Business Intelligence and Data Management

academic year 2019-2020

Dr. Ekaterini Ioannou, Department of Management, University of Tilburg

Lecture 7

topic: k-Nearest Neighbors

material: Chapters 7 (book “Data Mining for Business Intelligence”)

Summary of previous lectures / Agenda

Date		Lecture contents	Lecturer	Lab topics	Test
Jan-27	1	Intro. to BI+ Data Management	Caron		
Jan-30				SQL-1	1
Jan-28	2	Data warehousing	Caron		
Feb-06				SQL-2	2
Feb-03	3	OLAP business databases & dashboard	Caron		
Feb-13				SQL-3 & OLAP	3a & 3b
Feb-10	4	Data mining introduction	Ioannou		
	5	Regression models	Ioannou		
Feb-17	6	Naïve Bayes	Ioannou		
	7	k nearest neighbors	Ioannou		
Feb-20				Bayes & neighbors	4
Feb-27	8	Performance measures	Ioannou		
Mar-02	9	Decision trees	Ioannou		
Mar-05				Dec. trees	5
Mar-09	10	Association rules	Ioannou		
Mar-11,12&13				Ass. Rules	6
Mar-16	11	Clustering (+20 mins exam preparation)	Ioannou		
Mar-19				Clustering	7

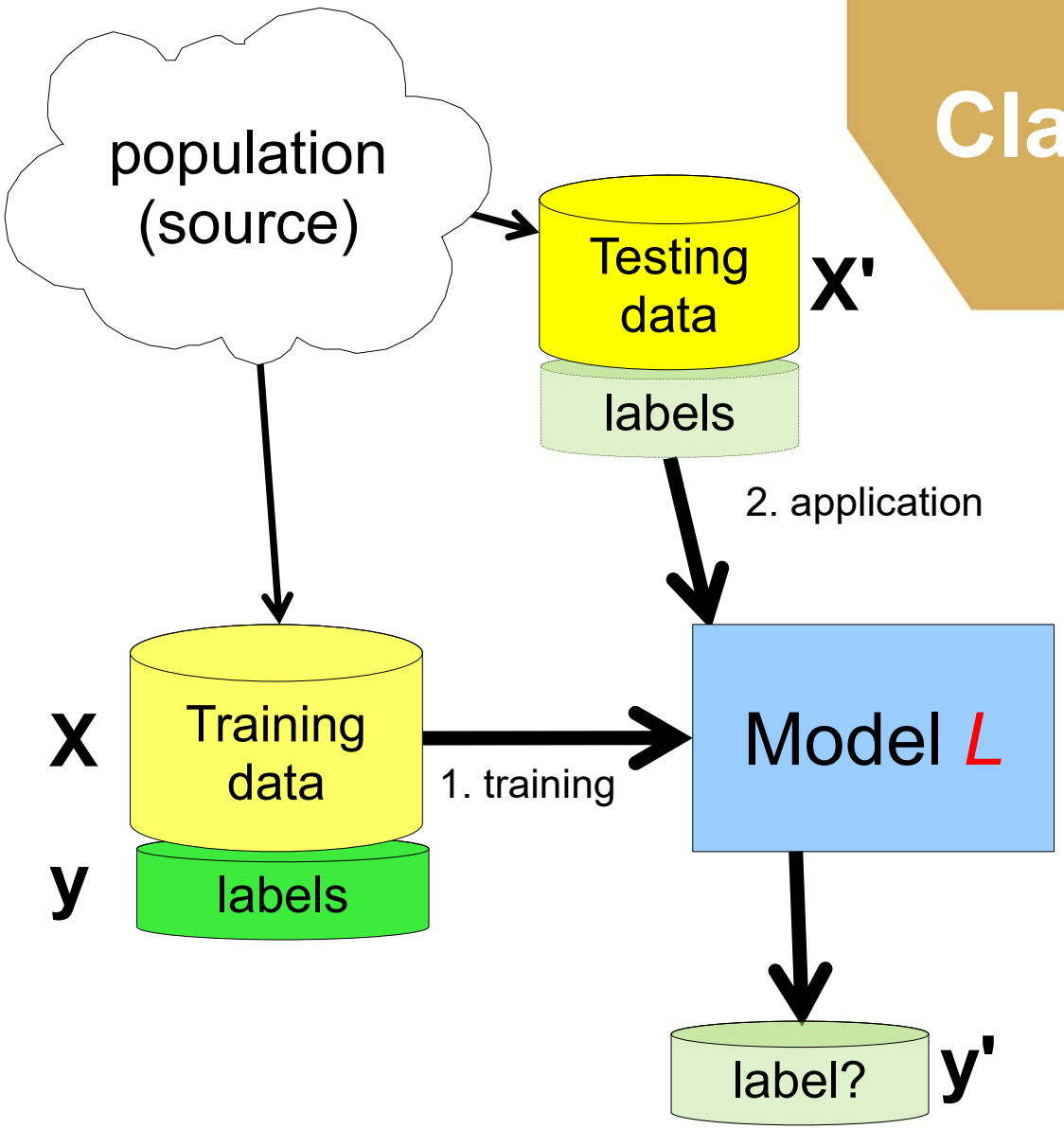
Our focus is currently on Data Mining models

