School of Computing and Information Systems The University of Melbourne
COMP30027 Machine Learning (Semester 1, 2021)

Sample Solutions: Week 12

1. [OPTIONAL] Consider the following dataset:

id	*apple*	*ibm*	*lemon*	*sun*	**label**
A	4	0	1	1	fruit
B	5	0	5	2	fruit
C	2	5	0	0	comp
D	1	2	1	7	comp
E	2	0	3	1	?
F	1	0	1	0	?

Treat the problem as an unsupervised machine learning problem (excluding the *id* and *label* attributes) and calculate the clusters according to (hard) *k*-**means** with *k* = 2, using the Manhattan distance, and instances A and F as the seeds.

This is an unsupervised problem, so we ignore (or don't have access to) the label attribute. (We're going to ignore id as well, because it obviously isn't a meaningful point of comparison.)

We begin by setting the initial centroids for our two clusters, let's say cluster 1 has centroid C1 = (4, 0, 1, 1) and C2 = (1, 0, 1, 0).

We calculate the (Manhattan) distances of each instance to each centroid:

$$d(A, C_1) = |4-4| + |0-0| + |1-1| + |1-1| = 0$$
$$d(A, C_2) = |4-1| + |0-0| + |1-1| + |1-0| = 4$$
$$d(B, C_1) = |5-4| + |0-0| + |5-1| + |2-1| = 6$$
$$d(B, C_2) = |5-1| + |0-0| + |5-1| + |2-0| = 10$$
$$d(C, C_1) = |2-4| + |5-0| + |0-1| + |0-1| = 9$$
$$d(C, C_2) = |2-1| + |5-0| + |0-1| + |0-0| = 7$$
$$d(D, C_1) = |1-4| + |2-0| + |1-1| + |7-1| = 11$$
$$d(D, C_2) = |1-1| + |2-0| + |1-1| + |7-0| = 9$$
$$d(E, C_1) = |2-4| + |0-0| + |3-1| + |1-1| = 4$$
$$d(E, C_2) = |2-1| + |0-0| + |3-1| + |1-0| = 4$$
$$d(F, C_1) = |1-4| + |0-0| + |1-1| + |0-1| = 4$$
$$d(F, C_2) = |1-1| + |0-0| + |1-1| + |0-0| = 0$$

Here, A is closer to cluster 1's centroid, B to cluster 1, C to cluster 2, D to cluster 2, F to cluster 2, and for E we have a tie.

Let's say we randomly break the tie for instance E by assigning it to cluster 2. (We'll see what would have happened if we'd assigned E to cluster 1 below.) So, cluster 1 is {A, B} and cluster 2 is {C, D, E, F}. We re-calculate the centroids:

$$C1 = \left(\frac{4+5}{2}, \frac{0+0}{2}, \frac{1+5}{2}, \frac{1+2}{2}\right) = (4.5,0,3,1.5)$$

$$C2 = \left(\frac{2+1+2+1}{4}, \frac{5+2+0+0}{4}, \frac{0+1+3+1}{4}, \frac{0+7+1+0}{4}\right) = (1.5,1.75,1.25,2)$$

Now, let's re-calculate the distances according to these new centroids:

$$d(A, C_1) = |4-4.5| + |0-0| + |1-3| + |1-1.5| = 3$$
$$d(A, C_2) = |4-1.5| + |0-1.75| + |1-1.25| + |1-2| = 5.5$$

