School of Computing and Information Systems
The University of Melbourne
COMP90049 Introduction to Machine Learning (Semester 1, 2021)

Sample Solutions: Week 11

1. When do we use semi-supervised learning? What is self-training?

   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.

   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.

2. 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*.

3. 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.

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

   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.

	Instance 1			Instance 2			Instance 3		
classifier	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.

4. Given the following univariate dataset, calculate a statistical model based on the assumption that your data is coming from a normal distribution. Determine whether the instance x=1.2 is anomalous or not if we use the boxplot test?

$$X = \{2, 2.5, 2.6, 3, 3.1, 3.2, 3.4, 3.7, 4, 4.1, 4.8\}$$

In statistical anomaly detection, we learn a model that fits the given data set, and then identify the objects in low probability regions of the model as anomalies. Since we are using the boxplot test, we need to calculate the z-score for x=1.2 and if the z-score is beyond $\mu \pm 3\sigma$, then we determine that it is an outlier. Remember that $\mu \pm 3\sigma$ contains 99.7% of the samples, so anything that occurs outside this range, would have a low probability of belonging to the model.

Since we are assuming a normal distribution, we calculate μ and σ.

$$\mu = \frac{1}{n}\sum_i x_i = \frac{1}{11}(2 + 2.5 + 2.6 + 3 + 3.1 + 3.2 + 3.4 + 3.7 + 4 + 4.1 + 4.8) = 3.31$$

$$\sigma^2 = \frac{1}{n}\sum_i (x_i - \mu)^2$$

$$= \frac{1}{11}((2-3.31)^2 + (2.5-3.31)^2 + (2.6-3.31)^2 + (3-3.31)^2 + (3.1-3.31)^2 + (3.2-3.31)^2 + (3.4-3.31)^2 + (3.7-3.31)^2 + (4-3.31)^2 + (4.1-3.31)^2 + (4.8-3.31)^2) = 0.59$$

$$\mu = 3.31, \sigma = 0.77$$

Now, let's calculate the z-score for x=1.2 to see how far away it is from the mean value:

$$z = \frac{|x-\mu|}{\sigma} = \frac{|1.2-3.31|}{0.77} = 2.74 < 3$$

The z-score indicates that x=1.2 is still within the $\mu \pm 3\sigma$ boundary and wouldn't be considered an outlier.

We can also check the boundary $\mu \pm 3\sigma$:

$$3.31 \pm 3 \times 0.77 = [1,5.62]$$

We can see that x=1.2 is in this range and therefore it is NOT an outlier.

5. Given the following univariate dataset, determine the outlier score for instances (x=0.5) and (x=4) using the following strategies:

Dataset = {1,1.05, 1.1, 1.15, 1.2, 1.21, 1.3, 1.4, 1.45, 1.5, 4.55, 5.6, 6.8, 7.58, 8.6, 9.7, 10.3, 11.4, 12.3,13.5}

(a) Inverse Relative density using 2-NN (Manhattan distance)

Density based anomaly detection assumes that "Outliers are objects in regions of low density." By using relative density, we are taking into account the compactness of each cluster of objects. Therefore, an instance will be penalized if its nearest neighbors are in a high-density region.

$$relative\ density(x,k) = \frac{density(x,k)}{\frac{1}{k}\sum_{y \in N(x,k)} density(y,k)}$$

where the density is calculated as:

$$density(x,k) = \left(\frac{1}{k}\sum_{y \in N(x,k)} distance(x,y)\right)^{-1}$$

For the first sample x=0.5, its density with respect to its 2 nearest neighbours can be calculated using:

$$density(x = 0.5\ , k = 2) = \left(\frac{1}{2}(|0.5 - 1| + |0.5 - 1.05|)\right)^{-1} = 1.9$$

Nearest neighbors of x=0.5 are 1 and 1.05, now we need to find the nearest neighbors for x=1 and x=1.05

$$relative\ density(x = 0.5, k = 2) = \frac{1.9}{\frac{1}{2}(density(x = 1, k = 2) + denisty(x = 1.05, k = 2))}$$

$$density(x = 1, k = 2) = \left(\frac{1}{2}(|1 - 1.05| + |1 - 1.1|)\right)^{-1} = 13.3$$

$$density(x = 1.05, k = 2) = \left(\frac{1}{2}(|1.05 - 1| + |1.1 - 1.05|)\right)^{-1} = 20$$

Replacing the neighbors' densities in the relative density formula, we get:

$$relative\ density(x = 0.5, k = 2) = \frac{1.9}{\frac{1}{2}(13.3 + 20)} = 0.11$$

As we mentioned, outliers occur in low density regions, so the inverse of relative density for x=0.5 is 1/0.11 = 9.1, i.e., high outlier score.

