National University of Singapore
School of Computing
CS3244: Machine Learning
Solution to Tutorial 1

The Learning Paradigms, and k-NN

1. **Learning Paradigms.** Describe different instances of learning problems for the following scenarios. For each scenario, describe a *supervised*, *unsupervised* and *reinforcement* learning problem. For one of the problems, formalize the given components in the learning problem: *input*, *output*, *data*. You need not describe the *hypothesis* nor the *target function* (to think about: why?).

   (a) In NUS (or other university) student domain. Students have problems, many of them, and you are the target audience. Describe problems that you encounter on a daily, weekly or semesterly basis.

   There are many possible model answers, your answer is likely to vary. Can you think of other problems?

   Supervised. *In supervised learning, for each instance, we get a correct output. For example, given a student's transcript and a list of modules that the student is currently taking, predict their performance in one module in terms of CAP. This particular task as a regression task, since the output is of a continuous value.*

   Unsupervised. *In unsupervised learning, we have no outputs (labels) given to us – the task is to explore and make groupings from the data based solely on the input. For example, given a historical mapping of which students took which NUS module, cluster the modules by similarity. This particular task models modules not in terms of content (e.g., host faculty), but by social signals – similar to the notion of collaborative filtering.*

   Another example is to cluster students based on historical data of whether they have attended other classes together.

   Reinforcement. *In reinforcement learning, we get some form of output but it does not correspond directly to the correct output for each instance. For example, we could get a single output after a batch of instances is processed (as commonly seen in games, where the final game win/loss outcome is the eventual output of a series of instance moves). For example, as a Year 1 student, I would like to choose modules that maximise my starting income upon graduation. If we consider the choice of modules during each of the nominal eight semesters as a problem instance, this would constitute a reinforcement learning problem.*

   In this final task, the **input** *might be the historical enrollment records of past students (and possibly their performance within each module), and the* **output** *would be their starting salary.* **Data** *is the set of inputs that correspond to the resultant output.*

(b) Transshipment Logistics. One of Singapore's mainstay sources of income for decades[1] has been transshipment and the logistics associated with this. Hypothesize problems that occur in this scenario.

There are many possible model answers, your answer is likely to vary. Can you think of other tasks?

We give just one instance of supervised prediction: vessel arrival prediction. This is a real problem that is discussed also in scholarly publications, in addition to being a problem sought after in industry. Given some **input** *attributes* **x** *that describe a vessel such as its tonnage, weather conditions on route, ports of call, owner,* **output** *the expected variation from the scheduled ETA of the vessel. Since this is a supervised problem, the* **data** *would consist of many tuples of such instances.*

We note that sometimes the resolution of the problems can change the learning problem paradigm. For the same set of actors as in the previous, let us consider the **vessel routing problem**. *Here instead of predicting a value over the entire route of a vessel's course, we need to decide among alternate routes for a ship at a port of call. Each port of call a ship stops at constitutes a new instance of the problem, and there is no immediate output – only when the ship arrives at the final destination port is there an output. This makes this problem a reinforcement learning problem.*

2. k Nearest Neighbor

(a) Suppose you are given the following data (as shown in Table 1) where x and y are the two input variables and *Origin* is the dependent variable.

x	y	*Origin*
-1	1	-
0	1	+
0	2	-
1	-1	-
1	0	+
1	2	+
2	2	-
2	3	+

Table 1: The dataset for kNN.

Figure 1 is a scatter plot which shows the above data in 2D space.

Suppose that you want to predict the class of new data point at $x = 1$ and $y = 1$ using Euclidian distance with 3-NN. Which class does this data point belong to? Does the classification change if we change the algorithm to 7-NN?

In 3-NN (0, 1), (1, 0) and (1, 2) will be the nearest neighbours, and all points except

[1] Accordingly to Wikipedia, as of 2016, Singapore remains the world's busiest transshipment port.

