School of Computing and Information Systems
The University of Melbourne
COMP30027 MACHINE LEARNING (Semester 1, 2019)

Tutorial exercises: Week 5

1. For the following dataset:

apple	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES				
4	0	1	1	FRUIT
5	0	5	2	FRUIT
2	5	0	0	COMPUTER
1	2	1	7	COMPUTER
TEST INSTANCES				
2	0	3	1	?
1	2	1	0	?

   (a) Classify the test instances according to the method of **Nearest Prototype**.
   (b) Using the **Euclidean distance** measure, classify the test instances using the 1-NN method.
   (c) Using the **Manhattan distance** measure, classify the test instances using the 3-NN method, for the three weightings we discussed in the lectures: majority class, inverse distance, inverse linear distance.
   (d) Can we do weighted k-NN using **cosine similarity**?

2. Revise SVMs, particularly the notion of "linear separability".
   (a) If a dataset isn't linearly separable, an SVM learner has two major options. What are they, and why might we prefer one to the other?
   (b) Contrary to many geometric methods, SVMs work better (albeit slower) with large attribute sets. Why might this be true?

3. We have now seen a decent selection of (supervised) learners:
   - Naive Bayes
   - 0-R
   - 1-R
   - Decision Trees
   - k-Nearest Neighbour
   - Nearest Prototype
   - Support Vector Machines

   (a) For each, identify the model built during training.
   (b) Rank the learners (approximately) by how fast they can classify a large set of test instances. (Note that this is largely independent of how fast they can build a model, and how well they work in general!)

1. For the following dataset:

apple	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES				
4	0	1	1	FRUIT
5	0	5	2	FRUIT
2	5	0	0	COMPUTER
1	2	1	7	COMPUTER
TEST INSTANCES				
2	0	3	1	?
1	2	1	0	?

(a) Classify the test instances according to the method of **Nearest Prototype**.

(b) Using the **Euclidean distance** measure, classify the test instances using the 1-NN method.

(c) Using the **Manhattan distance** measure, classify the test instances using the 3-NN method, for the three weightings we discussed in the lectures: majority class, inverse distance, inverse linear distance.

(d) Can we do weighted k-NN using **cosine similarity**?

1. (a) Nearest Prototype. 是 NN 的一种变体.

1. Prototype for each class : P_j = averaging of the attribute value

Centroid

$$P_{fruit} = \left(\frac{4+5}{2}, \frac{0+0}{2}, \frac{1+5}{2}, \frac{1+2}{2}\right) = (4.5, 0, 3, 1.5)$$

$$P_{Computer} = \left(\frac{2+1}{2}, \frac{5+2}{2}, \frac{0+1}{2}, \frac{0+7}{2}\right) = (1.5, 3.5, 0.5, 3.5)$$

2. test instance closest to which prototype. by using euclidean/Manhattan

Euclidean distance : $d_E(A,B) = \sqrt{\sum_k (a_k - b_k)^2}$

$$d_E(\text{Test}_1, P_f) = \sqrt{(4.5-2)^2 + (0-0)^2 + (3-3)^2 + (1.5-1)^2}$$

$$= \sqrt{6.5} \checkmark$$

$$d_E(\text{Test}_1, P_c) = \sqrt{25}$$

same for test 2.

b). K-NN 是算 distance between test instance and each training instance.

$$d_E(T_1, A) = \sqrt{(2-4)^2 + (0-0)^2 + (3-1)^2 + (1-1)^2} = \sqrt{8}$$

⋮

c) Manhattan: $d_M(A, B) = \sum_k |a_k - b_k|$

$d_M(T_1, A) = 4$

$d_M(T_1, B) = 6$

⋮

The nearest neighbours for Test 1 is A, B, C

for Test 2 is C, A, D

① Use "Majority Class" method:

Test 1 → Fruit

Test 2 → Computer

② Use "inverse distance" method: → ① first choose ε, e.g. $\varepsilon = 1$

For Test 1 with A: $\frac{1}{4+1} = \frac{1}{5}$

B: $\frac{1}{6+1} = \frac{1}{7}$ → ② then for K-NN, weighted $= \frac{1}{d+\varepsilon}$

C: $\frac{1}{9+1} = \frac{1}{10}$

Fruit $= \frac{1}{5} + \frac{1}{7} = 0.34$ ✓

Comp $= \frac{1}{10} = 0.1$

③ overall score: $\sum$ weighted in ②

③ Use " inverse linear distance " method:

$$W_j = \frac{d_k - d_j}{d_k - d_1}$$

d_k : farthest neighbor

d_1 : nearest neighbor

For test 1:

first neigh A: $\frac{d_3 - d_1}{d_3 - d_1} = \frac{9-4}{9-4} = 1$

second neigh B: $\frac{d_3 - d_2}{d_3 - d_1} = \frac{9-6}{9-4} = 0.6$

third neigh C: $\frac{d_3 - d_3}{d_3 - d_1} = \frac{9-9}{9-4} = 0$

Fruit = 1+0.6 = 1.6 ✓

Comp = 0

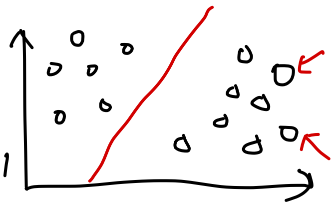
(d) Yes!

2. Revise SVMs, particularly the notion of "linear separability".

   (a) If a dataset isn't linearly separable, an SVM learner has two major options. What are they, and why might we prefer one to the other?

   (b) Contrary to many geometric methods, SVMs work better (albeit slower) with large attribute sets. Why might this be true?

(a) 1. Soft margins : permits a few points to be "wrong"



2. Kernal methods : $\Phi : \mathbb{R}^2 \to \mathbb{R}^n$ to high-dimensional space

Soft margin - 一般是 suspect data is essentially linear-separable

kernal method: for small data set. takes long time

(b) instance has many attributes, - 一些 useful. - 一些 not-so-useful.

Many geometric method assume all attribute are equally important

For example. using Manhattan. the distance between useful attribute may small, but non-useful may be very large, then two instance may not so similar. SVM weight each attribute, so predidiction more meaningful. SVM linear combination, distance linear independent.

3. We have now seen a decent selection of (supervised) learners:
   - Naive Bayes
   - 0-R
   - 1-R
   - Decision Trees
   - k-Nearest Neighbour
   - Nearest Prototype
   - Support Vector Machines

   (a) For each, identify the model built during training.

   (b) Rank the learners (approximately) by how fast they can classify a large set of test instances. (Note that this is largely independent of how fast they can build a model, and how well they work in general!)

(a) NB : a set of prior Prob $P(c_j)$ and a set of posterior Prob $P(a_k|c_j)$

0-R : Class distribution. label of the most frequency class

1-R : most useful attribute + majority class

Decision Tree : non-terminal node is attribute
each branch is attribute value
each leaf is labelled as class.

K-NN : just the dataset itself

Nearest Prototype : prototype (vector) of each class

SVM : maximum - margin hyperplane (w & b)

(b) N training set C classes D attributes.

For each test instance:

0-R : $O(0)$ 1-R : $O(1)$ DT : $O(D)$

NP : $O(CD)$ NB : $O(CD+C)$

slow

SVM : $O(CD+C)$ if using one vs one $O(C^2D+C^2)$

KNN : $O(ND+k)$