# CS512 Machine Learning, Fall 2019: Homework 1

Due: Oct 30 22:00 pm

**Instructions**

- Submit an online copy of your report and code through SUCourse.
- Please submit the report and the code separately. Name your submission as CS512-YourName.pdf and CS512-YourName-Code.zip (or .tar), where you substitute in your first and last names into the filenames in place of 'YourName'.
- You may code in any programming language you would prefer. If the question is asking for you to implement, you should not use any libraries when in doubt on what to use or what not to use please ask me.
- If you are submitting the homework late (see the late submission policy in syllabus), we will grade your homework based on the time stamp of submission and your remaining late days.

## 1 Thumbtacks Revisited [25 pts]

### Maximum Likelihood Estimation

Let's go back to our thumbtack example in class. $\theta$ is the probability of a thumbtack falling on heads.

**a) [6 pts]** Suppose we have flipped 10 tacks, of which 6 of them landed on heads and 4 of them landed on tails. Write a program that plots the likelihood function of this data for each value of $\hat{\theta}$ in $\{0, 0.01, 0.02, \ldots, 1.0\}$ and that marks the $x$-axis for the value of $\hat{\theta}^{\text{MLE}}$ that maximizes the likelihood.

**b) [4 pts]** Create three more likelihood plots: one where $N = 5$ and the data set contains three heads and two tails; one where $N = 100$ and the data set contains 60 heads and 40 tails; and one where $N = 10$ and there are five heads and five tails.

**c) [2 pts]** What is your observation, how the likelihood functions and maximum likelihood estimates compare for the different data sets?

### Maximum a Posteriori Probability Estimation

In the maximum likelihood estimate, we treated the true parameter value $\theta$ as a fixed (non-random) number. In cases where we have some prior knowledge about $\theta$, it is useful to treat $\theta$ itself as a random variable, and express our prior knowledge in the form of a prior probability distribution over $\theta$. For example, suppose that the $X_1, \ldots, X_N$ are generated in the following way:

- First, the value of $\theta$ is drawn from a given prior probability distribution
- Second, $X_1, \ldots, X_n$ are drawn independently from a Bernoulli distribution using this value for $\theta$.

Since both $\theta$ and the sequence $D = X_1, \ldots, X_N$ are random, they have a joint probability distribution. In this setting, a natural way to estimate the value of $\theta$ is to simply choose its most probable value given its prior distribution plus the observed data $X_1, \ldots, X_N$.

$$\hat{\theta}^{\text{MAP}} = \arg\max_{\hat{\theta}} \mathbf{P}\left(\theta = \hat{\theta} \,|\, X_1, \ldots, X_N\right). \tag{1.1}$$

This is called the maximum a posteriori probability (MAP) estimate of $\theta$. Using Bayes rule, we can rewrite the posterior probability as follows:

$$\mathbf{P}\left(\theta = \hat{\theta} \,|\, X_1, \ldots, X_N\right) = \frac{\mathbf{P}\left(X_1, \ldots, X_n \,|\, \theta = \hat{\theta}\right) \mathbf{P}\left(\theta = \hat{\theta}\right)}{\mathbf{P}\left(X_1, \ldots, X_n\right)}.$$

Since the probability in the denominator does not depend on $\hat{\theta}$, the MAP estimate is given by

$$\begin{aligned}\hat{\theta}^{\text{MAP}} &= \arg\max_{\hat{\theta}} \mathbf{P}\left(X_1, \ldots, X_n \,|\, \theta = \hat{\theta}\right) \mathbf{P}\left(\theta = \hat{\theta}\right) \\ &= \arg\max_{\hat{\theta}} L(\hat{\theta}) \mathbf{P}\left(\theta = \hat{\theta}\right).\end{aligned}$$

In words, the MAP estimate for $\theta$ is the value $\hat{\theta}$ that maximizes the likelihood function multiplied by the prior distribution on $\theta$. When the prior on $\theta$ is a continuous distribution with density function $p$, then the MAP estimate for $\theta$ is given by

$$\hat{\theta}^{\text{MAP}} = \arg\max_{\hat{\theta}} L(\hat{\theta}) p(\hat{\theta}).$$

If the posterior distributions $\mathbf{P}(\theta \,|\, D)$ are in the same family as the prior probability distribution $\mathbf{P}(\theta)$, the prior and posterior are then called conjugate distributions, and the prior is called a *conjugate prior* for the likelihood function.

**d) [8 pts]** Let's go back to our thumbtack example. We have flipped $N$ tacks, of which $\alpha_H$ of them landed on heads and $\alpha_T$ of them landed on tails. Now assume we set a Beta $(\beta_H, \beta_T)$ prior over $\theta$. Derive the analytic form of the posterior distribution, what kind of distribution it follows with which parameters?
In general Beta $(a, b)$ disribution is a two-parameter distribution with range $[0, 1]$ and pdf :

$$p(a, b) = \frac{1}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt} x^{a-1}(1-x)^{b-1} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

where $\Gamma$ is gamma function[1]. The beta function, $B$[2], appears as a normalization constant to ensure that the total probability integrates to unity.