d(B, C_1) = | 5 – 4.5 | + | 0 – 0 | + | 5 – 3 | + | 2 – 1.5 | = 3
d(B, C_2) = | 5 – 1.5 | + | 0 – 1.75 | + | 5 – 1.25 | + | 2 – 2 | = 9
d(C, C_1) = | 2 – 4.5 | + | 5 – 0 | + | 0 – 3 | + | 0 – 1.5 | = 12
d(C, C_2) = | 2 – 1.5 | + | 5 – 1.75 | + | 0 – 1.25 | + | 0 – 2 | = 7
d(D, C_1) = | 1 – 4.5 | + | 2 – 0 | + | 1 – 3 | + | 7 – 1.5 | = 13
d(D, C_2) = | 1 – 1.5 | + | 2 – 1.75 | + | 1 – 1.25 | + | 7 – 2 | = 6
d(E, C_1) = | 2 – 4.5 | + | 0 – 0 | + | 3 – 3 | + | 1 – 1.5 | = 3
d(E, C_2) = | 2 – 1.5 | + | 0 – 1.75 | + | 3 – 1.25 | + | 1 – 2 | = 5
d(F, C_1) = | 1 – 4.5 | + | 0 – 0 | + | 1 – 3 | + | 0 – 1.5 | = 7
d(F, C_2) = | 1 – 1.5 | + | 0 – 1.75 | + | 1 – 1.25 | + | 0 – 2 | = 4.5

What are the assignments of instances to clusters now? Cluster 1 {A, B, E} and cluster 2 {C, D, F}. (Note that we're at the same place now that we would have been if we'd randomly broke the tie for instance E to cluster 1 earlier.)

We calculate the new centroids based on these instances:

$$C1 = \left(\frac{4+5+2}{3}, \frac{0+0+0}{3}, \frac{1+5+3}{3}, \frac{1+2+1}{3}\right) \approx (3.67,0,3,1.33)$$

$$C2 = \left(\frac{2+1+1}{3}, \frac{5+2+0}{3}, \frac{0+1+1}{3}, \frac{0+7+0}{3}\right) \approx (1.33,2.33,0.67,2.33)$$

We re-calculate the distances according to these new centroids:

d(A, C_1) ≈ | 4 – 3.67 | + | 0 – 0 | + | 1 – 3 | + | 1 – 1.33 | ≈ 2.67
d(A, C_2) ≈ | 4 – 1.33 | + | 0 – 2.33 | + | 1 – 0.67 | + | 1 – 2.33 |≈ 6.67
d(B, C_1) ≈ | 5 – 3.67 | + | 0 – 0 | + | 5 – 3 | + | 2 – 1.33 | ≈ 4
d(B, C_2) ≈ | 5 – 1.33 | + | 0 – 2.33 | + | 5 – 0.67 | + | 2 – 2.33 | ≈ 10.67
d(C, C_1) ≈ | 2 – 3.67 | + | 5 – 0 | + | 0 – 3 | + | 0 – 1.33 | ≈ 11
d(C, C_2) ≈ | 2 – 1.33 | + | 5 – 2.33 | + | 0 – 0.67 | + | 0 – 2.33 | ≈ 6.33
d(D, C_1) ≈ | 1 – 3.67 | + | 2 – 0 | + | 1 – 3 | + | 7 – 1.33 | ≈ 12.33
d(D, C_2) ≈ | 1 – 1.33 | + | 2 – 2.33 | + | 1 – 0.67 | + | 7 – 2.33 | ≈ 5.67
d(E, C_1) ≈ | 2 – 3.67 | + | 0 – 0 | + | 3 – 3 | + | 1 – 1.33 | ≈ 2
d(E, C_2) ≈ | 2 – 1.33 | + | 0 – 2.33 | + | 3 – 0.67 | + | 1 – 2.33 | ≈ 6.67
d(F, C_1) ≈ | 1 – 3.67 | + | 0 – 0 | + | 1 – 3 | + | 0 – 1.33 | ≈ 6
d(F, C_2) ≈ | 1 – 1.33 | + | 0 – 2.33 | + | 1 – 0.67 | + | 0 – 2.33 |≈ 5.33

The new assignments of instances to clusters are cluster 1 {A, B, E} and cluster 2 {C, D, F}. This is the same as the last iteration, so we stop (and this is the final assignment of instances to clusters).

2. Repeat the previous question using "soft" k-means, and the "stiffness" $\beta = 1$.

Let's use initial centroids as A and F, like the previous question. We'll pick up from having calculated the distances of each point to the two initial centroids.

