School of Computing and Information Systems
The University of Melbourne
COMP90049 Knowledge Technologies (Semester 2, 2018)
Workshop sample solutions: Week 12

1. For the following set of instances:

a_1	a_2	a_3	c
hot	windy	dry	Yes
mild	windy	rainy	No
hot	windy	rainy	Yes
cool	still	dry	Yes
cool	still	rainy	No
hot	still	dry	No
mild	still	dry	Yes

Construct all of the **1-itemsets** and calculate their **confidences** and **supports**. Discuss how you would continue mining for effective **Association Rules**.

- The 1-itemsets are all of the values each of the attributes can take. Counting the "class" values (this is an unsupervised method), there are 9 such itemsets: {*hot*},{*mild*},{*cool*},{*windy*},{*still*}, {*dry*},{*rainy*},{*Yes*}, and {*No*}.
- When we consider support and confidence, it's typically over an **association rule**, not an itemset. There are two (defective) association rules that we can generate for a given 1-itemset, say {*hot*}: one with the item in the antecedent and one with the item in the consequent:

$$\{hot\} \rightarrow \emptyset \quad (1)$$
$$\emptyset \rightarrow \{hot\} \quad (2)$$

- Recall the formulae for confidence and support, for N instances in total:

$$\text{Conf}(A \rightarrow B) = \frac{n(A,B)}{n(A)}$$
$$\text{Supp}(A \rightarrow B) = \frac{n(A,B)}{N}$$

- For rules of type (1) (with an empty consequent), the confidence will be 1 (because $n(A,*) = n(A)$). For rules of type (2) (with an empty antecedent), the confidence will be equal to the support (because $n(*) = N$). Hence, the only interesting value we need to calculate at this point is support:

Item	*hot*	*mild*	*cool*	*windy*	*still*	*dry*	*rainy*	Yes	No
Support	$\frac{3}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{4}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{3}{7}$

- If at this point, we are going to continue mining for effective association rules, we will want rules with good confidence and good support. In particular, we will set a pair of thresholds τ_c and τ_s, and we will attempt to extract every rule whose confidence and support are above both of these thresholds.
- One way to attempt this is to just generate every rule and calculate both support and confidence, but in general, there will be far too many rules (as the number of rules is largely exponential in the number of attributes).
- To restrict the number of rules under consideration, we observe that, from a given k-itemset, we can construct a number of $k+1$ itemsets by adding each one of the attribute values that aren't already in this rule — but that the support can only go **down** by adding another attribute value. (Why?)

- So, from the 1-itemsets above, if any of them fail to pass our support threshold, then we can exclude that itemset from being a subset of an interesting 2-itemset.
- For example, if the support threshold is $\tau_s = 0.5$, the only interesting 1-itemsets are $\{still\}$, $\{dry\}$, and $\{\text{YES}\}$, and the only potentially interesting 2-itemsets are $\{still,dry\}$, $\{still,\text{YES}\}$, and $\{dry,\text{YES}\}$.
- We would then calculate the supports of those itemsets (and continue to $\{still,dry,\text{YES}\}$ if they continued to meet the support threshold of 0.5).
- Once we can no longer construct further itemsets that can meet the support threshold (because enough k itemsets have supports below the threshold so as to block all of the $k+1$ itemsets), we can generate all of the association rules from each itemset, calculate the confidences of these rules, and return with the ones that pass the confidence threshold τ_c.
- (We can construct about 2^k different association rules for a k-itemset. If we have many itemsets that meet the support threshold, there could still be too many rules to examine the confidences. In this case, we derive a similar procedure by observing that moving an attribute value from the antecedent to the consequent can only reduce the confidence of the rule. For a given k-itemset, we start with each of the k attribute values in the consequent one at a time, find the confidence, and proceed with rules that meet the confidence threshold.)

2. What does "correlation does not imply causation" mean? Why is it important to keep this adage in mind, when working in the field of Data Mining?

   - "Correlation does not imply causation" is an important adage which highlights the fact some events are reliably seen together, even though there isn't a causal relationship between the events.
   - There are a number of reasons why this might be the case:
     - The causal relationship exists, but not in the direction that we think; for example: "farmers usually have breakfast just before sunrise" leads us to wrongly conclude that "sunrise is caused by farmers having breakfast"
     - The casual relationship exists, but with an unseen intermediary; for example: "I usually have breakfast just after sunrise" leads us to wrongly conclude that "sunrise causes me to have breakfast" (whereas I might actually have breakfast just before work, whose starting time is correlated with sunrise)
     - There is an unseen factor which is the actual cause of both events; for example, "the month-over-month consumption of ice cream is correlated with observed incidences of sunburns" leads us to wrongly conclude either "eating ice cream causes sunburns" or "people eat ice cream when they have a sunburn", whereas the causal relationship is due to the exterior temperature ("people eat more ice cream and get more sunburns in summer")
     - The statistical correlation is entirely coincidental; there were a number of examples of this in the lecture, for example "the number of people who die annually from drowning by falling into a swimming pool is correlated with the number of films in which Nicholas Cage appears" leads us to wrongly conclude that "Nicholas Cage appearing in films causes people to drown in swimming pools" (!) or "people drowning in swimming pools causes Nicholas Cage to appear in films" (!!)
   - In Data Mining, we are looking for patterns about the data, and we are hoping to find patterns which are:
     (i) Valid: they are actually attested in the data; the data has support for this pattern
     (ii) Non-trivial: the pattern isn't immediately self-evident from the data, like: "the given instances are composed of attributes"; or, for something at least somewhat interpreted: "average household water use is correlated with the number of persons residing in the household"
     (iii) Previously unknown: finding such a rule tells us something we didn't already know about the data; many non-trivial patterns are already well-understood by experts, and so data

mining doesn't necessarily give any extra value in these circumstances (other than saving the experts a little bit of time)

- The point is that, for a given pattern, it is difficult to assess whether the pattern is **useful**, by meeting the three guidelines above (as well as being actionable in whatever context the data is from). An algorithm can posit pattens that it believes to be valid, but aren't actually useful for some reason.
- If we consider a large number of possible patterns (as we do in, say, Association Rule Mining), we are likely to find some which are entirely statistical quirks, and do not actually imply any causal relationship — hence the pattern is not actually useful, even though it might appear to be so.

3. What occurs in the training phase of a **neural network**? What occurs in the test phase?

   - In the training phase, we attempt to determine the weights on the branches in the network, so as to minimise the calculated **error function** between our predicted values at the output node(s), and the actual (true) values, across the training data.
   - Typically we use some gradient descent method over the partial derivatives of the error function (with respect to the various weights) to estimate the optimal weights iteratively.
   - Once we have our weights, we make predictions for a test instances by:
     - Multiplying the input node values by the corresponding weights to find the **pre-activation** values of the hidden nodes
     - Applying the activation function at the hidden nodes;
     - Iterating through each hidden layer;
     - Finally, applying the activation function at the output node(s) to observe the predicted value(s)