Homework 3 | KNN & HMM

Course: CS7616
Due: Sat, Apr 2, 11:55pm

KNN

In the lectures we discussed k-Nearest Neighbor or based classification. In this assignment, we are going to use them. We discuss the methods briefly in the following, but refer to slides and textbook for more details on these technique.

Using the notations from the class slide, for K-NN classifier we have:

$$\hat{f}_{kNN}(x) = \underset{y}{\text{argmax}} \hat{p}_{kNN}(x|y)\hat{P}(y) = \underset{y}{\text{argmax}} \frac{k_y}{n_y \Delta_{k,x}} \frac{n_y}{N} = \underset{y}{\text{argmax}} k_y$$

Distance metric $d(X^i, X^j) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ between data points $X \in \mathbb{R}^n$ defines a measure of *similarity*. So distance between data-points belonging to same cluster should be lower than that between points belonging to separate clusters. Distance metric plays an important role for successful classification or density estimation. And its true for K-NN also. You are encouraged to try different distance metric for your K-NN classifier apart from the usual Euclidean distance metric $d(X^i, X^j) = \|X^i - X^j\| = \sqrt{(X^i - X^j)^T (X^i - X^j)}$. You are required to scale the different dimensions of the data dfferently. We discussed whitening the data in the previous assignment and people reported that it made little to no difference. That can be a starting point for doing the scaling, but you can imagine that for determining neighbors certain dimensions are more important and thus should be weighted more. Another method [1, 2] is to learn the distance metric from the data itself. Usually we choose a fixed type of distance metric like Mahalanobis distance $d(X^i, X^j) = \|X^i - X^j\|_M = \sqrt{(X^i - X^j)^T M (X^i - X^j)}$ and we try to learn the parameters of the positive semidefinite matrix M. Learning the Mahalanobis distance metric is equivalent to warping a rescaling of a data that replaces each point X with $M^{\frac{1}{2}} X$. You can try either scaling the data manually as suggested in previous assignment or use metric learning algorithms like [1, 2]. You can use code available online for [1].

We discussed in class the effect different k can have on the classification. Cross-validation can be used to decide between different values for k.

In this problem set you will also try to estimate the test-error from training samples. This you will do through cross-validation. Then you will compare these estimates to the actual test error. You can create a test set by randomly selecting part of the dataset and holding them out of training and cross-validation.

KNN Datasets

For this HW you would be required to do the following for each of the three datasets listed later in this HW. 1. Randomly select 20% of the dataset for test data (TtD). The rest will be considered the training data (TD). [10 points] 2. Randomly select a subset of $d\%$ from the TD where $d = \{20, 50, 80, 100\}$. For the first three cases (except the 100% case) you should generate five training sets (TS) for each of the $d\%$ data. [10 points] 3. For each of the TS, do F-fold cross-validation (CV) on each of the generated TS. You are expected to do 2-fold, 5-fold and Leave-one-out cross validation to decide the best K. Report the CV error for the best K. [10 points] 4. Now, test the TtD with the best K and its associated TD and compare the error rate to CV error. Report which F worked best. [10 points] 5. Report on the stability of the results with respect to the TS of the same size. [10 points] 6. Report on the results with respect to the size of the TS. [10 points]

This are just the source links for technical reference. Please go to the **T-square resources** page to download the pre-processed data ready for you to use in this assessment.

Dataset	Description	Data
Wine Data Set	[Description]	[Data]
MNIST	[Description]	[Data]
Office Dataset	[Description]	[Data]

HMM

In this part of the assignment you will build HMM from data. Recall that an HMM involves hidden state that changes over time, as well as observable evidence, henceforth called the output of the HMM. An HMM involves these probabilities: 1. for each state s, the probability of observing each output o at that state 2. from each state s, the probability of traversing to every other state s' in one time step

In the first part of this question, you will be required to compute estimates of these probabilities from data. We are providing you with training data consisting of one or more sequences of state-output pairs. During this training phase, we assume that the state variables are visible. Given these sequences, you need to estimate the probabilities that define the HMM.

