Homework 2 MLE and Naive Bayes

Instructions

Answer the questions and upload your answers to courseville. Answers can be in Thai or English. Answers can be either typed or handwritten and scanned. the assignment is divided into several small tasks. Each task is weighted equally (marked with **T**). For this assignment, each task is awarded 0.55 points. There are also optional tasks (marked with **OT**) counts for 0.25 points each.

MLE

Consider the following very simple model for stock pricing. The price at the end of each day is the price of the previous day multiplied by a fixed, but unknown, rate of return, α, with some noise, w. For a two-day period, we can observe the following sequence

$y_2 = \alpha y_1 + w_1$

$y_1 = \alpha y_0 + w_0$

where the noises w_0, w_1 are iid with the distribution $N(0, \sigma^2)$, $y_0 \sim N(0, \lambda)$ is independent of the noise sequence. σ^2 and λ are known, while α is unknown.

T1. Find the MLE of the rate of return, α, given the observed price at the end of each day y_2, y_1, y_0. In other words, compute for the value of α that maximizes $p(y_2, y_1, y_0|\alpha)$

Hint: This is a Markov process, e.g. y_2 is independent of y_0 given y_1. In general, a process is Markov if $p(y_n|y_{n-1}, y_{n-2}, ...) = p(y_n|y_{n-1})$. In other words, the present is independent of the past $(y_{n-2}, y_{n-3}, ...)$, conditioned on the immediate past y_{n-1}.

OT1. Consider the general case, where

$y_{n+1} = \alpha y_n + w_n, n = 0, 1, 2, ...$

Find the MLE given the observed price $y_{N+1}, y_N, ..., y_0$

Simple Bayes Classifier

A student in Pattern Recognition course had finally built the ultimate classifier for cat emotions. He used one input features: the amount of food the cat ate that day, x (Being a good student he already normalized x to standard Normal). He proposed the following likelihood probabilities for class 1 (happy cat) and 2 (sad cat)

$P(x|w_1) = N(5, 2)$

$P(x|w_2) = N(0, 2)$

Figure 1: The sad cat and the happy cat used in training

T2. Plot the posteriors values of the two classes on the same axis. Using the likelihood ratio test, what is the decision boundary for this classifier? Assume equal prior probabilities.

T3. What happen to the decision boundary if the cat is happy with a prior of 0.8?

OT2. For the ordinary case of $P(x|w_1) = N(\mu_1, \sigma^2)$, $P(x|w_2) = N(\mu_2, \sigma^2)$, $p(w_1) = p(w_2) = 0.5$, prove that the decision boundary is at $x = \frac{\mu_1+\mu_2}{2}$
If the student changed his model to
$P(x|w_1) = N(5, 2)$
$P(x|w_2) = N(0, 4)$

- Plot the posteriors values of the two classes on the same axis. What is the decision boundary for this classifier? Assume equal prior probabilities.

Employee Attrition Prediction

Figure 2: Pictures of ~~employees~~ cats not used in this dataset.

In this part of the homework, we will work on employee attrition prediction using data from Kaggle IBM HR Analytics Employee Attrition & Performance [1].

[1] https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset/home

The data
For each employee, 34 features are provided. We will use these features to predict each employee attrition e.g whether the employee will leave the company (`yes` for leaving, `no` for staying)
Notable features are:

- Education: 1 'Below College', 2 'College', 3 'Bachelor', 4 'Master', 5 'Doctor'.
- Environment Satisfaction: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.
- Job Involvement: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.
- Job Satisfaction: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.
- Performance Rating: 1 'Low', 2 'Good', 3 'Excellent', 4 'Outstanding'.
- Relationship Satisfaction: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.
- WorkLifeBalance: 1 'Bad', 2 'Good', 3 'Better', 4 'Best'.

