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.

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.

- 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}.

- (Optional) 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

- 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 to the decision boundary if the cat is happy with a prior of 0.8?
- (Optional) 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.

Housekeeping Genes Prediction

In this part of the homework we will work on housekeeping genes classification. If you do not want to read through the biology terms, skip to **The database** section.

What are housekeeping genes?

Cells in our body all share basic functions and activities, such as production of proteins and cell growth, that are maintained by a set of genes called "**housekeeping genes**." As such, housekeeping genes are typically expressed at consistent levels in every cell and under every condition. In contrast, "**tissue-specific genes**" are those responsible for highly specialized cellular functions and each of them is expressed in only some tissues in an organism. Because housekeeping genes are tightly linked to basic cellular activities, they often served as potential drug targets and as evolutionary markers for distinguishing closely related species.

Classification of housekeeping genes

The most straightforward, but not the cheapest, way to identify housekeeping genes in an organism is to sample cells from each of its tissues/organs, quantify the expression level of each gene in each sample, and search for genes that are consistently expressed in all samples. Even without taking technical issues in measuring gene expression into consideration, this approach already requires considerable amount of budgets and efforts. For example, the cost of doing gene sequencing on one sample is around 110,000 baht. If we want to find housekeeping genes, we might want to sequence at least 10 samples from different organs which can cost millions. Is there a better way? Can we predict housekeeping genes using easy-to-obtain features instead?

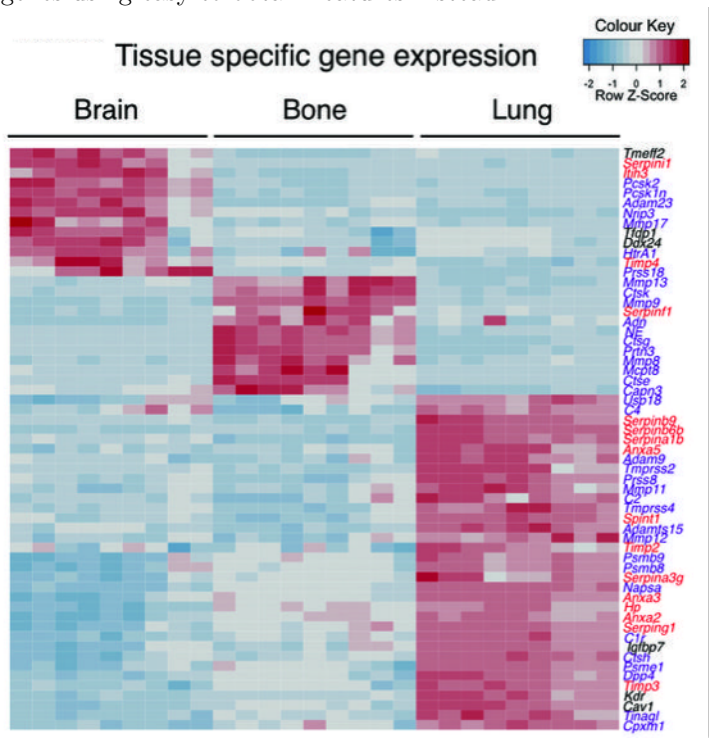

Figure 2: Example of tissue-specific gene identification via gene expression (Sevenich et al. Nature Cell Biology 16, 876-888, 2014). On the right side lists different gene types. Red cells correspond to higher confidence that a gene is from a particular organ. This figure only has tissue specific genes. Housekeeping genes would be expressed in all tissues.

Genomic features for predicting housekeeping genes

Compared to gene expression levels which differ from cell to cell, the genome sequences in every cell of an individual are identical. Furthermore, the cost of genome sequencing continued to decrease over the years and has become affordable to most laboratories. Several studies have indicated that many genomic features, such as the length of a gene and the presence of certain sequence patterns near a gene, may be associated with housekeeping and tissue-specific genes. For example, the Scaffold/Matrix Attachment Regions (S/MAR) elements are frequently present near tissue-specific genes while sequence patterns such as Poly(dA-dT) and (CCGNN)n are frequently present near housekeeping genes.