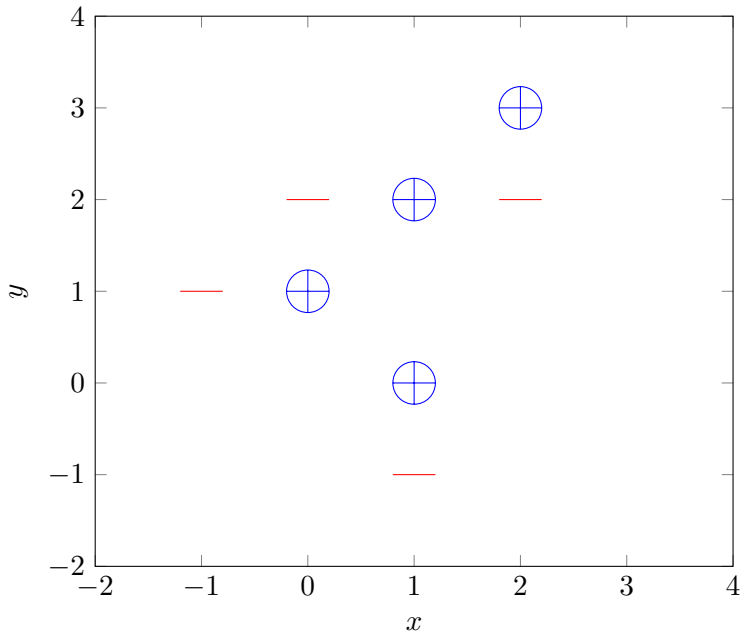

Figure 1: Data points of k-NN dataset on 2D space.

for the one at (2, 3) will be neighbours in 7-NN. Therefore, 3-NN will classify the new data point at (1, 1) as "+" whereas 7-NN will classify it as "−".

(b) Suppose you are given the following images (Figure 2). Your task is to compare the values of k in k-NN in each image where k_l, k_c and k_r are the left, center and right subfigures below, respectively.

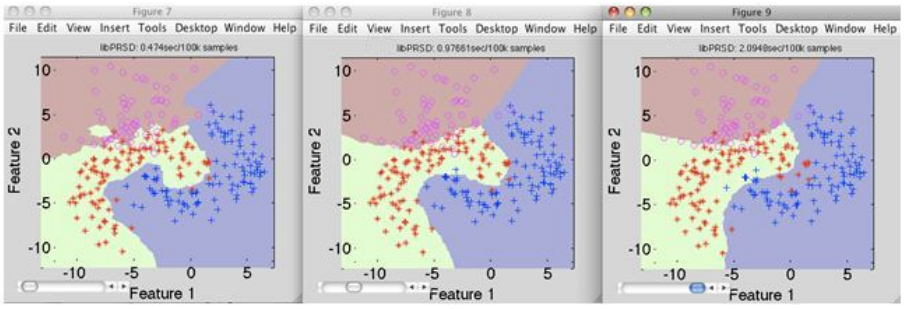

Figure 2: kNN runs for $k = 1, 2, 3$. Which is which?

Which one of k_l, k_c and k_r is the largest k and which one is the smallest?

The larger the k is, the smoother the decision boundaries will be. Overfitting (covered later in Week 05) will occur less as k increases. So simply by looking carefully at the decision boundaries, we can conclude $k_l < k_c < k_r$.

(c) Suppose that you have trained a kNN model and now you want to perform prediction

on test data. Before executing the prediction (*a.k.a.* inference) task, you want to calculate the time that kNN will take for predicting the class for test data. Let's denote the time to calculate the distance between 2 observations as t. What would the time taken by 1-NN be if there are m (some very large number) observations in the training data? What about for 2-NN or 3-NN? (We only consider the time used for calculating distances.)

Each time we need $m \times t$ time to calculate all the distances, so regardless of the value of k in kNN, we always need $m \times t$ time to predict one test example.

Food for thought: can you come up with ideas to speed this up? The numbers above assume a naïve strategy to calculate all above is a brute force calculation.

3. **Analysing k-NN Inference** Alice and Bob have proposed 2 ways of doing k-NN inference. Both algorithms are explained below. There are n number of training samples and the time taken to calculate the distance between two samples is $O(d)$.