For the second sample x=4, the density of x with respect to its 2 nearest neighbors can be calculated using:

$$density(x = 4\ , k = 2) = \left(\frac{1}{2}(|4 - 4.55| + |4 - 5.6|)\right)^{-1} = 0.93$$

Nearest neighbors of x=4 are 4.55 and 5.6, now we need to find the nearest neighbors for x=4.55 and x=5.6:

$$relative\ density(x = 4, k = 2) = \frac{0.93}{\frac{1}{2}(density(x = 4.55, k = 2) + denisty(x = 5.6, k = 2))}$$

$$density(x = 4.55, k = 2) = \left(\frac{1}{2}(|4.55 - 5.6| + |4.55 - 6.8|)\right)^{-1} = 0.61$$

$$density(x = 5.6, k = 2) = \left(\frac{1}{2}(|5.6 - 4.55| + |5.6 - 6.8|)\right)^{-1} = 0.89$$

Replacing the neighbors' densities in the relative density formula, we get:

$$relative\ density(x = 0.5, k = 2) = \frac{0.93}{\frac{1}{2}(0.61 + 0.89)} = 1.24$$

Again, we calculate the inverse of relative density for x=4 which is 1/1.24 = 0.81, i.e., a low outlier score.

Since we clearly have a very high-density cluster {1,1.05, 1.1, 1.15, 1.2, 1.21, 1.3, 1.4, 1.45, 1.5}, the instance x=0.5 is penalized more because its nearest cluster is very compact. On the other hand, for x=4, the sample is close to a low density cluster {4.55, 5.6, 6.8, 7.58, 8.6, 9.7, 10.3, 11.4, 12.3,13.5}, therefore it is not penalized as much as =0.5 although both test samples have equal distance to their first nearest neighbors.

(b) Distance to 2nd nearest neighbor (Manhattan distance)

Proximity based anomaly detection assumes that "an object is an anomaly if the nearest neighbors of the object are far away, i.e., the proximity of the object significantly deviates from the proximity of most of the other objects in the same data set. For the first sample x=0.5, the distance of x with respect to its 2nd nearest neighbor can be calculated as:

$$distance(x = 0.5\ , x = 1.05) = |0.5 - 1.05| = 0.55$$

For the second sample x=4, the distance of x with respect to its 2nd nearest neighbor can be calculated as:

$$distance(x = 4\ , x = 5.6) = |4 - 5.6| = 1.6$$

This method cannot capture the variability in cluster sizes, therefore it considers x=4 to have a higher outlier score than x=0.5, although x=4 is close to a very sparse cluster.

6. In Assignment 1 we worked with the 'animals' dataset. Suggest a suitable method to detect anomalies among animal instances. Would you use a supervised, semi-supervised or unsupervised approach? Can you think of a way to make anomaly detection more reliable?

The animal dataset (zoo.features and zoo.labels) was consist of 101 instances. Each instance corresponds to an animal and is characterized by 16 features. These animals are categorized into 7 groups (mammal, bird, reptile, fish, amphibian, insect, invertebrate). Anomalies are instances (animals) that that don't match the characteristics of any of the groups very well.

For example, the Platypus can be considered an anomaly (or outlier). Platypus shows some features on mammal (e.g., hair, milk) and some features of birds (e.g., eggs) and some features of amphibians (e.g., venomous). For this exercise we assume that (I) we have a machine learning model trained to group "normal" animals into one of 7 groups; and (II) that the anomalous Platypus was not part of the training set.

We have three categories of anomaly detection options: Supervised, Semi-supervised and Unsupervised methods.

We use *supervised* methods in cases where we have access to labels for both 'normal' data and anomalies. In the case of our 'zoo' dataset, we have the labels for the animal categories, but no label for 'normal' and 'not normal' instances. So, we cannot use the supervised anomaly detection method.

For *semi-supervised* methods, we have access to a dataset of 'normal' instances. We train a model on the 'normal' instances and use the model to indirectly detect the outliers. In our 'zoo' dataset, we can assume that our available labelled dataset includes only 'normal' instances. We train a supervised model on our 'normal' data (e.g., a Naive Bayes Classifier or Logistic Regression model). For Naive Bayes, outlier could be an instance where the posterior probability ($p(y|x)$) for an anomalous x (like the Platypus) has high entropy or: is evenly distributed among several classes. Given the Platypus features, we expect the classifier to assign similar probabilities to 'mammal' (because of feature: milk), 'bird' (because of feature: eggs) and amphibian (because of feature: venomous).

Taking the above approach can lead to a high false positive rate: a classifier may also produce high entropy distributions for noisy or just slightly atypical (but not anomalous) instances. To be more confident about the outcome of our anomaly detection method we can use an ensemble method. Using several different supervised classifiers, we can check the similarities between their detected outliers. We assume that if multiple classifiers agree that an instance doesn't 'fit' any of the classes we can be more certain about the output of our system. If all (or most) of our supervised models detect one instance as an anomaly, we can be more confident that the instance is indeed an outlier.

If we are not quite convinced that the given dataset is a good representative of 'normal' instances (or we have no access to the labels y), we can simply ignore the labels and treat the dataset as an *unsupervised* set.

In this case, we can for example use cluster-based outlier detection. Again, a version of ensemble/voting method can lead to more reliable predictions. For example, using the clustering-based method, we can run K-means with multiple random seeds or varying K. In this case, we can decide on a threshold when an instance is considered an outlier (e.g., consider as outliers the top-n instances with furthest distance from any centroid as outlier, using 'relative distance'). Each clustering algorithm will make a binary decision (outlier/not outlier) for every instance and given these decisions we can use voting. Alternatively, each clustering algorithm could return (relative) distance values directly, and we can use the average distance, or use the maximum distance by any model, or use the minimum distance by any model, and threshold the value.