The database
First let's look at the given data file `hr-employee-attrition-with-null.csv`. Load the data using pandas. Use describe() and head() to get a sense of what the data is like. Our target of prediction is `Attrition`. Other columns are our input features.
Data cleaning
There are many missing values in this database. They are represented with NaN. In the previous homework, we filled the missing values with the mean, median, or mode values. That is because classifiers such as logistic regression cannot deal with missing feature values. However, for the case of Naive Bayes which we will use in this homework compares $\prod_i p(x_i|class)$ and treat each x_i as independent features. Thus, if a feature i is missing, we can drop that term from the comparison without having to guess what the missing feature is.
First, convert the yes and no in this data table to 1 and 0. Then, we have to convert each categorical feature to number.

```
all.loc[all["Attrition"] == "no", "Attrition"] = 0.0
all.loc[all["Attrition"] == "yes", "Attrition"] = 1.0
for col in cat_cols:
    all[col] = pd.Categorical(all[col]).codes
```

We will also drop the employee numbers.

```
all = all.drop(columns = "EmployeeNumber")
```

There is no standard rule on how much data you should segment into as training and test set. But for now let's use 90% training 10% testing. Select 10% of the is Attrition == yes and 10% of the is Attrition == no as your testing set, `test_set`. Then, use the rest of the data as your training set, `train_set`.

Histogram discretization
In class, we learned that in order to create a Bayes Classifier we first need to estimate the posterior or likelihood probability distributions. The simplest way to estimate probability distributions is via histograms. To do histogram estimation, we divide the entire data space into a finite number of bins. Then, we count how many data points are there in each bin and normalize using the total number of data points (so that the probability sums to 1). Since we are grouping a continuous valued feature into a finite number of bins, we can also call this process, discretization.
The following code create a histogram of a column `col` from `train_set`

```
# remove NaN values
train_col_no_nan = train_set[~np.isnan(train_set[col])]

# bin the data into 40 equally spaced bins
# hist is the count for each bin
# bin_edge is the edge values of the bins
hist, bin_edge = np.histogram(train_col_no_nan, 40)

# make sure to import matplotlib.pyplot as plt
# plot the histogram
plt.fill_between(bin_edge.repeat(2)[1:-1],hist.repeat(2),facecolor='steelblue')
plt.show()
```

T4. Observe the histograms. Can we use a Gaussian to estimate this histogram? Why? What about a Gaussian Mixture Model (GMM)?

T5. How many bins have zero counts? Do you think this is a good discretization? Why?
The above discretization equally segments the space into equally spaced bins. This is the best method to segment if you know nothing about the data. Still, doing so may leave us with many bins with zero counts when we have too little data. To prevent this issue, we might assume that the distribution of our data is normal then draw the probabilities of each data point from this distribution instead. We will do this in the next section. For now, do

1) We will first set the number of bins to 10 for `Age`, `MonthlyIncome` and `DistanceFromHome`. **Make numbers of bin a parameter as we will change this later.**
2) Bin each values in the training set into bins using the function `np.digitize`, then count the number in each bins using `np.bincount`.

T6. Plot the histogram according to our discretization scheme just like the process above (with 10, 40, and 100 bins, and show 3 plots for Age, MonthlyIncome, and DistanceFromHome). Which bin size is most sensible for each features? Why?

T7. For the rest of the features, which one should be discretized? What are the criteria for choosing whether we should discretize a feature or not? Answer this and discretize those features into 10 bins each. In other words, figure out the bin_edge for each feature, then use digitize() to convert the features to discrete values.

The MLE for the likelihood distribution of discretized histograms

We would like to build a Naive Bayes classifier which compares the posterior $p(leave|x_i)$ against $p(stay|x_i)$. However, figuring out $p(class|x_i)$ is often hard (not true for this case). Thus, we turn to the likelihood $p(x_i|class)$, which can be derived from the discretized histograms.

T8. What kind of distribution should we use to model histograms? (Answer a distribution name) What is the MLE for the likelihood distributions of each of the 33 features? (Describe how to estimate the distributions for each type of feature). Plot the likelihood distributions of MonthlyIncome, JobRole, HourlyRate, and MaritalStatus.

T9. What is the prior distribution of the two classes?

Naive Bayes classification

We are now ready to build our Naive Bayes classifier. Which makes a decision according to

$$H(x) = \frac{p(leave)}{p(stay)} \prod_{i=1} \frac{p(x_i|leave)}{p(x_i|stay)} \tag{1}$$

If $H(x)$ is larger than 1, then classify it as `leave`. If $H(x)$ is smaller than 1, then classify it as `stay`.