In "hard" k-means, we would assign each instance to whichever cluster is closer, but in "soft" k-means, we probabilistically assign each point according to the "softmax" function:

$$z_{ij} = \frac{e^{-\beta d(i,j)}}{\sum_i e^{-\beta d(i,j)}}$$

For each point, we are essentially normalizing, but rather than by the sum of the raw distances, we raise each negated distance to the power of e — this effectively handles the fact that smaller distances mean more similar instances, as well as making a transformation to account for differences in the distances.

For instance A (conveniently also the centroid of one cluster), the distances to the two clusters are 0 and 4. The probabilistic assignment is consequently:

$$z_{1A} = \frac{e^{-0}}{e^{-0} + e^{-4}} \approx 0.982$$

$$z_{2A} = \frac{e^{-4}}{e^{-0} + e^{-4}} \approx 0.018$$

Since the first cluster is much closer, this instance almost entirely placed in that cluster — however, unlike with “hard” k-means, it is slightly in the second cluster, too.

And so on for the other instances:

$$z_{1B} = \frac{e^{-6}}{e^{-6} + e^{-10}} \approx 0.982$$

$$z_{2B} = \frac{e^{-10}}{e^{-6} + e^{-10}} \approx 0.018$$

$$z_{1C} = \frac{e^{-9}}{e^{-9} + e^{-7}} \approx 0.119$$

$$z_{2C} = \frac{e^{-7}}{e^{-9} + e^{-7}} \approx 0.881$$

$$z_{1D} = \frac{e^{-11}}{e^{-11} + e^{-9}} \approx 0.119$$

$$z_{2D} = \frac{e^{-9}}{e^{-11} + e^{-9}} \approx 0.881$$

$$z_{1E} = \frac{e^{-4}}{e^{-4} + e^{-4}} \approx 0.5$$

$$z_{2E} = \frac{e^{-4}}{e^{-4} + e^{-4}} \approx 0.5$$

$$z_{1F} = \frac{e^{-4}}{e^{-0} + e^{-4}} \approx 0.018$$

$$z_{2F} = \frac{e^{-0}}{e^{-0} + e^{-4}} \approx 0.982$$

Instance E is still tied between the two clusters, but that doesn’t create any issues here.

If you look closely, you can see that the probabilistic assignment doesn’t depend on the distance exactly, but rather, the difference between the distances. For example, instances C and D get soft-assigned exactly the same way, even though C is two units closer to both clusters.

We now update the centroids, just like with “hard” k-means, except this time it is a weighted average:

$$\begin{aligned} C_1^{(1)}: & \frac{0.982A + 0.982B + 0.119C + 0.119D + 0.5E + 0.018F}{0.982 + 0.982 + 0.119 + 0.119 + 0.5 + 0.018} \\ & = \frac{1}{2.72}[(0.982)\langle 4,0,11\rangle + (0.982)\langle 5,0,5,2\rangle + (0.119)\langle 2,5,0,0\rangle \ldots] \\ & = \frac{1}{2.27}[\langle 10.21,0.833,7.53,4.28\rangle] \\ & \approx \langle 3.75,0.307,2.77,1.57\rangle \end{aligned}$$

$$\begin{aligned} C_2^{(1)}: & \frac{0.018A + 0.018B + 0.881C + 0.881D + 0.5E + 0.982F}{0.018 + 0.018 + 0.881 + 0.881 + 0.5 + 0.982} \\ & \approx \langle 1.46,1.88,1.06,2.05\rangle \end{aligned}$$

With our new centroids, we are set to iterate. I’ll summaries the distances and corresponding z values in a table, for convenient reading:

	$d(1,j)$	$d(2,j)$	z_{1j}	z_{2j}
A	2.89	5.53	0.933	0.067
B	4.21	9.41	0.995	0.005
C	10.79	6.77	0.018	0.982
D	11.64	5.59	0.002	0.998
E	2.87	5.41	0.927	0.073
F	6.40	4.45	0.124	0.876

