School of Computing and Information Systems
The University of Melbourne
COMP90049 Knowledge Technologies (Semester 2, 2017)
Workshop exercises: Week 8

1. What is **overfitting**? What does it mean for a classifier to **generalise**?

   - A classifier "generalises" when it learns the target function well, rather than the (possibly meaningless) specifics of the training set. Overfitting is when the classifier fails to generalise — it builds a model which describes the training data very well, but doesn't describe the test data well.

2. A **confusion matrix** is an indication of the performance of a classifier over a set of test data, by counting the various output instances:

		Actual			
		a	*b*	*c*	*d*
Classified	*a*	10	2	3	1
	b	2	5	3	1
	c	1	3	7	1
	d	3	0	3	5

   (a) Calculate the classification **accuracy** of the system. Find the **error rate** for the system.

   - Accuracy is defined as the fraction of correctly identified instances, out of all of the instances. In the case of a confusion matrix, the correct instances are the ones enumerated along the main diagonal (classified as a and actually a etc.):

$$\begin{aligned} \textit{Accuracy} &= \frac{\text{\# of correctly identified instances}}{\text{total \# of instances}} \\ &= \frac{10+5+7+5}{10+2+3+1+2+5+3+1+1+3+7+1+3+0+3+5} \\ &= \frac{27}{50} = 54\% \end{aligned}$$

   - Error rate is just the complement of accuracy:

$$\begin{aligned} \textit{Error Rate} &= \frac{\text{\# of incorrectly identified instances}}{\text{total \# of instances}} \\ &= 1 - \textit{Accuracy} \\ &= 1 - \frac{27}{50} = 46\% \end{aligned}$$

   (b) Calculate the **precision**, **recall**, **F-score** (where $\beta = 1$), **sensitivity**, and **specificity** for class d. (Why can't we do this for the whole system? How can we consider the whole system?)

   - Precision for a given class is defined as the fraction of correctly identified instances of that class, from the times that class was attempted to be classified. We are interested in the true positives (TP) where we attempted to classify an item as an instance of said class (in this case, d) and it was actually of that class (d): in this case, there are 5 such instances. The false positives (FP) are those items that we attempted to classify as being of class d, but they were actually of some other class: there are $3 + 0 + 3 = 6$ of those.

$$\begin{aligned} \textit{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ &= \frac{5}{5+3+0+3} \\ &= \frac{5}{11} \approx 45\% \end{aligned}$$

- Recall for a given class is defined as the fraction of correctly identified instance of that class, from the times that class actually occurred. This time, we are interested in the true positives, and the false negatives (FN): those items that were actually of class d, but we classified as being of some other class; there are $1+1+1=3$ of those.

$$\begin{aligned}\textit{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ &= \frac{5}{5+1+1+1} \\ &= \frac{5}{8} \approx 62\%\end{aligned}$$

- F-score is a measure which attempts to combine precision (P) and recall (R) into a single score. In general, it is calculated as:

$$F_\beta = \frac{(1+\beta^2)P\cdot R}{(\beta^2\cdot P)+R}$$

- By far, the most typical formulation is where the parameter β is set to 1: this means that precision and recall are equally important to the score, and that the score is a harmonic mean:

$$F_{\beta=1} = \frac{2\cdot P\cdot R}{P+R}$$

- In this case, we have calculated the precision of class d to be $\frac{5}{11}$ and the recall to be $\frac{5}{8}$. The F-score where $(\beta=1)$ of class d is then:

$$\begin{aligned}F_{\beta=1} &= \frac{2\cdot P\cdot R}{P+R} \\ &= \frac{2\cdot\frac{5}{11}\cdot\frac{5}{8}}{\frac{5}{11}+\frac{5}{8}} \\ &= \frac{50}{95} \approx 53\%\end{aligned}$$

- Sensitivity is defined the same way as recall: $\frac{TP}{TP+FN}$.
- Specificity is precision with respect to the negative instances:

$$\begin{aligned}\textit{Specificity} &= \frac{\text{TN}}{\text{TN}+\text{FP}} \\ &= \frac{10+2+3+2+5+3+1+3+7}{10+2+3+2+5+3+1+3+7+3+0+3} \\ &= \frac{36}{42} \approx 86\%\end{aligned}$$

3. How is **holdout** evaluation different to **cross-validation** evaluation?

   - In a holdout evaluation strategy, we partition our data into a training set and a test set: we build the model on the former, and evaluate on the latter.
   - In a cross-validation evaluation strategy, we do the same as above, but a number of times, where each iteration uses one partition of the data as a test set and the rest as a training set (and the partition is different each time).