The applet here allows you to explore how the shape of the Beta distribution pdf changes with respect to the two parameters: http://web.mit.edu/jorloff/www/18.05/applets/beta-jmo.html and you may read on *Beta* distribution here: http://en.wikipedia.org/wiki/Beta_distribution.

**e) [3 pts]** Assume Beta(3,3) prior distribution for $\theta$. Suppose, as in part (c), that $N = 10$ and we observed 6 heads and 4 tails. Compare the MAP estimate to the MLE estimates, briefly comment on the difference.

**f) [2 pts]** Comment on the relationship between the MAP and MLE estimates as $N$ goes to infinity.

## 2 Building a Spam Classifier with Naive Bayes [45 pts]

Your job is to build a spam classifier that can predict whether an email is spam or not. The questions summarize the model, therefore, please read all the questions before starting coding. You are not allowed to use a Naive Bayes package; you are asked to implement.

[1] http://en.wikipedia.org/wiki/Gamma_function
[2] http://en.wikipedia.org/wiki/Beta_function

## Dataset

Your dataset is a preprocessed and modified subset of the Ling-Spam Dataset [1]. It is based on 960 real email messages from a linguistics mailing list. Emails have been preprocessed in the following ways:

- **Stop word removal:** Words like "and","the", and "of", are very common in all English sentences and are therefore not very predictive. These words have been removed from the tweets. .
- **Removal of infrequent words:** Words that occur only once are removed.
- **Removal of non-words:** Numbers and punctuation have both been removed. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character

The data has been already split into two subsets: a 11712-tweet subset for training and a 2928-tweet subset for testing (consider this as your validation set and imagine there is another test set which is not given to you). The features have been generated for you. You will use the following files:

- `tweet-train-features.csv`
- `tweet-train-labels.csv`
- `tweet-test-features.csv`
- `tweet-test-labels.csv`
- `tweet-vocab.txt`

The files that ends with `features.txt` contains the features and the files ending with `labels.txt` contains the ground truth labels.

In the feature files each row contains the feature vector for a tweet. The $j$-th term in a row $i$ is the occurrence information of the $j$-th vocabulary word in the $i$-th tweet. The size of the vocabulary is 5722. The label files include the ground truth label for the corresponding tweet, the order of the tweets (rows) are the same as the features file. That is the $i$-th row in the files corresponds to the same tweet. Any tweet is labeled as either `positive`, `negative` or `neutral`.

The file ending with `vocab.txt` is the vocabulary file in which the first element in $j$-th is the word that $j$-th feature represents and the second element is the term frequency of the word in the all data set regardless of the classes (positive, neutral and negative).

## Bag-of-Words Representation and Multinomial Naive Bayes Model

Recall the bag-of-words document representation makes the assumption that the probability that a word appears in email is conditionally independent of the word position given the class of the email. If we have a particular tweet $D_i$ with $n_i$ words in it, we can compute the probability that $D_i$ comes from the class $y_k$ as:

$$\mathbf{P}\left(D_i \,|\, Y = y_k\right) = \mathbf{P}\left(X_1 = x_1, X_2 = x_2, .., X_{n_i} = x_{n_i} \,|\, Y = y_k\right) = \prod_{j=1}^{n_i} \mathbf{P}\left(X_j = x_j \,|\, Y = y_k\right) \tag{2.1}$$

In Eq. (2.1), $X_j$ represents the $j^{th}$ position in email $D_i$ and $x_j$ represents the actual word that appears in the $j^{th}$ position in the email, whereas $n_i$ represents the number of positions in the email. As a concrete example, we might have the first email document ($D_1$) which contains 200 words ($n_1 = 200$). The document might be of positive tweet ($y_k = 1$) and the $5^{\text{th}}$ position in the tweet might have the word "well" ($x_5 =$ "well").

In the above formulation, the feature vector $\vec{X}$ has a length that depends on the number of words in the email $n_i$. That means that the feature vector for each email will be of different sizes. Also, the above formal definition of a feature vector $\vec{x}$ for a email says that $x_j = k$ if the $j$-th word in this email is the $k$-th word in the dictionary. This does not exactly match our feature files, where the $j$-th term in a row $i$ is the number of occurrences of the $j$-th dictionary word in that email $i$. As shown in the lecture slides, we can slightly change the representation, which makes it easier to implement:

$$\mathbf{P}\left(D_i \,|\, Y = y_k\right) = \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}} \tag{2.2}$$

,where $V$ is the size of the vocabulary, $X_j$ represents the appearing of the $j$-th vocabulary word and $t_{w_j,i}$ denotes how many times word $w_j$ appears in email $D_i$. As a concrete example, we might have a vocabulary of size of 1309, $V = 1309$. The first tweet ($D_1$) might be positive($y_k = 1$) and the 80-th word in the vocabulary, $w_{80}$, is "amazing" and $t_{w_{80},1} = 2$, which says the word "amazing" appears 2 times in email $D_1$. Contemplate on why these two models (Eq. (2.1) and Eq. (2.2)) are equivalent.

