CptS 570: Assignment 3

Abrar Akhyer Abir

Question 1:
a) We are given $x = (x_1, x_2, ...x_d)$ and $z = (z_1, z_2, ...z_d)$, two point in high dimensional space. We can write the distance function as:

$$F(x, z) = \sum_{i=1}^{d} (x_i - z_i)^2$$

Now, we can write the above function as:

$$E(F) = \frac{1}{d} \times \sum_{i=1}^{d} (x_i - z_i)^2$$

Now,

$$\begin{aligned} F(E[X - Z]) &\leq E[F] \\ \Rightarrow F(E[X] - E[Z]) &\leq E[F] \\ \Rightarrow (\frac{1}{d}\sum_{i=1}^{d} x_i - \frac{1}{d}\sum_{i=1}^{d} z_i)^2 &\leq \frac{1}{d} \times \sum_{i=1}^{d} (x_i - z_i)^2 \\ \Rightarrow (\frac{1}{\sqrt{d}}\sum_{i=1}^{d} x_i - \frac{1}{\sqrt{d}}\sum_{i=1}^{d} z_i)^2 &\leq \sum_{i=1}^{d} (x_i - z_i)^2 \end{aligned}$$

b) The equation, we derived for part (a), will provide us a way to make the calculation easier and also computationally less expensive. From this equation, we will not required to calculate distance for all features. We can sum up all the feature the test point and also for the existing point. Then we only need to perform subtraction once and we will be able to find the distance. We will achieve $n + d$ complexity instead of having nd. Here, n is the number of data points.

Question 2:
The family of closest neighbor algorithms that are based on Locality Sensitive Hashing is discussed by the authors in this study. A hashing-based technique for the d-dimensional Euclidean space is also discussed by the authors. As one of the effective algorithms for the approximate and precise closest neighbor problems, they mentioned the LSH. Locality Sensitive Hashing is used to address the dimensionality issue that affects the nearest neighbor method. The nearest neighbor approach becomes quite computationally expensive for high-dimensional datasets. Existing

techniques do not outperform a linear time algorithm when dealing with greater dimensional spaces.

The problem is attempted to be resolved utilizing the approximate nearest neighbor method via locality sensitive hashing. LSH accomplishes this by lowering the dimensionality of the data by hashing related input items to the same "buckets" with a high probability. It seeks to maximize the likelihood of a collision for related objects. The author also covered the application of LSH to Euclidean distance. The approach accomplishes a separation that is close to ideal using the collision probability. In the worst case scenario, this allows the closest NN problem to be reduced to the identical NN problem. The running time is $\frac{1}{c^2}$ (maximum) for a large number of data, where c is a constant.

Question 3:
It is possible to construct a decision tree from the given rules set. For the root node, we will go through each rule and take the rule that give us the highest information gain. We assume this rule is associated with an attribute. So, for each child nodes, we will ignore the rules that associated with this attribute and check rest of the rules for highest information gain. We also need to remember that the rules will be applied to a subset of our dataset that was filtered by the previous node's rule. However, if we were need to use all the rules from the rules set to construct the decision tree then it would not be possible because we will ignore the related rules from an attribute because it has already been chosen.

Question 4:
In this essay, the authors contrast discriminative and generative classifiers. Both logistic regression and the Naive Bayes classifier were chosen, one from each category. Authors have challenged the widespread assumption that discriminatory classifiers are usually always preferable. However, the authors of this work demonstrated how this condition could change with more training data. Although the authors concentrated on the comparison of naive Bayes and logistic regression, they also noted that this research might be expanded to include other generative-discriminative combinations of models.

The joint probability distribution of the input and output serves as the basis for the generative classifier's learning. Then it chooses the output that is the most probable given the distribution of the input data. Contrarily, the discriminative approach picks up on variations in the feature vectors. The generative technique performs better when the training data are small since it looks for relationships between the output and all of the distribution's properties. However, discriminative learning may be more effective to employ if there is a lot of data because it will eventually have lower error. A discriminative linear classifier is said to need O training examples in order to reach its asymptotic error $O(n)$. In comparison, a generative linear classifier requires $O(logn)$.

Question 5:
a) If the data is infinite, the logistic regression will function more effectively. In the discriminative process, error grows as data rises, but in the case of generative learning, error becomes saturated.
b) If the training data approaches infinity and the features are dependent, Naive Bayes will not perform properly under the presumption that the training data does not satisfy the Naive Bayes assumption. Instead, logistic regression would do better in this situation because there are nearly infinite numbers of training samples.
c) As one sort of generative learning method, the Naive Bayes can be used to determine P(X) using the following equation:

$$P(X) = \sum P(X|Y = y)P(Y = y)$$

Here, y is the class label.
d) The learned parameters of a logistic regression classifier cannot be used to compute P(X). This is due to the fact that logistic regression is a discriminative learning method and that we learn the conditional distribution from it rather than the general distribution.