4. Revise **linear regression**.

   (a) What are we attempting to model with linear regression? Why do we minimise "RSS"? What methods do we use? What assumptions are we making?

- With linear regression, we are trying to choose a "line of best fit" for the data that we are given. We have independent variable(s), that we know for both of the training data and the test data, and a dependent variable that we know for some training instances, and wish to predict for some test instances (this is a *supervised* algorithm). We will choose a polynomial of degree 1 in each of the independent variables, whose coefficients minimise the error in the training data.
- We minimise the residual sum of squares (RSS), because that is our mechanism for estimating the error rate of a given candidate line of best fit. The fact that each error (between the predicted value of the dependent variable according to the model, and the observed value from each instance in the data set) is squared means that we heavily penalise curves which miss a few points by a large degree, even if they are very close to a large number of training data points. This means that our model tends to **overfit** less. Also, the fact that the errors are squared makes the mathematical model simpler to solve (rather than if we were cubing the errors, or taking sinh or something).
- (Mostly we use gradient descent models to find the best coefficients for the line of best fit. The reason for this is that numerical error makes the matrix problem too unreliable to solve.)
- The main assumption here is that a polynomial of degree 1 is adequate to represent the data (which it often isn't). We're also assuming that our gradient descent model doesn't find a local optimum that isn't the global optimum (although there are ways of controlling for this). There are a few more subtle assumptions, of which the most important are: errors in the dependent variable aren't correlated; the independent variables are linearly independent of each other; the independent variable values don't have their own error. You can see a fair introduction to these on Wikipedia. (`http://en.wikipedia.org/wiki/Linear_regression#Assumptions`)

5. For the following dataset:

apple	*ibm*	*lemon*	*sun*	CLASS	label	length
TRAINING INSTANCES						
4	0	1	1	FRUIT	F_1	$\sqrt{18}$
5	0	5	2	FRUIT	F_2	$\sqrt{54}$
2	5	0	0	COMPUTER	C_1	$\sqrt{29}$
1	2	1	7	COMPUTER	C_2	$\sqrt{55}$
TEST INSTANCES						
2	0	3	1	?	T_1	$\sqrt{14}$
1	0	1	0	?	T_2	$\sqrt{2}$

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

- For this part, we are interested in the (Euclidean) distances between the instances — this is more sensitive to the length of the instance (vector) than the cosine similarity, which may or may not be appropriate, depending on the data set.
- Recall the Euclidean distance between two points A and B:

$$\text{dist}(A, B) = \sqrt{\sum_k (a_k - b_k)^2}$$

- For T_1, we find the Euclidean distances to the four training instances:

$$\begin{aligned}
\text{dist}(F_1, T_1) &= \sqrt{\sum_k (F_{1,k} - T_{1,k})^2} \\
&= \sqrt{(4-2)^2 + (0-0)^2 + (1-3)^2 + (1-1)^2} \\
&= \sqrt{8} \approx 2.828 \\
\text{dist}(F_2, T_1) &= \sqrt{(5-2)^2 + (0-0)^2 + (5-3)^2 + (2-1)^2} \\
&= \sqrt{14} \approx 3.742 \\
\text{dist}(C_1, T_1) &= \sqrt{(2-2)^2 + (5-0)^2 + (0-3)^2 + (0-1)^2} \\
&= \sqrt{35} \approx 5.916 \\
\text{dist}(C_2, T_1) &= \sqrt{(1-2)^2 + (2-0)^2 + (1-3)^2 + (7-1)^2} \\
&= \sqrt{45} \approx 6.708
\end{aligned}$$

- With this distance metric, close neighbours are ones with low scores. If we use the 1-nearest neighbour method, we observe that the closest instance is F_1. This is a FRUIT instance, so we choose FRUIT for this test instance.
- For the second test instance:

$$\begin{aligned}
\text{dist}(F_1, T_2) &= \sqrt{\sum_k (F_{1,k} - T_{2,k})^2} \\
&= \sqrt{(4-1)^2 + (0-0)^2 + (1-1)^2 + (1-0)^2} \\
&= \sqrt{10} \approx 3.162 \\
\text{dist}(F_2, T_2) &= \sqrt{(5-1)^2 + (0-0)^2 + (5-1)^2 + (2-0)^2} \\
&= \sqrt{36} = 6.000 \\
\text{dist}(C_1, T_2) &= \sqrt{(2-1)^2 + (5-0)^2 + (0-1)^2 + (0-0)^2} \\
&= \sqrt{27} \approx 5.196 \\
\text{dist}(C_2, T_2) &= \sqrt{(1-1)^2 + (2-0)^2 + (1-1)^2 + (7-0)^2} \\
&= \sqrt{53} \approx 7.280
\end{aligned}$$

- Once more, the best instance is F_1, so we choose FRUIT.