   - **Algorithm by Alice**
     (a) Initialize $S[i] = 0$ for all the training samples. Here $i \in \{1, 2, 3, ..., n\}$
     (b) For each training sample, compute $D[i]$. Here $D[i]$ denotes the distance between training sample i and the new observation.
     (c) Iterate k number of times through all the training samples to do the following procedure in every iteration. Find the smallest $D[i]$ with the condition $S[i] = 0$. After the full scan through all the samples, mark $S[min] = 1$. Here min is the location where $D[min]$ is small and $S[min] = 0$.
     (d) Return k samples with indices where $S[i] = 1$.
   - **Algorithm by Bob**
     (a) Initialize $S[i] = 0$ for all the training samples. Here $i \in \{1, 2, 3, ..., n\}$
     (b) Iterate k number of times through all the training samples to do the following procedure in every iteration. Find the distance between each sample and the new observation. This steps are only done for the locations is' with $S[i] = 0$. The minimum distance location will be marked with $S[min] = 1$.
     (c) Return k samples with indices where $S[i] = 1$.

(a) Verify whether the algorithms of Alice and Bob are correct or not? If they are correct, give the running time for single inference in terms of n, d, k. Which is the best algorithm with respect to the running time?

Both inference algorithms are correct. The algorithm of Alice runs in $O(n(d+k))$ and the algorithm of Bob runs in $O(ndk)$. Alice's one is asymptotically faster than Bob's one.

(b) Propose a way to improve the best algorithm in part (a). What is the running time of the new algorithm?

One way of improving the algorithm of Alice is to keep a BST with k nodes where the BST tracks the top-k smallest distances. This can be done using the following steps.

(a) Insert first k distances to a balanced BST. For the remaining $n - k$ distances follow the next steps.

(b) If the distance is greater than the maximum value of BST, no change needed for BST.

(c) Else, remove the maximum element from the BST and insert the current distance.

This would reduce the running time to $O(n(d + lgk))$.

Think of a way to reduce the running time even further to $O(nd)$. (Hint : Use Quick Select algorithm to find the k th smallest element of the distance list.)

4. ***k*-NN** Suppose you have to do a classification problem to predict whether it will rain on an area based on the available dataset using the k-NN algorithm. The input variables are the humidity which varies between $50 - 90\%$, and the average temperature which ranges between $25 - 35$ degrees Celcius. Do you think applying k-NN **directly** will yield a good prediction result? If not, what improvement will you propose?

   Considers the difference between the ranges of the two input variables. The humidity range is 0.4*, while the range of the temperature is* 10 *degrees Celcius. If we apply the k-NN directly, the Euclidian distance between points will be dominated by the temperature difference. Hence, this drastically reduces the contribution of the humidity variable in the k-NN algorithm.*

   An improvement can be made by applying techniques such as scaling (normalization or standardization) of both variables. You will learn this technique in the future lecture.

5. (Optional) **The Netflix Prize** The Netflix Prize was a competition held by Netflix, to improve its algorithm for recommending movies to its users. This was formalized by having a system predict a customer's numeric rating of a target movie. The winning entry was one that used collaborative filtering, which is a method that is based on the assumption that people who agreed in the past will agree in the future. It looked at the historical ratings that users had given movies, but not the features of the movie and user (e.g. genre, year, director, actor, etc.). However, it was never adopted. Can you think of several reasons why?

   There are many possible model answers, your answer is likely to vary.

   Other issues important in recommender systems:

   - **Diversity***: how different are the recommendations?*
     - *If you like 'Battle of Five Armies Extended Edition', recommend Battle of Five Armies?*
     - *Even if you really really like Star Wars, you might want non-Star-Wars suggestions.*
   - **Persistence***: how long should recommendations last?*
     - *If you keep not clicking on 'Hunger Games', should it remain a recommendation?*
   - **Trust***: tell user why you made a recommendation.*
   - **Social recommendation***: what did your friends watch?*
   - **Freshness***: people tend to get more excited about new/surprising things.*
     - *Collaborative filtering does not predict well for new users/movies.*
     - *New movies don't yet have ratings, and new users haven't rated anything.*