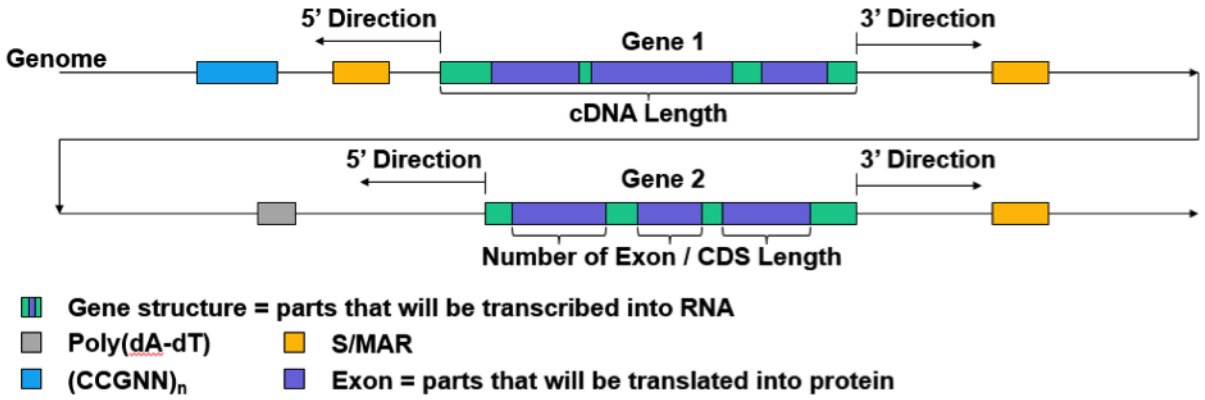

Figure 3: An example of gene structures and nearby sequence patterns on a genome.

Other features for predicting housekeeping genes – gene functions

Housekeeping genes and tissue-specific genes are responsible for different cellular functions. Gene ontology (GO) terms, the keywords which represent our biological knowledge of a gene, that are annotated to these two groups of genes also differ. We also would like to incorporate this knowledge as additional features to our model.

The data

For each gene, 9 features are provided:

- cDNA length [cDNA_length]: This is the length of RNA sequence that would be transcribed from the gene.
- Coding sequence (CDS) length [cds_length]: This is the length of the sequence portion that would be translated into proteins.

- Number of exons [exon_nr]: This is the number of separated CDS blocks located in the cDNA. It is related to the cds_length.
- Presence of S/MAR in the 5' region [5_MAR_presence]: This is the yes/no indicator of whether an S/MAR element is present somewhere in front of the gene on the genome.
- Presence of S/MAR in the 3' region [3_MAR_presence]: This is the yes/no indicator of whether an S/MAR element is present somewhere behind the gene on the genome.
- Presence of Poly(dA-dT) in the 5' region [5_polyA_18_presence]: This is the yes/no indicator of whether a Poly(dA-dT) element is present in front of the gene on the genome.
- Presence of (CCGNN)2-5 in the 5' region [5_CCGNN_2_5_presence]: This is the yes/no indicator of whether a (CCGNN)2-5 element is present in front of the gene on the genome.
- Percentage of gene ontology (GO) terms that match to "housekeeping" GO terms [perc_go_hk_match]: This is the % of matching between GO terms annotated to the gene and GO terms annotated to known housekeeping genes.
- Percentage of gene ontology (GO) terms that match to "tissue-specific" GO terms [perc_go_ts_match]: This is the % of matching between GO terms annotated to the gene and GO terms annotated to known tissue-specific genes.

We have data for three species: human, mouse, and fruit fly. Here are some data statistics. However, we will only work on human data for this homework.

Species	Total Genes	# of HK	# of TS
Human	47229	103	667
Mouse	22356	87	335
Fruit fly	20016	80	412

Table 1: Number of total genes, known housekeeping genes (HK), and known tissue-specific genes (TS).

The database

First let's look at the given data file `12864_2006_660_MOESM1_ESM.csv`.
Load the data using pandas. Use describe() and head() to get a sense of what the data is like. `EMBL_transcript_id` is the name of each genes. `cDNA_length`, `cds_length`, `exons_nr`, `5_MAR_presence`, `3_MAR_presence`, `5_polyA_18_presence`, `5_CCGNN_2_5_presence`, `perc_go_hk_match`, `perc_go_ts_match` are our input features. Our target of prediction is `is_hk`.

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. We can also drop the name of the genes and just refer to them by their index values.

```
all.loc[all["5_MAR_presence"] == "no", "5_MAR_presence"] = 0.0
all.loc[all["5_MAR_presence"] == "yes", "5_MAR_presence"] = 1.0
all.loc[all["3_MAR_presence"] == "no", "3_MAR_presence"] = 0.0
all.loc[all["3_MAR_presence"] == "yes", "3_MAR_presence"] = 1.0
all.loc[all["5_polyA_18_presence"] == "no", "5_polyA_18_presence"] = 0.0
all.loc[all["5_polyA_18_presence"] == "yes", "5_polyA_18_presence"] = 1.0
all.loc[all["5_CCGNN_2_5_presence"] == "no", "5_CCGNN_2_5_presence"] = 0.0
all.loc[all["5_CCGNN_2_5_presence"] == "yes", "5_CCGNN_2_5_presence"] = 1.0
all.loc[all["is_hk"] == "no", "is_hk"] = 0.0
all.loc[all["is_hk"] == "yes", "is_hk"] = 1.0
del all["EMBL_transcript_id"]
```

Unsupervised data
Let's look at the `is_hk` column, our target prediction.

