COMP5318 Machine Learning and Data Mining

Week 2 Tutorial exercises
K-Nearest Neighbor. Rule-based classifiers: PRISM

Welcome to your first COMP5318 tutorial! Please note the following:

- **For most of the weeks there will be 2 documents with tutorial exercise:**
  1) **theoretical (as this one), involving paper-based exercises and calculations, testing your understanding of the algorithms**
  2) **practical using Python and its machine learning and neural network libraries.**
- **Theoretical: We will do some of these exercises at the lecture (usually the first exercise). The rest should be done at your own time. Make sure that you do all theoretical exercises as they are similar in style to the quiz and exam questions.**
- **Practical: This will be the main focus of the tutorial. Sometimes it may not be possible to finish the practical part during the tutorial. Please do this at home as this part is important for your assignments. We have prepared very detailed notes for the practical part, we hope you will find them useful.**
- **The solutions for both type of exercises will be provided on Wednesday evening – see the unit outline.**

Exercise 1. ***Nearest Neighbor (do in class)***
The dataset below consists of 4 examples described with 3 numeric features (a1, a2 and a3); the class has 2 values: yes and no.

What will be the prediction of 1-Nearest Neighbor (1-NN) and 3-Nearest Neighbor (3-NN) with Euclidian distance for the following new example: a1=2, a2=4, a3=2?

Assume that all attributes are measured on the same scale – no need for normalization.

	a1	**a2**	**a3**	**class**
1	1	3	1	yes
2	3	5	2	yes
3	3	2	2	no
4	5	2	3	no

Exercise adapted from M. Kubat, Introduction to Machine Learning, Springer, 2017

Solution:

D(new, ex1) = sqrt[$(2-1)^2$+$(4-3)^2$+$(2-1)^2$]=sqrt(3) yes
D(new, ex2) = sqrt[$(2-3)^2$+$(4-5)^2$+$(2-2)^2$]=sqrt(2) yes
D(new, ex3) = sqrt[$(2-3)^2$+$(4-2)^2$+$(2-2)^2$]=sqrt(5) no
D(new, ex4) = sqrt[$(2-5)^2$+$(4-2)^2$+$(2-3)^2$]=sqrt(14) no

The closest nearest neighbor is ex. 2, hence 1-NN predicts class=yes
The closest 3 nearest neighbors are ex.2 (yes), ex.1 (yes) and ex.3 (no); the majority class is yes.
Hence, 3-NN predicts class =yes

Exercise 2. ***Nearest neighbor with nominal features (do at your own time)***
Consider the *iPhone* dataset given below. There are 4 nominal attributes (age, income, student, and credit_rating) and the class is buys_iPhone with 2 values: yes and no.

What would be the prediction of 1-NN and 3-NN for the following new example:
age<=30, income=medium, student=yes, credit-rating=fair

If there are ties, make random selection.

Tip: As the examples are described with nominal attributes, when calculating the distance use the following rule:

difference=1 between 2 values that are not the same
difference=0 between 2 values that are the same
e.g. D(1, new)=sqrt(0+1+1+0=)=sqrt(2)

	age	**income**	**student**	**credit rating**	**buy iPhone**
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	[31,40]	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	excellent	no
6	[31,40]	low	yes	excellent	yes
7	<=30	medium	no	fair	no
8	[31,40]	medium	no	excellent	yes
9	>40	medium	no	excellent	no

Dataset adapted from J. Han and M. Kamber, Data Mining, Concepts and Techniques, Morgan Kaufmann.

Solution:

new example: age<=30, income=medium, student=yes, credit-rating=fair
D(1, new) = sqrt(0+1+1+0) = sqrt(2) no
D(2, new) = sqrt(0+1+1+1) = sqrt(3)
D(3, new) = sqrt(1+1+1+0) = sqrt(3)
D(4, new) = sqrt(1+0+1+0) = sqrt(2) yes
D(5, new) = sqrt(1+1+0+1) = sqrt(3)
D(6, new) = sqrt(1+1+0+1) = sqrt(3)
D(7, new) = sqrt(0+0+1+0) = sqrt(1)=1 no
D(8, new) = sqrt(1+0+1+1) = sqrt(3)
D(9, new) = sqrt(1+0+1+1) = sqrt(3)

- 1-NN: ex. 7 (D=1) is the closest neighbor, hence 1-NN predicts buy_iPhone=no
- 3-NN: the 3 closest neighbors are: ex.7 (D=1), ex.1 and ex.4 (D=2); 2 no and 1 yes => the majority class is no. Hence, 3-NN predicts buy_iPhone=no

Exercise 3. PRISM (do at your own time)

Given the training data in the table below, generate the PRISM rules for class=no. In case of ties, make random selection.

