Homework 2

Week 2 - MLE and Naive Bayes

Patthadon Phengpinij
Collaborators. ChatGPT

1 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 $\mathcal{N}(0, \sigma^2)$, $y_0 \sim \mathcal{N}(0, \lambda)$ 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 $p(y_2, y_1, y_0 \mid \alpha)$.

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 $p(y_n \mid y_{n-1}, y_{n-2}, \ldots) = p(y_n \mid y_{n-1})$. In other words, the present is independent of the past $(y_{n-2}, y_{n-3}, \ldots)$, conditioned on the immediate past y_{n-1}. You may also find the steps of the proof for logistic regression we did in class useful.

Solution. First, we can express the joint probability $p(y_2, y_1, y_0 \mid \alpha)$ using the Markov property:

$$p(y_2, y_1, y_0 \mid \alpha) = p(y_2 \mid y_1, \alpha) \times p(y_1 \mid y_0, \alpha) \times p(y_0)$$

Next, we can write down the conditional probabilities based on the model:

1. The conditional probability $p(y_2 \mid y_1, \alpha)$ is given by the Gaussian distribution:

$$p(y_2 \mid y_1, \alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_2 - \alpha y_1)^2}{2\sigma^2}\right)$$

2. Likewise, the conditional probability $p(y_1 \mid y_0, \alpha)$ is:

$$p(y_1 \mid y_0, \alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_1 - \alpha y_0)^2}{2\sigma^2}\right)$$

3. The prior probability $p(y_0)$ is:

$$p(y_0) = \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-\frac{y_0^2}{2\lambda}\right)$$

Thus, the joint probability becomes:

$$\begin{aligned} p(y_2, y_1, y_0 \mid \alpha) &= p(y_2 \mid y_1, \alpha) \times p(y_1 \mid y_0, \alpha) \times p(y_0) \\ &= \frac{1}{(2\pi\sigma^2)\sqrt{2\pi\lambda}} \exp\left(-\frac{(y_2 - \alpha y_1)^2 + (y_1 - \alpha y_0)^2}{2\sigma^2} - \frac{y_0^2}{2\lambda}\right) \end{aligned}$$

We want to find the value of α that maximizes this joint probability. To do this, we can take the logarithm of the joint probability (log-likelihood):

$$\log\left(p(y_2, y_1, y_0 \mid \alpha)\right) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\log(2\pi\lambda) - \frac{(y_2 - \alpha y_1)^2 + (y_1 - \alpha y_0)^2}{2\sigma^2} - \frac{y_0^2}{2\lambda}$$

To find the MLE, we take the derivative of the log-likelihood with respect to α and set it to zero:

$$\begin{aligned}
\frac{d}{d\alpha}\left[\log\left(p(y_2, y_1, y_0 \mid \alpha)\right)\right] &= -0 - 0 - \frac{1}{2\sigma^2}\left[-2\alpha(y_2 - \alpha y_1) - 2\alpha(y_1 - \alpha y_0)\right] - 0 \\
0 &= \frac{1}{\sigma^2}\left[(y_2 - \alpha y_1)y_1 + (y_1 - \alpha y_0)y_0\right] \\
0 &= y_2 y_1 - \alpha y_1^2 + y_1 y_0 - \alpha y_0^2 \\
\alpha(y_1^2 + y_0^2) &= y_2 y_1 + y_1 y_0 \\
\alpha &= \frac{y_2 y_1 + y_1 y_0}{y_1^2 + y_0^2}
\end{aligned}$$

Therefore, the MLE for α is:

$$\boxed{\hat{\alpha} = \frac{y_2 y_1 + y_1 y_0}{y_1^2 + y_0^2}}$$

OT1. Consider the general case, where

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

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

Solution. Just like in the two-day case, we can express the joint probability using the Markov property:

$$p(y_{N+1}, y_N, \ldots, y_0 \mid \alpha) = p(y_{N+1} \mid y_N, \alpha) \times p(y_N \mid y_{N-1}, \alpha) \times \ldots \times p(y_1 \mid y_0, \alpha) \times p(y_0)$$