Note we often work in the log scale to prevent floating point underflow. In other words,

$$lH(x) = logp(leave) - logp(stay) +$$
$$\Sigma_{i=1}[logp(x_i|leave) - logpp(x_i|stay)]$$

If $lH(x)$ is larger than 0, then classify it as leave. If $lH(x)$ is smaller than 0, then classify it as stay.

T10. Use the learned distributions to classify the `test_set`. Don't forget to allow your classifier to handle missing values in the test set. Report the overall Accuracy. Then, report the Precision, Recall, and F score for detecting attrition. See Lecture 1 for the definitions of each metric.

Probability density function

Now, instead of using histogram discretization, we will assume that our features are normally distributed. By doing so, we can estimate the mean and standard deviation for each feature and compute the probability of each test feature by using the Gaussian probability density function instead. You can do this by calling

```
scipy.stats.norm(mean, std).pdf(feature_value)
```

T11. Use the learned distributions to classify the `test_set`. Report the results using the same metric as the previous question

Baseline comparison

In machine learning, we need to be able to evaluate how good our model is. We usually compare our model with a different model and show that our model is better. Sometimes we do not have a candidate model to evaluate our method against. In this homework, we will look at two simple baselines, the random choice, and the majority rule.

T12. The random choice baseline is the accuracy if you make a random guess for each test sample. Give random guess (50% leaving, and 50% staying) to the test samples. Report the overall Accuracy. Then, report the Precision, Recall, and F score for attrition prediction using the random choice baseline.

T13. The majority rule is the accuracy if you use the most frequent class from the training set as the classification decision. Report the overall Accuracy. Then, report the Precision, Recall, and F score for attrition prediction using the majority rule baseline.

T14. Compare the two baselines with your Naive Bayes classifier.

Threshold finding

In practice, instead of comparing $lH(x)$ against 0, we usually compare against a threshold, t. We can change the threshold so that we maximize the accuracy, precision, recall, or F score (depending on which measure we want to optimize).

T15. Use the following threshold values

```
t = np.arange(-5,5,0.05)
```

find the best accuracy, and F score (and the corresponding thresholds)

Receiver Operating Characteristic (RoC) curve

The recall rate (true positive rate) and the false alarm rate can change as we vary the threshold. The false alarm rate will deteriorate as we decrease the threshold (more false alarms). On the other hand, the recall rate will improve. This is also another trade-off machine learning practitioners need to consider. If we plot the false alarm vs recall as we vary the threshold (false alarm as the x-axis and recall as the y-axis), we get a plot called the "Receiver operating characteristic (RoC) curve." The RoC curve illustrates the performance of a binary classifier (Will this person leave? Will this person survive the Titanic? yes or no) as the threshold is varied. An example RoC curve is shown below

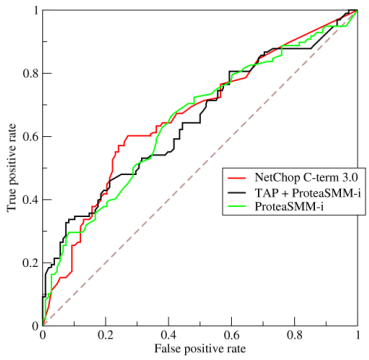

Figure 3: An example RoC curve. Source `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`

T16. Plot the RoC of your classifier.

T17. Change the number of discretization bins to 5. What happens to the RoC curve? Which discretization is better? The number of discretization bins can be considered as a hyperparameter, and must be chosen by comparing the final performance.

T18. Submit your predictions and code on mycourseville.

If you've made it this far, **congratulations!** you've just created simple models that can help HR deal with one of their biggest problems. Simple, isn't it? This is a real world task with real implications, and I personally have been approached by big companies to help with this.

(Optional) Classifier Variance

Recall, in class, we talked about the variance of a classifier as the training set changes. In this section, we will evaluate our model if we shuffle the training and test data. This will give a measure whether our recognizer is good just because we are lucky (and give statistical significance to our experiments).

OT3. Shuffle the database, and create new test and train sets. Redo the entire training and evaluation process 10 times (each time with a new training and test set). Calculate the mean and variance of the accuracy rate.