10-715 Advanced Introduction to Machine Learning: Homework 2
MLE, MAP and Naive Bayes

Released: Wednesday, September 12, 2018
Due: 11:59 p.m. Wednesday, September 19, 2018

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that.
- **Collaboration policy:** Collaboration on solving the homework is allowed. Discussions are encouraged but you should think about the problems on your own. When you do collaborate, you should list your collaborators! Also cite your resources, in case you got some inspiration from other resources (books, websites, papers). If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Online submission:** You must submit your solutions online on Autolab (link: `https://autolab.andrew.cmu.edu/courses/10715-f18/assessments`). Please use LaTeX to typeset your solutions and save the output into a single pdf called **hw2.pdf** and submit to Homework2-Report.

Problem 1: MLE and MAP [40 Points]

This problem explores maximum likelihood estimation (MLE), which is a technique for estimating an unknown parameter of a probability distribution based on observed samples. Suppose we observe the values of n iid[1] random variables $X_1, \ldots, X_n$ drawn from a single Bernoulli distribution with parameter θ. In other words, for each X_i, we know that

$$P(X_i = 1) = \theta \quad \text{and} \quad P(X_i = 0) = 1 - \theta.$$

Our goal is to estimate the value of θ from these observed values of X_1 through X_n.

For any hypothetical value $\hat{\theta}$, we can compute the probability of observing the outcome $X_1, \ldots, X_n$ if the true parameter value θ were equal to $\hat{\theta}$. This probability of the observed data is often called the *data likelihood*, and the function $L(\hat{\theta}) = P(X_1, \ldots, X_n|\hat{\theta})$ that maps each $\hat{\theta}$ to the corresponding likelihood is called the *likelihood function*. A natural way to estimate the unknown parameter θ is to choose the $\hat{\theta}$ that maximizes the likelihood function. Formally,

$$\hat{\theta}^{\text{MLE}} = \underset{\hat{\theta}}{\operatorname{argmax}} L(\hat{\theta}).$$

Often it is more convenient to work with the log likelihood function $\ell(\hat{\theta}) = \log L(\hat{\theta})$. Since the log function is increasing, we also have

$$\hat{\theta}^{\text{MLE}} = \underset{\hat{\theta}}{\operatorname{argmax}} \ell(\hat{\theta}).$$

1. [**4 Points**] Write a formula for the log likelihood function, $\ell(\hat{\theta})$. Your function should depend on the random variables $X_1, \ldots, X_n$, the hypothetical parameter $\hat{\theta}$, and should be simplified as far as possible (i.e., don't just write the definition of the log likelihood function). Does the log likelihood function depend on the order of the random variables?

2. [**10 Points**] Compute a closed form expression for the maximum likelihood estimate (hint:recall that x is a critical point of $f(x)$ if $f'(x) = 0$. If you use part (1), remember to argue that finding the maximizer of $\log f(x)$ yields the maximizer of $f(x)$)

   Use your formula to compute θ^{MLE} for the following sequence of 10 samples:

   $$X = (0, 1, 1, 1, 1, 0, 1, 0, 1, 1).$$

3. [**10 Points**] Now we will consider a related distribution. Suppose we observe the values of m iid random variables $Y_1, \ldots, Y_m$ drawn from a single Binomial distribution $B(n, \theta)$. A Binomial distribution models the number of 1's from a sequence of n independent Bernoulli variables with parameter θ. In other words,

   $$P(Y_i = k) = \binom{n}{k}\theta^k(1-\theta)^{n-k} = \frac{n!}{k!(n-k)!} \cdot \theta^k(1-\theta)^{n-k}.$$

   Write a formula for the log likelihood function, $\ell(\hat{\theta})$. Your function should depend on the random variables $Y_1, \ldots, Y_m$ and the hypothetical parameter $\hat{\theta}$.

4. [**10 Points**] Compute a closed form expression for the maximum likelihood estimate. Then use your formula to compute $\hat{\theta}^{\text{MLE}}$ for the two Binomial random variables Y_1 and Y_2 which both have parameters $n = 5$ and θ. The Bernoulli variables for Y_1 and Y_2 resulted in $(0, 1, 1, 1, 1)$ and $(0, 1, 0, 1, 1)$, respectively, so $Y_1 = 4$ and $Y_2 = 3$.

5. [**6 Points**] How do your answers for parts 2 and 4 compare? If you got the same or different answers, why was that the case?

[1] iid means Independent, Identically Distributed.

Problem 2: Implementing Naive Bayes [60 Points]

In this question you will implement a Naive Bayes classifier for a text classification problem. You will be given a collection of text articles, each coming from either the serious European magazine *The Economist*, or from the not-so-serious American magazine *The Onion*. The goal is to learn a classifier that can distinguish between articles from each magazine.