- How many items are NaN in the `is_hk` column? How many items are known housekeeping genes? How many items are known tissue specific genes?

So far we have only looked at the condition where our target labels are known. For example, if I want to make a cat vs. dog classifier, I show a bunch of cat pictures and a bunch of dog pictures to the classifier. This is known as **supervised learning**, where I have to provide the class or target labels. What if I only have pictures, but I do not know whether it's a cat or a dog or a mouse? This kind of learning process is known as **unsupervised learning**.
Since most of our data has unknown `is_hk` labels, we would like to make use of this data somehow. We will do so via a discretization process. But before we get to that point, let's create a training and test set.
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_hk == yes` and 10% of the `is_hk == no` as your testing set, `test_set`. Then, use the rest of the data as your training set, `train_set`.
From `train_set`, filter the ones with `is_hk == NaN` as the unsupervised training set, `unsup_train_set`. Filter the ones with `is_hk == yes or no` as the supervised training set, `sup_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 the `cDNA_length` from `train_set`

```
# remove NaN values
train_set_clength_no_nan = train_set_clength[~np.isnan(train_set_clength)]
# bin the data into 1000 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_set_clength_no_nan,1000)
# 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()
# plot the first 100 bins only
plt.fill_between(bin_edge.repeat(2)[1:100],hist.repeat(2)[1:100],facecolor='steelblue')
plt.show()
# plot the first 500 bins only
plt.fill_between(bin_edge.repeat(2)[1:500],hist.repeat(2)[1:500],facecolor='steelblue')
plt.show()
```

- Observe the histograms. Can we use a Gaussian to estimate this histogram? Why? What about a Gaussian Mixture Model (GMM)?
- 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. However, if you know about the specific characteristics of the data, you can try to put more bins around the region where the important information resides. One way to accomplish this is to segment according to the density of the data points. The regions with many data points, you segment more finely. To do so

1) Sort the data points into a ranked list, `train_set_clength_no_nan_sorted`.
2) Define the bin_edge at equally spaced ranks. You can do so by `train_set_clength_no_nan_sorted[0::spacing]`
3) If we do it this way, each values in bin_edge is not necessary unique. We can remove duplicate values by `bin_edge = np.unique(bin_edge)`
4) We can then bin each values in the training set into bins using the function `np.digitize`, then count the number in each bins using `np.bincount`.

- Plot the histogram according to our new discretization scheme just like the process above (with ∼1000 bins, and show 3 plots). Does it come out like how it should be?
- Discretize the values of `cDNA_length` and `cds_length` according to the `train_set`. 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 $p(housekeeping|x_i)$ against $p(nothousekeeping|x_i)$. 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.

- What is the MLE for the likelihood distributions of each of the 9 features? Plot the likelihood distributions. You should learn the discretization using `train_set`, but estimate the MLE using the `sup_train_set`.
- 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

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

If $H(x)$ is larger than 1, then classify it as housekeeping. If $H(x)$ is smaller than 1, then classify it as not housekeeping.
Note we often work in the log scale to prevent floating point underflow. In other words,

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

If $lH(x)$ is larger than 0, then classify it as housekeeping. If $lH(x)$ is smaller than 0, then classify it as not housekeeping.

- 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 housekeeping gene. See Lecture 1 for the definitions of each metric.

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.

- The random choice baseline is the accuracy if you make a random guess for each test sample. Give random guess (50% being housekeeping, and 50% being not housekeeping) to the test samples. Report the overall Accuracy. Then, report the Precision, Recall, and F score for detecting housekeeping gene using the random choice baseline.
- 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 detecting housekeeping gene using the majority rule baseline.
- 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).

- 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 (Is this gene a housekeeping gene? Will this person survive the Titanic? yes or no) as the threshold is varied. An example RoC curve is shown below

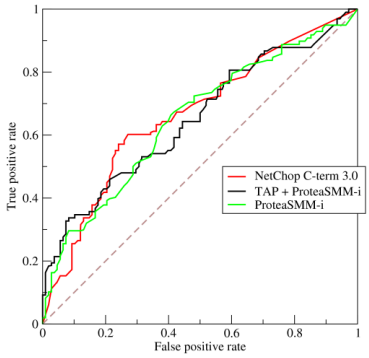

Figure 4: An example RoC curve. Source `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`

- Plot the RoC of your classifier.
- Change the number of discretization bins from ∼1000 to ∼500. 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.

Solving real world problems, one homework at a time
Apply your best model on `unsup_train_set` to make class predictions.

- Submit your predictions and code on mycourseville.

If you've made it this far, **congratulations!** You've just contributed to the advancement of modern medicine! Simple, isn't it?
(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).

- (Optional) 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.