In the classification problem, we are interested in the probability distribution over the tweet classes (in this case positive, negative and neutral tweets) given a particular tweet $D_i$. We can use Bayes Rule to write:

$$\mathbf{P}\left(Y = y_k | D_i\right) = \frac{\mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y\right)^{t_{w_j,i}}}{\sum_k \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}}} \tag{2.3}$$

Note that, for the purposes of classification, we can actually ignore the denominator here and write:

$$\mathbf{P}\left(Y = y_k | D_i\right) \propto \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y\right)^{t_{w_j,i}} \tag{2.4}$$

$$\hat{y}_i = \arg\max_{y_k} \mathbf{P}\left(Y = y_k \,|\, D_i\right) = \arg\max_{y_k} \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}} \tag{2.5}$$

Probabilities are floating point numbers between 0 and 1, so when you are programming it is usually not a good idea to use actual probability values as this might cause numerical underflow issues. As the logarithm is a strictly monotonic function on [0,1] and all of the inputs are probabilities that must lie in [0,1], it does not have an effect on which of the classes achieves a maximum. Taking the logarithm gives us:

$$\hat{y}_i = \arg\max_{y} \left( \log \mathbf{P}\left(Y = y_k\right) + \sum_{j=1}^{V} t_{w_j,i} * \log \mathbf{P}\left(X_j \,|\, Y = y_k\right) \right) \tag{2.6}$$

, where $\hat{y}_i$ is the predicted label for the $i$-th example.

**Question 2.1 [2 points]** If the the ratio of the classes in a dataset is close to each other, it is a called "balanced" class distribution if not it is skewed. What is the percentage of spam emails in the `train.labels.txt`. Is the training set balanced or skewed towards a one of the classes?

The parameters to learn and their MLE estimators are as follows:

$$\theta_{j \,|\, y=\text{neutral}} \equiv \frac{T_{j,y=neutral}}{\sum_{j=1}^{V} T_{j,y=neutral}}$$

$$\theta_{j \,|\, y=\text{positive}} \equiv \frac{T_{j,y=positive}}{\sum_{j=1}^{V} T_{j,y=positive}}$$

$$\theta_{j \,|\, y=\text{negative}} \equiv \frac{T_{j,y=negative}}{\sum_{j=1}^{V} T_{j,y=negative}}$$

$$\pi_{y=positive} \equiv \mathbf{P}\left(Y = \text{positive}\right) = \frac{N_{positive}}{N}$$

- $T_{j,neutral}$ is the number of occurrences of the word j in neutral tweets in the training set including the multiple occurrences of the word in a single tweet.
- $T_{j,positive}$ is the number of occurrences of the word j in positive tweets in the training set including the multiple occurrences of the word in a single tweet.
- $T_{j,negative}$ is the number of occurrences of the word j in negative tweets in the training set including the multiple occurrences of the word in a single tweet.
- $N_{positive}$ is the number of positive tweets in the training set.
- $N$ is the total number of tweets in the training set.
- $\pi_{y=\text{positive}}$ estimates the probability that any particular tweet will be positive.
- $\theta_{j \,|\, y=\text{neutral}}$ estimates the probability that a particular word in a neutral tweet will be the $j$-th word of the vocabulary, $\mathbf{P}\left(X_j \,|\, Y = \text{neutral}\right)$

- $\theta_{j\,|\,y=\text{positive}}$ estimates the probability that a particular word in a positive tweet will be the $j$-th word of the vocabulary, $\mathbf{P}\left(X_j \,|\, Y = \text{ positive}\right)$
- $\theta_{j\,|\,y=\text{negative}}$ estimates the probability that a particular word in a negative tweet will be the $j$-th word of the vocabulary, $\mathbf{P}\left(X_j \,|\, Y = \text{negative}\right)$

**Question 2.2 [3 points]** How many parameters do we need to estimate for this model?

**Question 2.3 (Coding) [30 points]** Train a Naive Bayes classifier using all of the data in the training set ( `question-4-train-features.csv` and `question-4-train-labels.csv`). Test your classifier on the test data (`question-4-test-features.txt` and `question-4-test-labels.txt`), and report the testing accuracy as well as how many wrong predictions were made. In estimating the model parameters use the above MLE estimator. If it arises in your code, define $0 * \log 0 = 0$ (note that $a * \log 0$ is as it is, that is -inf ). In case of ties, you should predict "neutral". Report your test set accuracy. What did your classifier end up predicting? Why is using the MLE estimate a bad idea in this situation?

