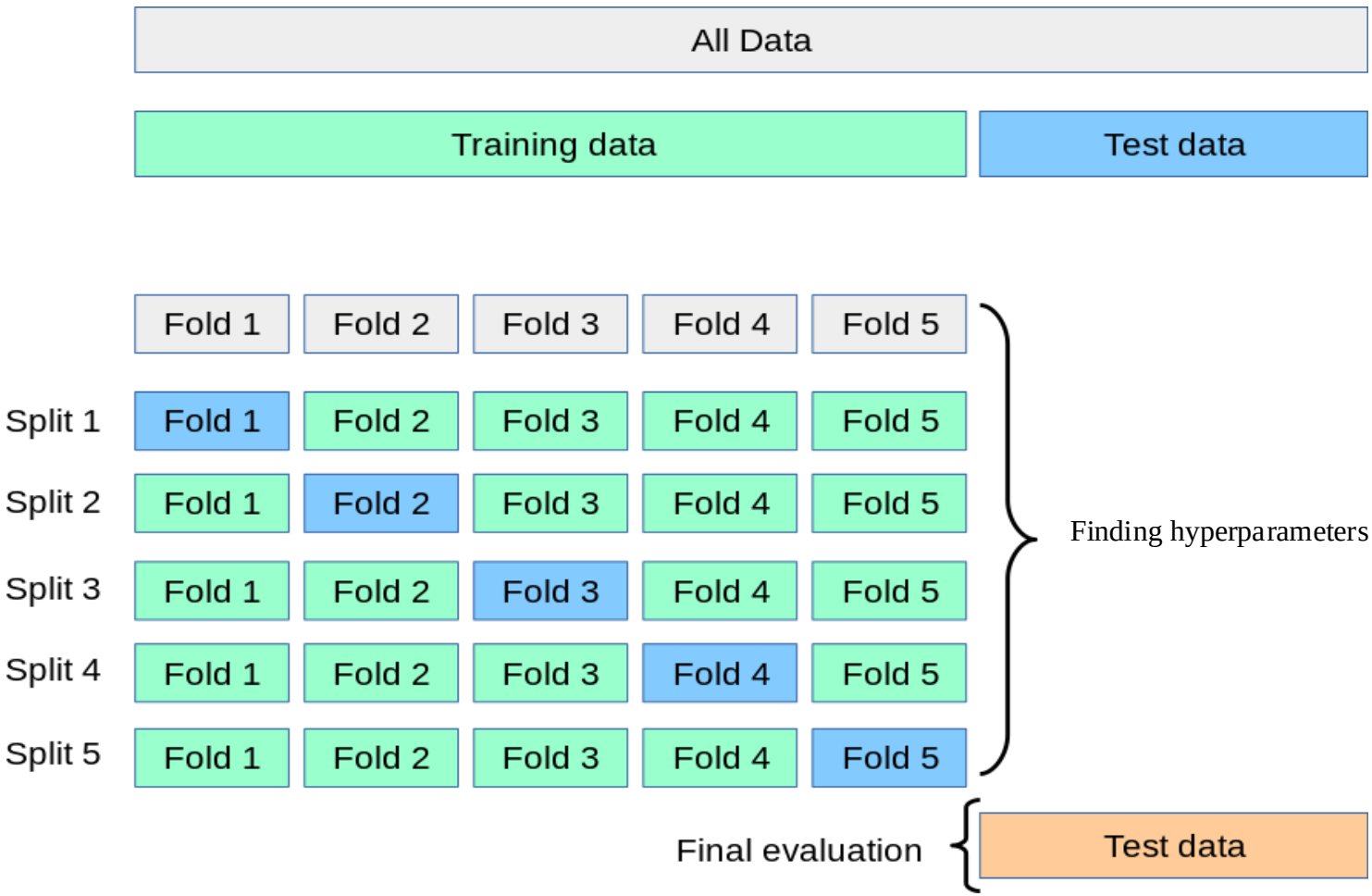
How to select K?

- Increase K
  - Make KNN classifier less sensitive to noise
  - Avoid overfitting
- Decrease K
  - Allow capturing finer structure of space
  - Avoid underfitting
- Pick K not too large, but not too small
  - Data-specific
  - This is model selection by hyperparameter tuning
  - Cross validation can be used to find a suitable K
  - Theoretically, this is related to bias-variance tradeoff

How to Estimate Testing Error ?

- Option 1: randomly divide the available set of samples into two parts: training data and validating data
  - Randomness will not give a robust testing error estimation
  - Wasting of data: the training data fail to contribute testing error while the validating data fail to contribute to training
    - We hope all data instances contribute to better model training and more robust testing error estimation
- Option 2: ***k*-fold cross validation**
  - Randomly partition data into k subsets (k is usually 5, 10)
  - In each round, leave a subset out as the validating data
  - Combined results from multiple rounds are reported as the robust testing error estimation

k-fold Cross Validation



Automatic Tuning

- Challenges in hyperparameter turning
  - Many hyperparameters are continuous
    - E.g., complexity parameter α in Ridge regression
    - Search space is continuous
  - Need to tuning multiple hyperparameters simultaneously
- Automation strategies
  - Grid search
    - Exhaustive search over specified parameter values
  - Random search
    - A fixed number of parameter settings is sampled from the specified distributions

Summary

- Learning and machine learning
- Data, model, and parameter spaces
- Supervised learning vs unsupervised learning
- K-NN classifier
- Model selection and model complexity
- Training/testing error
- k-fold cross validation
- Automatic hyperparameter tuning