Question 6:
The author of this paper discussed the relative value of ensemble approaches over single classifiers. Although Bayesian averaging is the original Ensemble approach, error correcting output coding, Bagging, and boosting work better. The classifiers in the ensemble approach are a collection of classifiers from which the individual decisions are created, sorted, and then aggregated using techniques like weighted or unweighted voting. Three key factors explain why the ensemble technique should be used instead of a single classifier for determining performance.

When the training data is small, the first cause is statistical. There is a danger of choosing incorrectly for the statistical hypothesis when the sample size is limited. ensemble thereby assists in calculating the average of all classifiers and lowers the possibility of incorrect categorization. Computation is the second reason. An ensemble can be designed to prevent algorithms from becoming stranded in local optima, which happens to them occasionally. By beginning from a variety of starting places, Ensemble is able to locate the optimal location. The ensemble can also aid in having a better representation when testing the hypothesis, which brings us to our third and last reason.

The author also discusses the techniques for building ensembles, including Bayesian voting, modifying the training example, modifying the input characteristics, modifying the output targets, and adding randomization. The author then contrasted the various approaches. Dietterich (2000) asserts that ADABOOST frequently produces the best outcomes. Trees that are randomized and bagged also function well. However, randomization would be more effective than Bagging by Bootstrapping if the dataset were really large. Another benefit of ADABOOST is that its stage-wise design allows it to control over fitting. In summary, the author came to the conclusion that ensemble methods are superior to individual algorithms. Of the three basic ensemble methods, ADABOOST performs better in the majority of situations, despite the fact that each has its own benefits and drawbacks.

Question 7:
This paper discusses five frequently employed statistical tests: concentrating on the situations where a certain test should be used, situations where a incorrect conclusion from the test, and how to select the best test each of the potential choices. The author briefly explored the topic of Type I and Type II error, as well as test power, in order to address these concerns. In order to demonstrate the results for various tests in various contexts, the author also presents the outcomes of a simulation-based study.

The McNemar's test, the proportion test for comparing two proportions, the paired t-test using resampled data, the K-fold cross-validation paired test, and the 5×2 CV paired test are the five tests that are addressed in the study.

For the McNemar test, which compares the frequency of row and column margins in matched pairs of samples. Two issues exist with this assessment.This test simply takes into account the practice data. Second, it doesn't account for data variability brought on by the selection of the training set or the intrinsic randomness of the learning process. This test is performed to find the optimal algorithm for single-execution and low variability.

The proportion test is used to compare two sections that are just measuring the differences in error rates between two algorithms. We assume that the error rates of the two methods are independent, which is the primary flaw in our test. In practice, it might not be the case. Additionally, it is unable to determine how the statistics vary.

In a resampled test, the data is divided into different pieces at random and the algorithms are run numerous times. It frequently contains high type I errors. However, it performs better when testing the hypothesis for a variety of real-world issues.

When using K (let's say 10) fold cross validation, the test works well with big training sets and reduces type II error. However, because the accuracy findings are not independent, it frequently experiences overlapping problems.

In the 5×2 CV paired test, the mean difference is estimated using the data from the first of five replications after running 2 fold cross validation five times.

Additionally, the variances are determined and reduced using data from all folds. With type-I mistake in mind, its performance is at its peak. The type-II error, however, has a wide range of error rates.

The authors' conclusion was that there isn't a test that works in every circumstance. Each test has unique benefits and restrictions depending on the context in which it is used. In order to acquire the greatest results from the application of the tests, the test selection must be done carefully.

Question 8:
a) We can find one real world scenario like finding the best route to go someplace. There will be different road that will cost different amount of money (including toll, fuel cost.
Now, we can also imagine this scenario as choosing each as an action and cost is based on choosing that route.
b) The algorithm for the original finite horizon value iteration is:

$$\begin{aligned} V^0(s) &= R(s) \\ V^{k+1}(s) &= R(S) + \max_{\alpha \in A} \sum_{s' \in S} T(s,a,s')V^k(s') \end{aligned}$$

In case of state action reward, the functions will be updated. Initially there will be no action. So, we have the following formula:

$$\begin{aligned} V^0(s) &= 0 \\ V^{k+1}(s) &= \max_{\alpha \in A} R(S,a) + \max_{\alpha \in A} \sum_{s' \in S} T(s,a,s')V^k(s') \end{aligned}$$

c) The new transformed state-action reward function is defined as $R'(s)$. We will also need to keep track of the states in our new MDP, say M', that were just executed. The state, action, transition function and reward function for M is defined as: S, A, T and $R(s,a)$. Now, we will define a new state space called S' of M' and it will contains all old states as well as the new states. So, we have $Q_{s,a}$ for each state-action pair of M. The new reward function $R'(s)$ and transition function T' of M' are:

$$\begin{aligned} T'(s,a,Q_{s,a}) &= 1, \text{ for all } s \text{ and } a \\ T'(Q_{s,a},a',s') &= T(s,a,s'), \text{ for all } s \in S,\, a \in A,\, a' \in A',\, s' \in S \\ R'(s) &= 0 \text{ for all } s \in S \\ R'(Q_{s,a}) &= R(s,a) \text{for all } s \in S,\, a \in A \end{aligned}$$