(b) Using the **cosine similarity** measure, classify the test instances using the 3-NN method. Extend this to the **weighted** 3-NN method.

- We're using the cosine measure of similarity, interpretting the instances as vectors in the feature space, and we'll find the angles between the vectors to find the nearest neighbours among the training instances to each of the test instances.
- Recall that the cosine measure between two vectors A and B is calculated as:

$$\cos(A, B) = \frac{A \cdot B}{\mid A \mid \cdot \mid B \mid}$$

- Let's start by pre-calculating the lengths of the vectors (they're shown in the table above). For example:

$$\begin{aligned}
\mid F_1 \mid &= \sqrt{4^2 + 0^2 + 1^2 + 1^2} \\
&= \sqrt{18} \approx 4.24
\end{aligned}$$

- To find the nearest neighbours for T_1, we'll calculate the cosine measure for each of the four training instances:

$$
\begin{aligned}
\cos(F_1, T_1) &= \frac{F_1 \cdot T_1}{\mid F_1 \mid \cdot \mid T_1 \mid} \\
&= \frac{\langle 4,0,1,1\rangle \cdot \langle 2,0,3,1\rangle}{\mid \langle 4,0,1,1\rangle \mid \cdot \mid \langle 2,0,3,1\rangle \mid} \\
&= \frac{4 \cdot 2 + 0 \cdot 0 + 1 \cdot 3 + 1 \cdot 1}{\sqrt{18} \cdot \sqrt{14}} \\
&= \frac{12}{\sqrt{18} \cdot \sqrt{14}} \approx 0.7559 \\
\cos(F_2, T_1) &= \frac{F_2 \cdot T_1}{\mid F_2 \mid \cdot \mid T_1 \mid} \\
&= \frac{\langle 5,0,5,2\rangle \cdot \langle 2,0,3,1\rangle}{\mid \langle 5,0,5,2\rangle \mid \cdot \mid \langle 2,0,3,1\rangle \mid} \\
&= \frac{27}{\sqrt{54} \cdot \sqrt{14}} \approx 0.9820 \\
\cos(C_1, T_1) &= \frac{C_1 \cdot T_1}{\mid C_1 \mid \cdot \mid T_1 \mid} \\
&= \frac{\langle 2,5,0,0\rangle \cdot \langle 2,0,3,1\rangle}{\mid \langle 2,5,0,0\rangle \mid \cdot \mid \langle 2,0,3,1\rangle \mid} \\
&= \frac{4}{\sqrt{29} \cdot \sqrt{14}} \approx 0.1985 \\
\cos(C_2, T_1) &= \frac{C_2 \cdot T_1}{\mid C_2 \mid \cdot \mid T_1 \mid} \\
&= \frac{\langle 1,2,1,7\rangle \cdot \langle 2,0,3,1\rangle}{\mid \langle 1,2,1,7\rangle \mid \cdot \mid \langle 2,0,3,1\rangle \mid} \\
&= \frac{12}{\sqrt{55} \cdot \sqrt{14}} \approx 0.4324
\end{aligned}
$$

- At this point, we consider the four values that we calculated. We're looking for the 3-best nearest neighbours: for the cosine measure, these are the instance with the greatest values. For T_1, this is F_2 with a score of 0.9820, and F_1 and C_2.
- Two of the 3-best neighbours are of class FRUIT, whereas only one is of class COMPUTER: we apply a voting procedure, and here FRUIT (with 2) out-votes COMPUTER (with 1), so we classify T_1 as FRUIT.
- What if we were using "weighted" k-nearest neigbour? Well, each of the k best neighbours gets a weighted vote, according to the cosine similarity score we calculated above. We then accumulate these weighted votes, and the class with the greater weighting is the one that we choose.
- In this case, FRUIT has a total weight of 1.7379 (from F_2 and F_1) and COMPUTER only 0.4324 (from C_2), so we choose FRUIT.