We can see that there are some instances where we have become more confident — like B and D — and others where we appear to be changing our mind — like E and F.

$$C_1^{(1)}: \frac{0.933A + 0.995B + 0.018C + 0.002D + 0.927E + 0.124F}{0.933 + 0.995 + 0.018 + 0.002 + 0.927 + 0.124}$$
$$\approx \langle 3.58,0.31,2.94,1.29\rangle$$

$$C_2^{(1)}: \frac{0.067A + 0.005B + 0.982C + 0.998D + 0.073E + 0.876F}{0.067 + 0.005 + 0.982 + 0.998 + 0.073 + 0.876}$$
$$\approx \langle 1.43,2.30,0.729,2.38\rangle$$

The centroids have only moved a small way (a couple of tenths in most cases), but that causes our predictions to change further:

	$d(1,j)$	$d(2,j)$	z_{1j}	z_{2j}
A	2.68	6.52	0.979	0.021
B	4.23	10.52	0.998	0.002
C	10.77	6.38	0.012	0.988
D	12.19	5.62	0.001	0.999
E	1.96	6.52	0.990	0.010
F	5.83	5.38	0.387	0.613

Already, we can see that we are quite certain about most of the instances, and in the next couple of iterations, instance F will get "pulled into" Cluster 1, at which point the method will gradually converge.

(If you're curious, the two cluster centroids eventually become approximately $\langle 3,0,2.5,1\rangle$and $\langle 1.5,3.4,0.5,3.5\rangle$ — after about 6 iterations — you might like to compare these to the corresponding "hard" centroids.)

3. What is logic behind the EM algorithm, when used for clustering?

Basically, that we start with a random (or uniform) guess, and then we progressively improve our guess by evaluating the expected likelihood on the given data.

Another way of looking at it: our initial estimate probabilistically defines some labels for the training data; we can then use the counts over these labels to re-estimate the corresponding elements of the model (typically probabilities).

(i) Explain the significance of the "E" step, and the "M" step.

Expectation: assign weighted labels to the training data, and use these to calculate the log-likelihood function

Maximization: re-estimate the parameter based on these labels

(ii) Identify the "E" and "M" steps in GMM method.

In the *Expectation* step in GMM method, we start from some assumed parameters (μ, σ) for each distribution and with the use the log-likelihood of the GMM, we calculate the responsibility (γ) for each instance. In other words, we estimate how much each distribution has been responsible for generating each instance.

In the *Maximization* step, we update our parameters (μ, σ) for each distribution to improve the log-likelihoods.

We keep iterating between these two steps until the algorithm converges (e.g. the log-likelihood cannot be improved anymore).

4. Revise the concept of *unsupervised* and *supervised* **evaluation** for clustering evaluation

(i) Explain the two main concepts that we use to measures the goodness of a clustering structure without respect to external information.

The two main concepts that we check in unsupervised clustering evaluation are the clusters *cohesiveness* and *separability*. We want the members of each clusters to be as integrated and close to each other as possible and meanwhile we want the clusters to be as separate and independent as possible from other clusters. Using the SSE metric, we can use W-SSE (Within Cluster Sum of Squared Errors) to measure intra-cluster cohesion and B-SSE (Between Cluster Sum of Squared Errors) to measure inter-cluster separation.

(ii) Explain the two main concepts that we use to measure the how well do cluster labels match externally supplied class labels.

The two main factors in supervised clustering evaluation metrics are *homogeneity* and *completeness*. Homogeneity measures if all the elements of each cluster have the same true label. We can use measure the homogeneity of a cluster using Entropy and Purity. Completeness metric measure if all the members of a class are assigned to the same cluster.

5. Revise the difference between **supervised**, **semi-supervised** and **unsupervised** machine learning. When do we use semi-supervised learning?

The main big difference lays in the presence of the label. In supervised learning (e.g., classification and regression) the train (and evaluation) datasets include the label for all instances. Supervised learning methods use that information to find a pattern and also to evaluate themselves.