Here, when the agent is state s and take a action a, it's reward is zero. The agent keeps this transition $Q_{s,a}$ as memory. In $Q_{s,a}$, the agent will transit to a regular state

$s^{'}$ with transition probability as $T(s, a, s^{'})$. So, we get one action that is equivalent to two actions in $M^{'}$.

Question 9:
A standard MDP is described by a set of states S, a set of actions A, a transition function T, and a reward function R and $T(s, a, s^{'})$ gives the probability of transitioning to s_0 after taking a action from from state s and $R(s)$ is the reward for taking that action.

The state space $M^{'}$ is defined as $S^{'} = S^k$. For each state in $M^{'}$, we get a tuple of states in S. It indicates that each state in $S^{'}$ is of the form $(s, s_1, ..., s_{k-1})$ where each component is a state in S. The action of $M^{'}$ is similar to M. So we can write $A = A^{'}$. Each state $(s, s_1, ..., s_{k-1})$ of $M^{'}$ encodes the state s at the current time and previous $k - 1$ states too and all these state information is needed to determine the distribution over next states too. The reward function will be $R^{'}(s, s_1, ..., s_{k-1}) = R(s)$. The transition function can be written as:

$$T^{'}((s, s_1, ..., s_{k-1}), a, \bar{S}) = \begin{cases} Pr(s^{'}|a, s, , s_1, ..., s_{k-1} & \text{if } \bar{S} = (s, s_1, ..., s_{k-1}) \\ 0 & \text{otherwise} \end{cases}$$

Here, the transition function uses a k^{th} order model to relate the transition probability to a different state $s^{'}$. It won't move on to that particular stage if the history is incorrectly modified. Sequences of state M and M' that have a non-zero probability of being produced by some policy correlate one to one. As a result, we can observe that there is an analogous MDP M' for any k order MDP M. We can see $|S^{'} = S^k|$ shows that the number of state in $M^{'}$ is exponential in k.

Question 10:
We have a reward function $R(s, a)$ that depends on the action taken in a state or a reward function $R(s, a, s^{'})$ that also depends on the result state $s^{'}$ when we take action a in state s and then transition to $s^{'}$. From the bellman equation which depends on the state only is:

$$V^*(s) = R(S) + \beta \max_{\alpha \in A} \sum_{s^{'} \in S} T(s, a, s^{'}) V^*(s^{'})$$

We need to move the reward function inside the max over actions for the state-action combined reward function called $R(s, a)$:

$$V^*(s) = \max_{\alpha \in A}(R(s, a) + \beta \sum_{s^{'} \in S} T(s, a, s^{'}) V^*(s^{'}))$$

Now, if the reward function is also depended on the future state $s^{'}$, the reward is moved into expectation since it depend on the next state:

$$V^*(s) = \max_{\alpha \in A} \sum_{s^{'} \in S} T(s, a, s^{'})(R(s, a, s^{'}) + \beta V^*(s^{'}))$$

Question 11:
a) We are given simple MDP with two states $S = s_0, s_1$ and a single action defined as $A = a$. The reward functions are $R(s_0) = 0$ and $R(s_1) = 1$. The transition functions are $T(s_0, a, s-1) = 1$ and $T(s_1, a, s-1) = 1$.

Under the discount factor $\beta = 1$, if the policy is π, let $V_0 = V^{\pi}(s_0)$ and $V_1 = V^{\pi}(s_1)$. The linear equations for evaluating the policy are:

$$V_0 = R(s_0) + \beta V_1 = \beta V_1$$
$$V_1 = R(s_1) + \beta V_1 = 1 + \beta V_1$$

In case of $\beta = 1$, we have:

$$V_0 = V_1$$
$$V_1 = 1 + V_1$$

From above, we can say the system has no solution since the policy does not have a well defined function. **b)** For $\beta = 0.9$:

$$V_0 = 0.9V_1$$
$$V_1 = 1 + 0.9V_1$$

After solving the two equations from above, we get $V_0 = 9$ and $V_1 = 10$

Programming and Empirical Analysis:
Question 1:
Training and testing accuracy of Naive Bayes classifier with Laplace smoothing:
Training Accuracy : 0.9596273291925466
Testing Accuracy : 0.8217821782178217