Classification

- Common task in DM
- Examine data where the classification is unknown using data with known outcome
- Goal is to predict what that classification

- Learn classification from the training data
  - Relationship between predictors and outcome
- Apply on testing data, which also includes known outcomes, using the selected model finally
  - Measure how well it will do on unknow data

Classification



Training:

$y = L(X)$

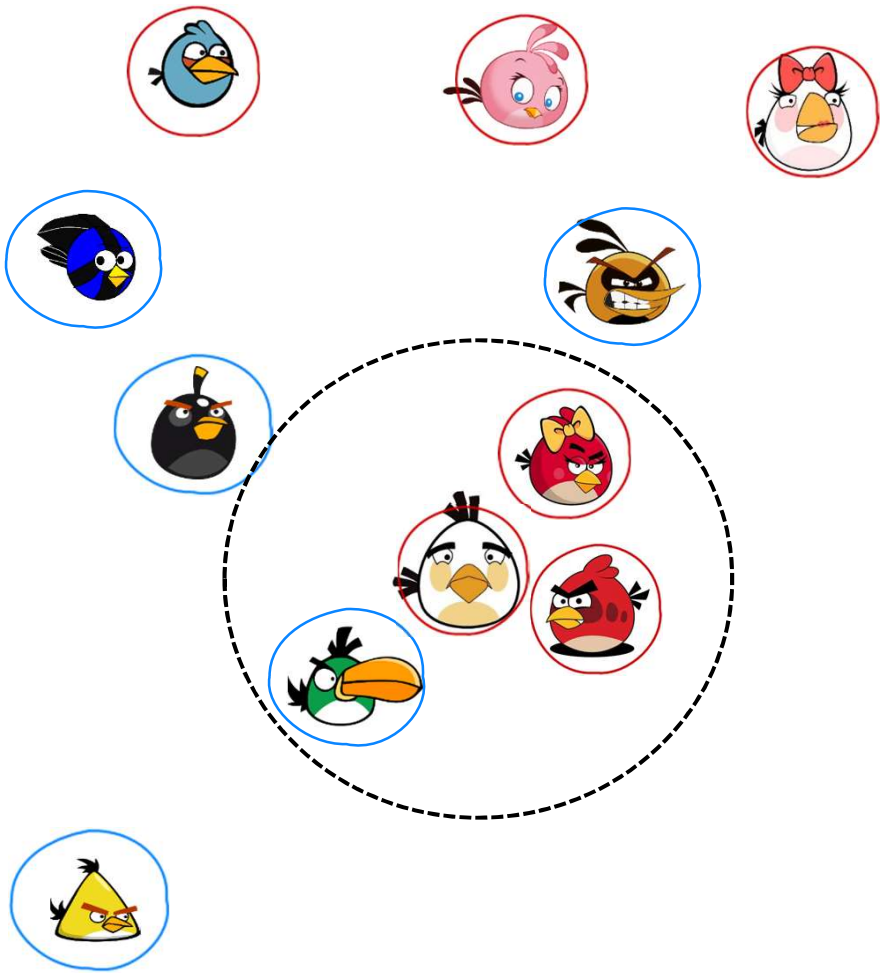
Application:
use *L* over
unseen data

$y' = L(X')$

The *k*-Nearest Neighbors Classifier

- Identify the neighbors of the new record that we wish to classify
  - I.e., the *k* records in the training dataset that are similar to / close by the new record
- Use the neighbors (i.e., these *k* records) to classify the new record into a class
- Assign the new record to the predominant class among these neighbors