In the second part, compute the most probable sequence of states (according to the HMM that you built from data) for a given sequence of outputs. This is essentially the problem of implementing the Viterbi algorithm as described in class. Your Viterbi code will be provided with just the output part of each of these sequences, and from this, must compute the most likely sequence of states to produce such an output sequence. The state part of these sequences is provided so that you can compare the estimated state sequences generated by your code to the actual state sequences that generated this data.

HMM Dataset

In this dataset, a robot is wandering through the following small world:

The robot can only occupy the colored squares. At each time step, the robot attempts to move up, down, left or right, where the choice of direction is made at random. If the robot attempts to move onto a black square, or to leave the confines of its world, its action has no effect and it does not move at all. The robot can only sense the color of the square it occupies. However, its sensors are only 90% accurate, meaning that 10% of the time, it perceives a random color rather than the true color of the currently occupied square. The robot begins each walk in a randomly chosen colored square.

In this problem, state refers to the location of the robot in the world in x:y coordinates, and output refers to a perceived color (r, g, b or y). Thus, a typical random walk looks like this:

3:3 r 3:3 r 3:4 y 2:4 b 3:4 y 3:3 r 2:3 b 1:3 g 2:3 b 2:4 r 3:4 y 4:4 y

Here, the robot begins in square 3:3 perceiving red, attempts to make an illegal move (to the right), so stays in 3:3, still perceiving red. On the next step, the robot moves up to 3:4 perceiving yellow, then left to 2:4 perceiving blue (erroneously), and so on.

By running your program on this problem, you will build an HMM model of this world. Then, given only sensor information (i.e., a sequence of colors), your program will re-construct an estimate of the actual path taken by the robot through its world.

The training data for this problem is in robot_train.data, a file containing 200 training sequences (random walks) each sequence consisting of 200 steps. The testing data for this problem is in robot_test.data, a file containing 200 test sequence each sequence consisting of 200 steps. In both the files sequences are separated by the dot '.' delimiter.

For this assignment you will be required to do the following for each of the robot dataset. 1. Given the training dataset, compute the observation probabilities (probability of observing a particular output 'o') and transition probabilities (probability of traversing from one state to another). Visualize the observation probabilites in the grid as shown in the above figure. Comment about how accurate it is. [15 points] 2. Implement the Viterbi algorithm and compute the most probable states for the given test sequences.

Compute the average error across all test sequences. Error for each test sequence is defined as the average misclassification rate between the estimated path and the actual path taken by the robot. For example if the actual robot location is (1:1) and estimated location is (1:2), error is 1 and if the estimated location is (1:1), error is 0. Average it over all locations in a given sequence. [15 points] 3. You should write up briefly (say, in 1-3 paragraphs) what you found. Your write-up should include any observations you may have made about how well HMMs work on this problems and why. [10 points]

These are just the source links for technical reference. Please go to the **T-square resources** page to download the pre-processed data ready for you to use in this assessment.

Dataset	Train	Test
Robot Data Set	[Train]	[Test]

Submit

Submit a your code and a PDF report with results in a zip file. You can use any programming language, but we will prefer `Python` or `Matlab`. The code should be easily runnable. Please include a README.md file that describes how the TAs are to run your code.

We expect the report to contain all of the following parts. The Report should contain the following:

- All of the requirements listed in the sections above.
- You must include your code as well to get credit. No code submitted means a zero in the assignment.
- If you are using a library from somewhere else, please mention it here as well. We hope you use Piazza for this so more people benefit.
- Again, do not simply use a library functions that already implement boosting or random forests for you, although you may build from an existing decision tree structure implementation.
- You are encouraged to discuss and help others with anything short of giving them your code. There are many references on-line, especially with MNIST. However, you MAY NOT use code from the Internet.

References

[1] Kilian Q. Weinb erger and Lawrence K. Saul *Distance Metric Learning for Large Margin Nearest Neighbor Classification.* JMLR, 2009.
[2] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan and Stuart Russell *Distance metric learning, with application to clustering with side-information.* NIPS, 2002.