2/2566 FRA501: Pattern Recognition

Homework 2 MLE and Naive Bayes

Instructions

Answer the questions and upload your answers to google drive. Answers can be in Thai or English. Answers can be either typed or handwritten and scanned.

CHAPTER01

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, a, with some noise, w. For a two-day period, we can observe the following sequence:

$y_2 = ay_1 + w_1$

$y_1 = ay_0 + w_0$

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 a is unknown.

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

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 $p(y_n \mid y_{n-1}, y_{n-2}, \ldots) = p(y_n \mid y_{n-1})$.

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. He proposed the following likelihood probabilities for class 1 (happy cat) and 2 (sad cat)

$P(x \mid w_1) = N(5, 2)$

$P(x \mid w_2) = N(0, 2)$

2. 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. What happen with $x < 2$ (Happy cat or Sad Cat)

3. What happen to the decision boundary if the cat is happy with a prior of 0.8? Plot the posteriors values of the two classes on the same axis. What happen with x < 2 (Happy cat or Sad Cat)

CHAPTER02

In this part of the homework, we will work on employee attrition prediction using data from Kaggle IBM HR Analytics Employee Attrition & Performance (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 'Doc- tor'.
- 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% from the "Attrition == yes" and 10% from the "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()
```

1. Observe the histogram for Age, MonthlyIncome and DistanceFromHome. How many bins have zero counts? Do you think this is a good discretization? Why?

2. Can we use a Gaussian to estimate this histogram? Why? What about a Gaussian Mixture Model (GMM)?

Note: Bin each values in the training set into bins using the function np.digitize, then count the number in each bins using np.bincount. Be careful with the maximum and minimum values, your first bin should cover $-inf$, and your final bin should cover inf, so that you can handle test data that might be outside of the minimum and maximum values.

3. Now plot the histogram according to the method described above (with 10, 40, and 100 bins) and show 3 plots each for Age, MonthlyIncome, and DistanceFromHome. Which bin size is most sensible for each feature? Why?

4. For the rest of the features (Numeric and Category), which one should be discretized in order to be modeled by histograms? What are the criteria for choosing whether we should discretize a feature or not?

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.

5. Plot the likelihood distributions of the features from (4) for different Attrition values.

Naive Bayes classification

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

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

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.

6. If we use the current Naive Bayes with our current Maximum Likelihood Estimates, we will find that some $P(x_i|attrition)$ will be zero and will result in the entire product term to be zero. Propose a method to fix this problem.

7. Implement your Naive Bayes classifier. 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 F1 score for detecting attrition.

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)
```

8. Use the learned distributions to classify the test set. Report the results using the same metric as the previous question.