We have pre-processed the articles so that they are easier to use in your experiments. We extracted the set of all words that occur in any of the articles. This set is called the *vocabulary* and we let V be the number of words in the vocabulary. For each article, we produced a feature vector $X = \langle X_0, \ldots, X_{V-1} \rangle$, where X_i is equal to 1 if the i^{th} word appears in the article and 0 otherwise. Each article is also accompanied by a class label of either 0 for The Economist or 1 for The Onion.

When we apply the Naive Bayes classification algorithm, we make two assumptions about the data: first, we assume that our data is drawn iid from a joint probability distribution over the possible feature vectors X and the corresponding class labels Y; second, we assume for each pair of features X_i and X_j with $i \neq j$ that X_i is conditionally independent of X_j given the class label Y (this is the Naive Bayes assumption). Under these assumptions, a natural classification rule is as follows: Given a new input X, predict the most probable class label $\hat{Y}$ given X. Formally,

$$\hat{Y} = \operatorname*{argmax}_y P(Y = y|X).$$

Using Bayes Rule and the Naive Bayes assumption, we can rewrite this classification rule as follows:

$$\begin{aligned}
\hat{Y} &= \operatorname*{argmax}_y \frac{P(X|Y=y)P(Y=y)}{P(X)} && \text{(Bayes Rule)} \\
&= \operatorname*{argmax}_y P(X|Y=y)P(Y=y) && \text{(Denominator does not depend on } y) \\
&= \operatorname*{argmax}_y P(X_1, \ldots, X_V|Y=y)P(Y=y) \\
&= \operatorname*{argmax}_y \left(\prod_{w=1}^{V} P(X_w|Y=y) \right) P(Y=y) && \text{(Conditional independence).}
\end{aligned}$$

The advantage of the Naive Bayes assumption is that it allows us to represent the distribution $P(X|Y = y)$ using many fewer parameters than would otherwise be possible. Specifically, since all the random variables are binary, we only need one parameter to represent the distribution of X_w given Y for each $w \in \{1, \ldots, V\}$ and $y \in \{1, 2\}$. This gives a total of $2V$ parameters. On the other hand, without the Naive Bayes assumption, it is not possible to factor the probability as above, and therefore we need one parameter for all but one of the 2^V possible feature vectors X and each class label $y \in \{1, 2\}$. This gives a total of $2(2^V - 1)$ parameters. The vocabulary for our data has $V \approx 26,000$ words. Under the Naive Bayes assumption, we require on the order of $52,000$ parameters, while without it we need more than 10^{7000}!

Of course, since we don't know the true joint distribution over feature vectors X and class labels Y, we need to estimate the probabilities $P(X|Y = y)$ and $P(Y = y)$ from the training data. For each word index $w \in \{1, \ldots, V\}$ and class label $y \in \{1, 2\}$, the distribution of X_w given $Y = y$ is a Bernoulli distribution with parameter θ_{yw}. In other words, there is some unknown number θ_{yw} such that

$$P(X_w = 1|Y = y) = \theta_{yw} \quad \text{and} \quad P(X_w = 0|Y = y) = 1 - \theta_{yw}.$$

We believe that there is a non-zero (but maybe very small) probability that any word in the vocabulary can appear in an article from either The Onion or The Economist. To make sure that our estimated probabilities are always non-zero, we will impose a Beta(2,1) prior on θ_{yw} and compute the MAP estimate from the training data.

Similarly, the distribution of Y (when we consider it alone) is a Bernoulli distribution with parameter ρ. In other words, there is some unknown number ρ such that

$$P(Y = 1) = \rho \quad \text{and} \quad P(Y = 0) = 1 - \rho.$$

In this case, since we have many examples of articles from both The Economist and The Onion, there is no risk of having zero-probability estimates, so we will instead use the MLE.

Programming Instructions

Parts (a) through (d) of this question each ask you to implement one function related to the Naive Bayes classifier. You will submit your code online to Homework2-Code on Autolab, which will execute it remotely against a suite of tests. Your grade will be automatically determined from the testing results. Since you get immediate feedback after submitting your code and you are allowed to submit as many different versions as you like (without any penalty), it is easy for you to check your code as you go.

To get started, log into the autolab website (`https://autolab.andrew.cmu.edu/courses/10715-f18/assessments/homework2`). Download the code template for Problem 2 and extract the contents (i.e., by executing `tar xvf prob2.tar` at the command line). In the archive you will find a data folder, containing the data required for completing this assignment, and a file naive_bayes.py which is where you would write your code. For submitting, create a new tar archive of the top-level directory (i.e., by executing `tar cvf prob2.tar prob2`) and upload that through the Autolab website. The scores will show up after 30 seconds. We use Python 2.7.5 for autograding. Note that only the scores of your last submission will be kept.