Next, we can write down the conditional probabilities based on the model:

1. The conditional probability $p(y_{n+1} \mid y_n, \alpha)$ is given by the Gaussian distribution:

$$p(y_{n+1} \mid y_n, \alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_{n+1} - \alpha y_n)^2}{2\sigma^2}\right)$$

2. The prior probability $p(y_0)$ is:

$$p(y_0) = \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-\frac{y_0^2}{2\lambda}\right)$$

Thus, the joint probability becomes:

$$\begin{aligned}
p(y_{N+1}, y_N, \ldots, y_0 \mid \alpha) &= \prod_{n=0}^{N} p(y_{n+1} \mid y_n, \alpha) \times p(y_0) \\
&= \frac{1}{(2\pi\sigma^2)^{(N+1)/2}\sqrt{2\pi\lambda}} \exp\left(-\sum_{n=0}^{N} \frac{(y_{n+1} - \alpha y_n)^2}{2\sigma^2} - \frac{y_0^2}{2\lambda}\right)
\end{aligned}$$

We want to find the value of α that maximizes this joint probability. To do this, we can take the logarithm of the joint probability (log-likelihood):

$$\log\left(p(y_{N+1}, y_N, \ldots, y_0 \mid \alpha)\right) = -\frac{N+1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\log(2\pi\lambda) - \sum_{n=0}^{N} \frac{(y_{n+1} - \alpha y_n)^2}{2\sigma^2} - \frac{y_0^2}{2\lambda}$$

To find the MLE, we take the derivative of the log-likelihood with respect to α and set it to zero:

$$\begin{aligned}
\frac{d}{d\alpha}\left[\log\left(p(y_{N+1}, y_N, \ldots, y_0 \mid \alpha)\right)\right] &= -\sum_{n=0}^{N} \frac{1}{2\sigma^2}\left[-2(y_{n+1} - \alpha y_n)y_n\right] \\
0 &= \sum_{n=0}^{N} \frac{1}{\sigma^2}(y_{n+1} - \alpha y_n)y_n \\
0 &= \sum_{n=0}^{N} y_{n+1}y_n - \alpha \sum_{n=0}^{N} y_n^2 \\
\alpha \sum_{n=0}^{N} y_n^2 &= \sum_{n=0}^{N} y_{n+1}y_n \\
\alpha &= \frac{\sum_{n=0}^{N} y_{n+1}y_n}{\sum_{n=0}^{N} y_n^2}
\end{aligned}$$

Therefore, the MLE for α in the general case is:

$$\boxed{\hat{\alpha} = \frac{\sum_{n=0}^{N} y_{n+1}y_n}{\sum_{n=0}^{N} y_n^2}}$$

2 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)

$$P(x \mid w_1) = \mathcal{N}(4, 2)$$

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

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.

Solution. The solution goes here.

T3. What happen to the decision boundary if the cat is happy with a prior of 0.75?

Solution. The solution goes here.

OT2. For the ordinary case of $P(x \mid w_1) = \mathcal{N}(\mu_1, \sigma^2)$, $P(x \mid w_2) = \mathcal{N}(\mu_2, \sigma^2)$, $p(w_1) = p(w_2) = 0.5$, prove that the decision boundary is at $x = \frac{\mu_1 + \mu_2}{2}$.

Solution. The solution for the **OT** problem goes here.

OT3. If the student changed his model to

$$P(x \mid w_1) = \mathcal{N}(4, 2)$$

$$P(x \mid w_2) = \mathcal{N}(0, 4)$$

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

Solution. The solution for the **OT** problem goes here.

3 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.

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 $\prod_i p(x_i \mid class)$ 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`.

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

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

Solution. The solution goes here.