- T_2 is similar:

$$\begin{aligned}
\cos(F_1, T_2) &= \frac{F_1 \cdot T_2}{\mid F_1 \mid \cdot \mid T_2 \mid} \\
&= \frac{\langle 4,0,1,1\rangle \cdot \langle 1,0,1,0\rangle}{\mid \langle 4,0,1,1\rangle \mid \cdot \mid \langle 1,0,1,0\rangle \mid} \\
&= \frac{4 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 0}{\sqrt{18} \cdot \sqrt{2}} \\
&= \frac{5}{\sqrt{18} \cdot \sqrt{2}} \approx 0.8333 \\
\cos(F_2, T_2) &= \frac{F_2 \cdot T_2}{\mid F_2 \mid \cdot \mid T_2 \mid} \\
&= \frac{\langle 5,0,5,2\rangle \cdot \langle 1,0,1,0\rangle}{\mid \langle 5,0,5,2\rangle \mid \cdot \mid \langle 1,0,1,0\rangle \mid} \\
&= \frac{10}{\sqrt{54} \cdot \sqrt{2}} \approx 0.9623 \\
\cos(C_1, T_2) &= \frac{C_1 \cdot T_2}{\mid C_1 \mid \cdot \mid T_2 \mid} \\
&= \frac{\langle 2,5,0,0\rangle \cdot \langle 1,0,1,0\rangle}{\mid \langle 2,5,0,0\rangle \mid \cdot \mid \langle 1,0,1,0\rangle \mid} \\
&= \frac{2}{\sqrt{29} \cdot \sqrt{2}} \approx 0.2626 \\
\cos(C_2, T_2) &= \frac{C_2 \cdot T_2}{\mid C_2 \mid \cdot \mid T_2 \mid} \\
&= \frac{\langle 1,2,1,7\rangle \cdot \langle 1,0,1,0\rangle}{\mid \langle 1,2,1,7\rangle \mid \cdot \mid \langle 1,0,1,0\rangle \mid} \\
&= \frac{2}{\sqrt{55} \cdot \sqrt{2}} \approx 0.1907
\end{aligned}$$

- Here again, F_2 is the nearest neighbour, followed by F_1 and C_2.
- Hence, in the unweighted version, we choose FRUIT $(2 > 1)$, and in the weighted version, we still choose FRUIT $(1.7956 > 0.2626)$.

6. For the following dataset:

apple	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES				
Y	N	Y	Y	FRUIT
Y	N	Y	Y	FRUIT
Y	Y	N	N	COMPUTER
Y	Y	Y	Y	COMPUTER
TEST INSTANCES				
Y	N	Y	Y	?
Y	N	Y	N	?

Use the method of **Naive Bayes** classification, as shown in lectures, to classify the test instances. Revise some of the assumptions that are built into the model.

- The assumption most central to the formulation of the Naive Bayes classifier that we have discussed is the **conditional independence assumption**, namely, that each attribute is independent to all of the other attributes, given the class under consideration. This assumption (while false) is necessary to make the problem tractable, where finding reliable estimates of the joint distribution of features requires more data than we are likely to have.
- Another important assumption is in the way we find probabilities from the training data. This is most important for the **maximum likelihood estimation** we perform over the class priors (in contrast, we hedge our bets on the posterior probabilities of the terms using **smoothing**).
- The fact that these assumptions are demonstrably untrue makes it seem like the classifier should not be effective at predicting the classes of unseen data. However, Naive Bayes is a pretty solid performer! It turns out that the methodology is robust enough to produce decent predictions, despite small (and predictable) discrepancies in the individual probabilities under consideration.
- There are also a number of more minor assumptions:
  - We assume that the distribution of classes in the test set is (roughly) the same as the distribution of classes in the training set, and also that all of the classes in the test data are attested in the training data. This is often phrased as "the training and test instances were sampled from the same underlying distribution." (In fact, this is an assumption of most supervised machine learning algorithms.)
  - We typically require some kind of assumption for our smoothing method, depending on which smoothing method we actually use.
  - We need to make some assumptions about the distribution of continuous attributes (typically Gaussian) if they exist in our data set.
- Anyway, let's classify these test instances:
- Naive Bayes selects a class c from a set of classes C for a test instance $T = \langle t_1, t_2, \ldots, t_n \rangle$ using a set of training instances $\mathcal{D}$ according to:

$$c = \text{argmax}_{c_j \in C} P(c_j) P(T \mid c_j) \tag{1}$$

- We will expand $P(T \mid c_j)$ based on our conditional independence assumption (according the attribute values t_i seen in our test instance):

$$P(T \mid c_j) = \prod P(t_i \mid c_j) \tag{2}$$

- We can calculate the prior probabilities of the classes straight from the training data, using maximum likelihood estimation. For FRUIT, 2 of the 4 instances are FRUIT:

$$P(f) = \frac{2}{4} = \frac{1}{2}$$

- We find the conditional probabilities $P(t_i \mid c_j)$ based on maximum likelihood estimation from the data set, smoothing by replacing 0 values with a small, positive, non-zero value ϵ. (In the real world, we would probably use Laplacian smoothing, or some other more serious smoothing method.)
- To do this, we will observe what proportion of the instances for the given class contain the term that we're looking for. For example, for $P(a = \mathrm{T} \mid f)$: of the two FRUIT instances, both of them contain *apple*.