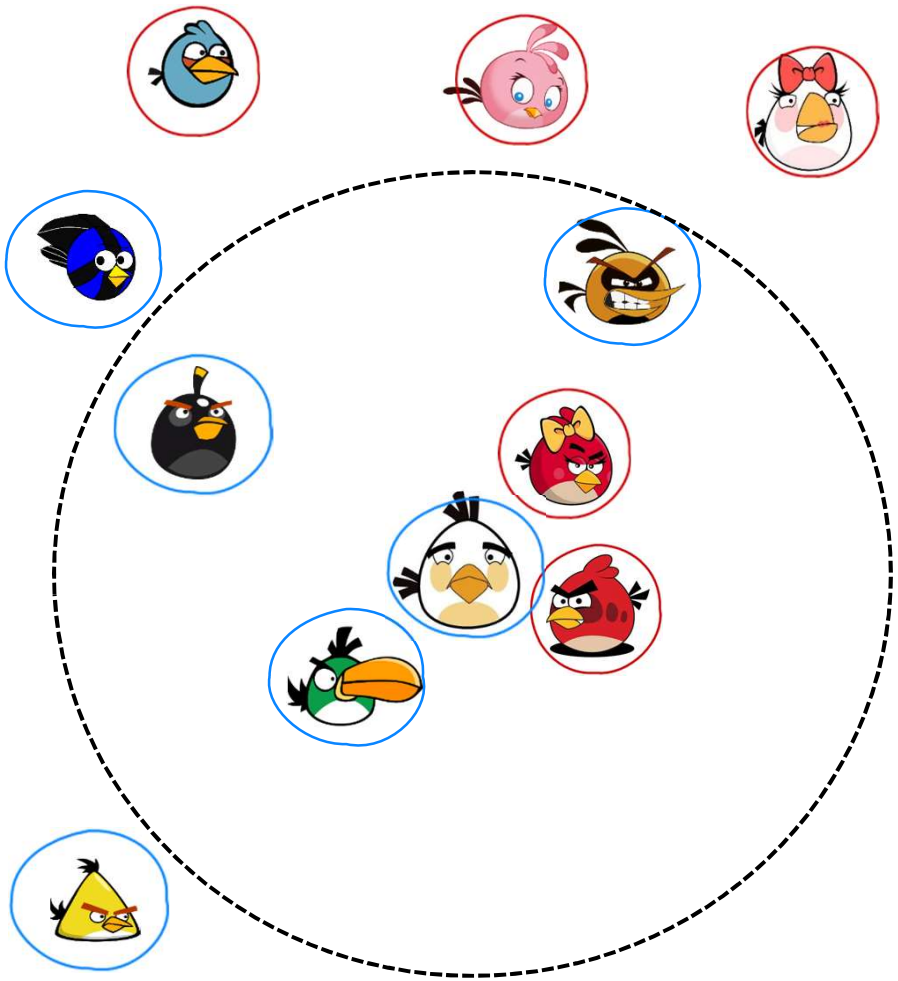
Example



- Items classified into a red and blue class (training)
- New item arrives and we must set its class

- Compare with all items
- Find the 3 items that are most close with it
- Set its class to the predominant class among these 3 neighbors

Example



- Items classified into a red and blue class
- New item arrives and we must set its class

- Compare with all items
- Find the 5 items that are most close with it
- Set its class to the predominant class among these 5 neighbors

The *k*-Nearest Neighbors Classifier

- Identify the neighbors in the training dataset that are similar/close to a new record that we wish to classify
- Use the neighbors (i.e., these *k* records) to classify the new record into a class
- Assign the new record to the predominant class among these neighbors

Sim( , ) = ?