**Question 2.4 (Coding) [10 points]** Extend your classifier so that it can compute an MAP estimate of $\theta$ parameters using a fair Dirichlet prior. This corresponds to additive smoothing. The prior is fair in the sense that it "hallucinates" that each word appears additionally $\alpha$ times in the train set.

$$\theta_{j\,|\,y=neutral} \equiv \frac{T_{j,y=neutral}+\alpha}{\sum_{j=1}^{V} T_{j,y=neutral}+\alpha*V}$$

$$\theta_{j\,|\,y=positive} \equiv \frac{T_{j,y=positive}+\alpha}{\sum_{j=1}^{V} T_{j,y=positive}+\alpha*V}$$

$$\theta_{j\,|\,y=negative} \equiv \frac{T_{j,y=negative}+\alpha}{\sum_{j=1}^{V} T_{j,y=negative}+\alpha*V}$$

$$\pi_{y=positive} \equiv \mathbf{P}\left(Y = \text{positive}\right) = \frac{N_{positive}}{N}$$

For this question set $\alpha = 1$. Train your classifier using all of the training set and have it classify all of the test set and report test-set classification accuracy. Comment on the results.

IMPORTANT: In this question, while you can use libraries for simple arithmetics, you cannot use a library for implementing Naive Bayes classifier.

# 3 Logistic Regression [30 pts]

In this section you will implement a logistic regression classifier, You will use a preprocessed version of Titanic dataset, which contains data about Titanic passengers. You will predict if a passenger survived or not based on his/her age, sex and passenger class. In the sex column, 1 shows the passenger is female and 2 shows that the passenger is male.

As the data ranges vary significantly across the feature dimensions, you should normalize your features. You can do this using the following formula for each feature dimension $x_i$:

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}. \tag{3.1}$$

Remember that in logistic regression, our goal is to learn a set of parameters by maximizing the conditional log-likelihood of the data. Assuming you are given a dataset with $n$ training examples and $p$ features, the formula for the conditional log likelihood of the training data in terms of the the class labels $y^{(i)}$, the features $x_1^{(i)}, \ldots, x_d^{(i)}$, and the parameters $w_0, w_1, \ldots, w_d$, where the superscript $(i)$ denotes the sample index is given below. This will be your objective function for gradient ascent:

$$\begin{aligned} l(w_0, w_1, \ldots, w_d) &= \log \prod_{i=1}^{n} \mathbf{P}\left(y^{(i)} \,|\, x_1^{(i)}, \ldots, x_d^{(i)}; w_0, w_1, \ldots, w_d\right) \\ &= \sum_{i=1}^{n} \left[ y^{(i)}(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)}) - \log\left(1 + \exp(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)})\right) \right] \end{aligned}$$

The partial derivative of the objective function with respect to $w_0$ and with respect to an arbitrary $w_j$, are given below, you will use these to update your parameter weights according to Gradient Ascent algorithm.

$$\frac{\sigma f}{\sigma w_0} = \sum_{i=1}^{n} \left[ y^{(i)} - \frac{\exp(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)})}{1 + \exp(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)})} \right]$$

$$\frac{\sigma f}{\sigma w_j} = \sum_{i=1}^{n} x_j^{(i)} \left[ y^{(i)} - \frac{\exp(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)})}{1 + \exp(w_0 + \sum_{j=1}^{d} w_j x_j^{(i)})} \right]$$

**Question 3.1 [10 points]** First implement a function for gradient ascent optimization which takes learning rate as a parameter and then use it to implement your logistic regression classifier.

**Question 3.2 [20 points]** Split the data into three random sets, 60% as train, 20% validation and 20% as test. Tune hyper-parameters (number of iterations and learning rate) using validation set accuracy. Report your accuracy on the test set using the hyperparameters you have chosen. Report which values you tested for learning rate and the maximum number of iterations you used when tuning the model.

IMPORTANT: In this question, while you can use libraries for simple arithmetics, you cannot use a library for implementing Logistic regression classifier for the gradient ascent algorithm.

**References**

1. Twitter Airline Sentiment dataset. https://www.kaggle.com/crowdflower/twitter-airline-sentiment
2. "On Discriminative vs. Generative Classifiers: A comparison of logistic regression and Naive Bayes" by Andrew Ng and Michael I. Jordan.
3. Manning, C. D., Raghavan, P., and Schutze, H. (2008). Introduction to information retrieval. New York: Cambridge University Press.
http://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html
4. CMU Lecture Notes.
http://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture5.pdf