Weather data with nominal attributes:

	outlook	temperature	humidity	windy	play
1.	sunny	hot	high	false	no
2.	sunny	hot	high	true	no
3.	overcast	hot	high	false	yes
4.	rainy	mild	high	false	yes
5.	rainy	cool	normal	false	yes
6.	rainy	cool	normal	true	no
7.	overcast	cool	normal	true	yes
8.	sunny	cool	high	false	no
9.	sunny	mild	normal	false	yes
10.	rainy	cool	normal	false	yes
11.	sunny	mild	normal	true	yes
12.	overcast	mild	high	true	yes
13.	overcast	hot	normal	false	yes
14.	rainy	mild	high	true	no

Solution:

Let's start with generating rules for class `no`

```
if ? than class=no
```

10 possible tests with their corresponding accuracy p/t:

outlook=sunny 3/5
outlook=overcast 0/4
outlook=rainy 2/5

temperature=hot 2/4
temperature=cool 2/5
temperature=mild 1/5

humidity=high 4/7
humidity=normal 1/7

windy=true 3/6
windy=false 2/8

Best test: `outlook=sunny`
Rule: `if outlook=sunny then class=no`
Examples covered by the current rule:

	outlook	temperature	humidity	windy	play
1.	sunny	hot	high	false	no
2.	sunny	hot	high	true	no
8.	sunny	cool	high	false	no
9.	sunny	mild	normal	false	yes
11.	sunny	mild	normal	true	yes

Not a perfect rule as it covers also 2 examples from class yes => add other tests to this rule

```
if outlook=sunny and ? then class=no
```

Possible tests with the corresponding accuracy p/t:

temperature=hot 2/2
temperature=cool 1/1
temperature=mild 0/2

humidity=high 3/3
humidity=normal 0/2

windy=true 1/2
windy=false 2/3

Best test: `humidity=high` (bigger coverage than `temperature=hot` and `temperature=cool`)
Rule: `if outlook=sunny and humidity=high then class=no`, perfect rule as it covers only examples from class no => stop adding other tests to this rule; delete the examples covered by the rule; there are still uncovered examples from class no, so generate another rule for class=no

	outlook	temperature	humidity	windy	play
3.	overcast	hot	high	false	yes
4.	rainy	mild	high	false	yes
5.	rainy	cool	normal	false	yes
6.	rainy	cool	normal	true	no
7.	overcast	cool	normal	true	yes
9.	sunny	mild	normal	false	yes
10.	rainy	cool	normal	false	yes
11.	sunny	mild	normal	true	yes
12.	overcast	mild	high	true	yes
13.	overcast	hot	normal	false	yes
14.	rainy	mild	high	true	no

`if ? than class=no`
10 possible tests with their corresponding accuracy p/t:
outlook=sunny 0/2
outlook=overcast 0/4
outlook=rainy 2/5

temperature=hot 0/2
temperature=cool 1/4
temperature=mild 1/5

humidity=high 1/4
humidity=normal 1/7

windy=true 2/5
windy=false 0/6

Best test: `outlook=rainy` (draw with `windy=true,` random selection)
Rule: `if outlook=rainy then class=no`
Examples covered by the current rule:

	outlook	temperature	humidity	windy	play
4.	rainy	mild	high	false	yes
5.	rainy	cool	normal	false	yes
6.	rainy	cool	normal	true	no
10.	rainy	cool	normal	false	yes
14.	rainy	mild	high	true	no

Not a perfect rule as it covers also 3 examples from class yes => add other tests to this rule
`if outlook=rainy and ? then class=no`
Possible tests with the corresponding accuracy p/t:
temperature=cool 1/3
temperature=mild 1/2

humidity=high 1/2
humidity=normal 1/3

windy=true 2/2
windy=false 0/3

Best test: windy=`true`
Rule: `if outlook=rainy and windy=true then class=no`, perfect rule as it covers only examples from class no => stop adding other tests to this rule; delete the examples covered by the rule; there are no more uncovered examples from class no, so the rules for class no are:

```
if outlook=rainy and windy=true then class=no
if outlook=sunny and humidity=high then class=no
```

Repeat the procedure for class yes. PRISM will generate the following rules:

```
if outlook=overcast then class=yes
if humidity=normal and windy=false than class=yes
if temperature=mild and humidity=normal than class=yes
if outlook=rainy and windy=false than class=yes
```