$$\begin{aligned}
P(a = \mathrm{T} \mid f) &= \frac{2}{2} = 1 \\
P(i = \mathrm{T} \mid f) &= \frac{0}{2} = 0 \\
P(l = \mathrm{T} \mid f) &= \frac{2}{2} = 1 \\
P(s = \mathrm{T} \mid f) &= \frac{2}{2} = 1 \\
P(a = \mathrm{T} \mid c) &= \frac{2}{2} = 1 \\
P(i = \mathrm{T} \mid c) &= \frac{2}{2} = 1 \\
P(l = \mathrm{T} \mid c) &= \frac{1}{2} \\
P(s = \mathrm{T} \mid c) &= \frac{1}{2}
\end{aligned}$$

- When we substitute these values into (2) above, we will replace 0 values with ϵ.
- We will also need the conjugate probabilities, for example $P(t = \mathrm{F} \mid c)$. To find these, we will observe that $P(a \mid b) + P(\bar{a} \mid b) = 1$. Consequently, $P(t = \mathrm{F} \mid c) = 1 - P(t = \mathrm{T} \mid c)$.
- Going all the way back to (1), and substituting our simplification from (2), we will now consider the values for FRUIT and COMPUTER (they aren't really probabilities any more, but it isn't important now).

$$\begin{aligned}
\textsc{fruit} \quad &: \quad P(a = \mathrm{T} \mid f)P(i = \mathrm{F} \mid f)P(l = \mathrm{T} \mid f)P(s = \mathrm{T} \mid f)P(f) \\
&= \quad P(a = \mathrm{T} \mid f)(1 - P(i = \mathrm{T} \mid f))P(l = \mathrm{T} \mid f)P(s = \mathrm{T} \mid f)P(f) \\
&= \quad 1 \times (1 - 0) \times 1 \times 1 \times \frac{1}{2} \\
&= \quad \frac{1}{2} \\
\textsc{comp} \quad &: \quad P(a = \mathrm{T} \mid c)P(i = \mathrm{F} \mid c)P(l = \mathrm{T} \mid c)P(s = \mathrm{T} \mid c)P(c) \\
&= \quad P(a = \mathrm{T} \mid c)(1 - P(i = \mathrm{T} \mid c))P(l = \mathrm{T} \mid c)P(s = \mathrm{T} \mid c)P(c) \\
&= \quad 1 \times (1 - 1) \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \\
&\approx \quad 1 \times \epsilon \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \\
&\approx \quad \frac{\epsilon}{8}
\end{aligned}$$

- For this test instance, T_1, we choose FRUIT, because $\frac{1}{2} > \frac{\epsilon}{8}$ (because ϵ is small).

- For T_2, the calculations are similar, except for the fact that $s = \mathrm{F}$:

$$\begin{aligned}
\textsc{fruit} &: P(a=\mathrm{T} \mid f)P(i=\mathrm{F} \mid f)P(l=\mathrm{T} \mid f)P(s=\mathrm{F} \mid f)P(f) \\
&= P(a=\mathrm{T} \mid f)(1-P(i=\mathrm{T} \mid f))P(l=\mathrm{T} \mid f)(1-P(s=\mathrm{T} \mid f))P(f) \\
&= 1 \times (1-0) \times 1 \times (1-1) \times \frac{1}{2} \\
&\approx \frac{\epsilon}{2} \\
\textsc{comp} &: P(a=\mathrm{T} \mid c)P(i=\mathrm{F} \mid c)P(l=\mathrm{T} \mid c)P(s=\mathrm{F} \mid c)P(c) \\
&= P(a=\mathrm{T} \mid c)(1-P(i=\mathrm{T} \mid c))P(l=\mathrm{T} \mid c)(1-P(s=\mathrm{T} \mid c))P(c) \\
&= 1 \times (1-1) \times \frac{1}{2} \times (1-\frac{1}{2}) \times \frac{1}{2} \\
&\approx \frac{\epsilon}{8}
\end{aligned}$$

- And again, this is classified as FRUIT.

7. [EXTENSION] Revise the **multinomial distribution**. Naive Bayes can be extended to account for integer frequencies in the data (like in Question 2) using this model. Read up on so-called **multinomial Naive Bayes**.