In unsupervised learning the instances do not include any label, and the algorithm needs to find patterns in the dataset without knowing the label (or the number of the labels). Also, the evaluation of the results is not necessarily based on the pre-existing labels. Instead, these methods use other metrics (e.g., cohesion and separation) to evaluate their performance.

The semi-supervised learning something in between. In these methods, we have a small proportion of labelled data that can be used to gain some information about the data (e.g., possible number of the labels/clusters) the method then expand this information to find the patterns over lager datasets (from the same distribution).

In semi-supervised learning, we have a small number of labelled instances, and a large number of unlabeled instances. Typically, this means that we don’t have enough data to train a reliable classifier (purely supervised), but we can potentially leverage the labelled instances to build a better classifier than a purely unsupervised method might come up with.

(i) What is self-training?

Self-training is a method of using a learner to build a training data set as follows:

- Train the learner on the currently labelled instances
- Use the learner to predict the labels of the unlabeled instances
- Where the learner is very confident, add newly labelled instances to the training set
- Repeat until all instances are labelled, or no new instances can be labelled confidently.

(ii) What is the logic behind active learning, and what are some methods to choose instances for the oracle?

In active learning, the learner is allowed to choose a small number of instances to be labelled by oracle (a human judge).

The idea here is two–fold: many instances are easy to classify; and a small number of instances are difficult to classify but would be easier to classify with more training data.

In some *cases*, the instances to be given to the oracle are selected by measuring the *uncertainty*. In other cases, we use different models and select the instances for query that raised the *highest disagreements*.

6. One of the strategies for Query sampling was query-by-committee (QBC). Using the equation below, which captures vote entropy, determine the instance that our active learner would select first.

$$x^*_{VE} = \underset{x}{\mathrm{argm}ax}(-\sum_{y_i} \frac{V(y_i)}{C} log_2 \frac{V(y_i)}{C})$$

Respectively y_i, $V(y_i), and\ C$ are the possible labels, the number of "votes" that a label receives from the classifiers, and the total number of classifiers.

classifier	**Instance 1**			**Instance 2**			**Instance 3**		
	y_1	y_2	y_3	y_1	y_2	y_3	y_1	y_2	y_3
C_1	0.2	0.7	0.1	0.2	0.7	0.1	0.6	0.1	0.3
C_2	0.1	0.3	0.6	0.2	0.6	0.2	0.21	0.21	0.58
C_3	0.8	0.1	0.1	0.05	0.9	0.05	0.75	0.01	0.24
C_4	0.3	0.5	0.2	0.1	0.8	0.1	0.1	0.28	0.62

In the QBC method we have a group of classifiers trained over a fixed training set, and the instance that results in the highest disagreement amongst the classifiers, is selected for querying.

In this example, for each instance, we calculate the total number of votes that each class label receives:

classifier	Instance 1 Votes			Instance 2			Instance 3		
	y_1	y_2	y_3	y_1	y_2	y_3	y_1	y_2	y_3
C_1	0	1	0	0	1	0	1	0	0
C_2	0	0	1	0	1	0	0	0	1
C_3	1	0	0	0	1	0	1	0	0
C_4	0	1	0	0	1	0	0	0	1
	V(1)=1	V(2)=2	V(3)=1	V(1)=0	V(2)=4	V(3)=0	V(1)=2	V(2)=0	V(3)=2

We have 4 classifiers in total, and after placing the vote values in the vote entropy, we get the following for each instance:

Instance 1: $H = -\left(\frac{1}{4}log_2\frac{1}{4} + \frac{2}{4}log_2\frac{2}{4} + \frac{1}{4}log_2\frac{1}{4}\right) = 1.5$

Instance 2: $H = -(1log_2 1) = 0$

Instance 3: $H = -\left(\frac{2}{4}log_2\frac{2}{4} + \frac{2}{4}log_2\frac{2}{4}\right) = 1$

The instance that we select is instance 1, for which the classifiers have the highest disagreement. This sample is most difficult to classify and may lie on the boundary between the three classes, therefore by querying this instance, we might learn more about the data space.