Processing involves:

- Determining the item's neighbors
- Choosing the number of neighbors, i.e., value *k*
- Computing classification (for a categorical outcome) or prediction (for a numerical outcome)

Determining record's neighbors

Choosing the value for *k*

Computing classification or prediction

Determining record's neighbors

- Based on the similarity / closeness between records

➔ Measure the distance based on their values

distance between records r_i and r_j is d_{ij}

- Distances can be defined in multiple ways
- Typically, some properties are required:

P.1. Non-negative: $d_{ij} > 0$

P.2. Self-proximity: $d_{ii} = 0$ (dist. from a record to itself)

P.2. Symmetry: $d_{ij} = d_{ji}$

P.3. Triangle inequality: $d_{ij} \leq d_{ik} + d_{kj}$

I.e., the distance between any pair cannot exceed the sum of distances between the other two pairs

Euclidean Distance

- Most popular distance measure for numerical values
- Record r_i has values $x_{i1}, x_{i2}, \ldots, x_{ip}$
- Euclidean distance between r_i and r_j is

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

Example:

Company	Fixed	RoR	Cost	Load	Demand	Sales	Nuclear	Fuel Cost
Arizona Public Service	1.06	9.2	151	54.4	1.6	9077	0.0	0.628
Boston Edison Co.	0.89	10.3	202	57.9	2.2	5088	25.3	1.555
Central Louisiana Co.	1.43	15.4	113	53.0	3.4	9212	0.0	1.058

- Record r_1
- Record r_2
- Euclidean distance between r_1 and r_2 is

$$d_{12} = \sqrt{(1.06 - 0.89)^2 + (9.2 - 10.3)^2 + \cdots + (0.628 - 1.555)^2} = 3989{:}408$$

Euclidean Distance

- Highly scale dependent
  - I.e., Changing the units of one variable can have a huge influence on the results, for example from cents to dollars
- Solution is normalizing the values before computing
- This converts all measurements to the same scale

→Subtract average and divide by standard deviation

Example:

- Average sales amount across 22 utilities is 8914.045
- Standard deviation is 3549.984
- Sales for Arizona Public Service is 9077
- Normalized sales is $(9077-8914.045)/3549.984 = 0.046$

Euclidean Distance

- Sensitive to *outliers* (discussed in next lectures)
  - Record value(s) that differs significantly from other observations
  - Occur from natural deviations, errors, etc.
- Requires a more robust distance
- Good option is the Manhattan distance
- It looks at the absolute differences rather than squared differences
- Manhattan distance between r_i and r_j is

$$d_{ij} = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

Distance measures for binary values

- Records with binary values (categorical data)
- Use similarity measures and not distance measures
- When for all x_{ij}'s we have binary values

		Record r_j: 0	Record r_j: 1	
Record r_i	0	a	b	$a+b$
	1	c	d	$c+d$
		$a+c$	$b+d$	n

- Matching coefficient: $(\alpha + d) / n$
- Jaquard's coefficient: $d / (b + c + d)$ ignores zeros

Choosing the value for *k*

- *k* is too low:

  may be fitting to the noise in the dataset

- *k* is too high:

  miss out on the method's ability to capture the local structure in the dataset, one of its main advantages

- *k* is the number of records in training dataset:

  assign all records to the majority class in the training data

Choosing the value for *k*

- Balanced choice depends on the nature of the data
- E.g., the more complex and irregular the structure of the data, the lower the optimum value of *k*

- Typically:
  - Values of *k* fall in the range 1 to 20
  - Use odd numbers to avoid ties

How is *k* chosen?