To make sure that the autograding is run correctly, please do not change the function names or the directory name. Only problem (a) to problem (d) will be autograded. After submitting your solution, you can click on the scores to see the grading log.

The folder `Data` contains the following elements

- `Vocabulary.txt`: A text file containing the words occurring in the news articles. Each line in this file is a word (note that some of the words may look a little strange because we have run them through a stemming algorithm that tries to make words with common roots look the same. For example, "stemming" and "stemmed" would both become "stem".) The line number (zero indexed; i.e the first word maps to 0) indicates the id of that word. The file contains 26048 words.

- `XTrain`: is a $n \times V$ dimensional matrix describing the n documents used for training your Naive Bayes classifier. The entry `XTrain(i,j)` is 1 if word j appears in the i^{th} training document and 0 otherwise.

- `yTrain` is a $n \times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 0 if the i^{th} document belongs to The Economist and 1 if it belongs to The Onion.

- `XTest` and `yTest` are the same as `XTrain` and `yTrain`, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.

- Finally, `XTrainSmall` and `yTrainSmall` are subsets of `XTrain` and `yTrain` which are used in the final question.

`XTrain` and `XTest` are stored in a sparse array format. Use the `loadmat` function provided to load the feature Matrices (X*). Load the label matrices (y*) using `np.load`.

Logspace Arithmetic

When working with very large or very small numbers (such as probabilities), it is useful to work in *logspace* to avoid numerical precision issues. In logspace, we keep track of the logs of numbers, instead of the numbers themselves. For example, if $p(x)$ and $p(y)$ are probability values, instead of storing $p(x)$ and $p(y)$ and computing $p(x) * p(y)$, we work in log space by storing $\log(p(x))$, $\log(p(y))$, and we can compute the log of the product, $\log(p(x) * p(y))$ by taking the sum: $\log(p(x) * p(y)) = log(p(x)) + log(p(y))$.

Note that while working in logspace, you may run into errors with wrt $\log[1]$ and $\log[0]$. Use a tolerance of $1e-5$ for those (using np.clamp might be a good idea)

Training Naive Bayes [34 Points]

(a) **[8 Points]** Complete the function `NB_XGivenY(XTrain, yTrain)`. The output `D` is a $2 \times V$ matrix, where for any word index $w \in \{0, \ldots, V-1\}$ and class index $y \in \{0, 1\}$, the entry `D(y,w)` is the MAP estimate of $\theta_{y,w} = \log[P(X_w = 1|Y = y)]$ with a Beta(2,1) prior distribution.

(b) **[8 Points]** Complete the function `NB_YPrior(yTrain)`. The output `p` is the MLE for $\log[\rho] = \log[P(y = 1)]$. (A straight forward way of doing that would be computing MLE of $P(Y = 1) = \rho_{MLE}$ and returning $\log[\rho_{MLE}]$).

(c) **[16 Points]** Complete the function `NB_Classify(X, D, logp)`. The input `X` is an $n \times V$ matrix containing n data points. The output `yHat` is a $n \times 1$ vector of predicted class labels, where `yHat(i)` is the predicted label for the i^{th} data point.

(d) **[2 Points]** Complete the function `ClassificationError(yHat, yTruth)`, which takes two vectors of equal length and returns the proportion of entries [0 - 1 range] that they disagree on.

Evaluating Naive Bayes [26 Points]

(e) **[8 Points]** Train your classifier on the data contained in `XTrain` and `yTrain` by running

```
D = NB_XGivenY(XTrain, yTrain);
p = NB_YPrior(yTrain);
```

Use the learned classifier to predict the labels for the article feature vectors in `XTrain` and `XTest` by running

```
yHatTrain = NB_Classify(XTrain, D, logp);
yHatTest = NB_Classify(XTest, D, logp);
```

Use the function `ClassificationError` to measure and report the training and testing error by running

```
trainError = ClassificationError(yHatTrain, yTrain);
testError = ClassificationError(yHatTest, yTest);
```

How do the train and test errors compare? Explain any significant differences.

(f) **[8 Points]** Repeat the steps from part (e), but this time use the smaller training set `XTrainSmall` and `yTrainSmall`. Explain any difference between the train and test error in this question and in part (g). [Hint: When we have less training data, does the prior have more or less impact on our classifier?]

(g) **[10 Points]** Finally, we will try to interpret the learned parameters. Train your classifier on the data contained in `XTrain` and `yTrain`. For each class label $y \in \{0, 1\}$, list the five words that the model says are most likely to occur in a document from class y. Also for each class label $y \in \{0, 1\}$, list the five words w that maximize the following quantity:

$$\frac{P(X_w = 1|Y = y)}{P(X_w = 1|Y \neq y)}.$$

Which list of words describes the two classes better? Briefly explain your reasoning.