T5. Can we use a Gaussian to estimate this histogram? Why? What about a Gaussian Mixture Model (GMM)?

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 dat is Normal then draw the probabilities of each data point from this distribution instead. We will do this later. For now, do

1. 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`. 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.

You do not need to submit anything for this task. You might want to make this a function so that you can change the number of bins.

Solution. The solution goes here.

T6. 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 features? Why?

Solution. The solution goes here.

T7. For the rest of the features, 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? 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.

Solution. The solution goes here.

The MLE for the likelihood distribution of discretized histograms

We would like to build a Naive Bayes classifier which compares the posterior $p(leave \mid x_i)$ against $p(stay \mid x_i)$. However, figuring out $p(class \mid x_i)$ is often hard (not true for this case). Thus, we turn to the likelihood $p(x_i \mid class)$, 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 this likelihood distribution? (Describe how to do the MLE). Plot the likelihood distributions of `MonthlyIncome`, `JobRole`, `HourlyRate`, and `MaritalStatus` for different Attrition values.

Hint: In class we talk about how a fair coin can be modeled using the Bernoulli distribution. A histogram is very similar to a dice in the sense that the outcome is a set of possibilities.

Solution. The solution goes here.

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

Solution. The solution goes here.

Naive Bayes classification

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

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

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,

$$lH(x) = \log\left(p(leave)\right) - \log\left(p(stay)\right) + \sum_{i=1} \left[\log\left(p(x_i \mid leave)\right) - \log\left(p(x_i \mid stay)\right)\right]$$

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. 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.

Solution. The solution goes here.

T11. 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 F score for detecting attrition. See Lecture 1 for the definitions of each metric.

Solution. The solution goes here.

Probability density function

Now, instead of using histogram discretization, we will assume that our features are normally distributed. In other words, for certain feature types, $P(x_i|attrition)$ is now 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)
```

T12. Use the learned distributions to classify the `test_set`. Report the results using the same

metric as the previous question.

Solution. The solution goes here.

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.

T13. 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.

Solution. The solution goes here.

T14. 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.

Solution. The solution goes here.

T15. Compare the two baselines with your Naive Bayes classifier.

Solution. The solution goes here.

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).

T16. Use the following threshold values

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

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

Solution. The solution goes here.

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

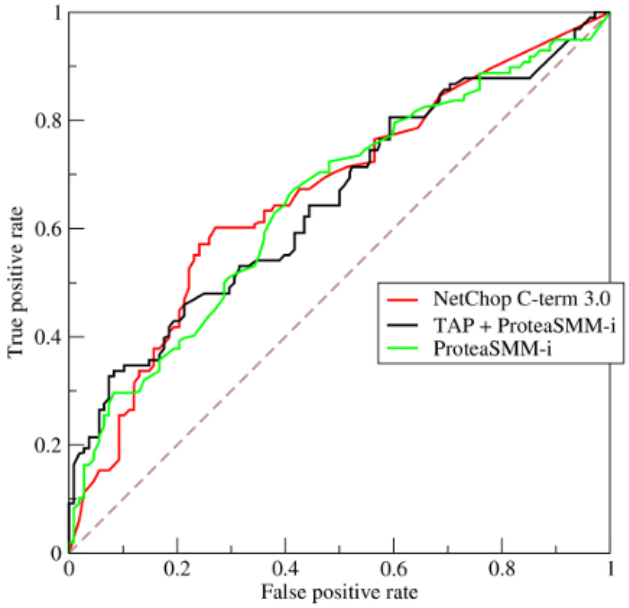

Figure 3: An example RoC curve. Source wikipedia

T17. Plot the RoC of your classifier.

> **Solution.** The solution goes here.

T18. 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.

> **Solution.** The solution goes here.

T19. Submit your code (.py or .ipynb) 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.

> **Solution.** The solution goes here.

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).

OT4. 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.

Solution. The solution for the **OT** problem goes here.