- We use the training data to classify the records in the testing dataset, i.e., use different values for *k*
- Compute error rates for various choices of *k*
- Choose *k* with the best classification performance

BUT

- Testing dataset set is now used as part of the training process (to set *k*)
- We need a new dataset to evaluate the model performance on data that it did not see

… to be discussed in the next lecture

How is *k* chosen?

Validation dataset:

- Take a subset of the training dataset
- Use them for the selecting the model

Error rate:

- Percentage of mistakes

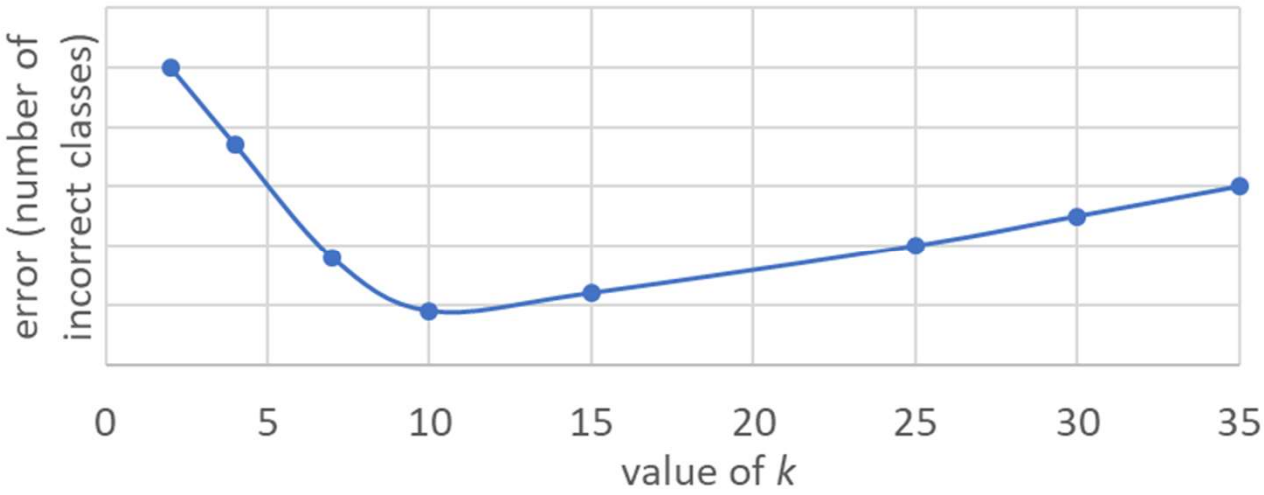
  I.e., assigned an incorrect class to records

How is *k* chosen?

- Predict the class for the records in validation
- Use different values of *k*, e.g., equal to 3, 4, 5, etc.
- Choose *k* that minimize validation error

The follow plot shows the error for different values of *k* over the validation dataset.

What is a good value for *k*?



Numerical Outcome

- Algorithm can be extended to predict continuous values, instead of categorical values

- First step remains unchanged

  I.e., determining neighbors by computing distances

- Second step must be modified

  I.e., determining class through majority voting

- Determine the prediction by taking the average outcome value of the k-nearest neighbors

Advantages

- Simplicity of the method
- Lack of parametric assumptions

Perform surprisingly well especially when

- There is a large enough training set present
- Each class is characterized by multiple combinations of predictor values

Shortcomings

1. Computing the nearest neighbors can be time consuming

Possible solutions:

- Reduce time taken to compute distances by working on less dimensions, generated using dimension reduction techniques
- Speed up identification of nearest neighbors using specialized data structures

Shortcomings

1. Computing the nearest neighbors can be time consuming
2. For every record to be predicted, we compute its distance from the entire set of training records only at the time of prediction

   Known as “lazy learner”

   → This behavior prohibits using this algorithm for real-time prediction of a large number of records simultaneously

Shortcomings

1. Computing the nearest neighbors can be time consuming
2. For every record to be predicted, we compute its distances from the entire set of training records only at the time of prediction
3. Number of records required in the training set to qualify as large increases exponentially with the number of predictors

   Known as “curse of dimensionality”

Possible solution:

- Reduce the number of predictors