# EECS 545 HW 2

Due Wednesday, Sept. 29, by 11:59 PM Eastern Time via Gradescope and Canvas

When you upload your solutions to Gradescope, please indicate which pages of your solution are associated with each problem. Also please note that your code for problems 2 and 4 should be uploaded to Gradescope (as a .pdf) and Canvas (as .py) as discussed below.

## 1. Maximum Likelihood Estimation

Consider a random variable $\boldsymbol{X}$ (possibly a vector) whose distribution (density function or mass function) belongs to a parametric family. The density or mass function may be written $f(\boldsymbol{x}\,;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is called the parameter, and can be either a scalar or vector. For example, in the Gaussian family, $\boldsymbol{\theta}$ can be a two-dimensional vector consisting of the mean and variance. Suppose the parametric family is known, but the value of the parameter is unknown. It is often of interest to estimate this parameter from an observation of $\boldsymbol{X}$. Maximum likelihood estimation is one of the most important parameter estimation techniques. Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be iid (independent and identically distributed) random variables distributed according to $f(\boldsymbol{x}\,;\boldsymbol{\theta})$. By independence, the joint distribution of the observations is the product

$$\prod_{i=1}^{n} f(\boldsymbol{x}_i\,;\boldsymbol{\theta}).$$

Viewed as a function of $\boldsymbol{\theta}$, this quantity is called the *likelihood* of $\boldsymbol{\theta}$. Because many common pdfs/pmfs have an exponential form, it is often more convenient to work with the *log-likelihood*,

$$\sum_{i=1}^{n} \log f(\boldsymbol{x}_i\,;\boldsymbol{\theta}).$$

A *maximum likelihood estimator* (MLE) of $\boldsymbol{\theta}$ is a function $\widehat{\boldsymbol{\theta}}$ of the observed values $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ satisfying

$$\widehat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log f(\boldsymbol{x}_i\,;\boldsymbol{\theta}),$$

where "arg max" denotes the set of all values achieving the maximum. If there is always a unique maximizer, it is called *the* maximum likelihood estimator.

**(a)** Let $X_1, \ldots, X_n \overset{iid}{\sim} \text{bernoulli}(\theta)$, a Bernoulli trial taking values 0 and 1, and "success probability" $\theta$, where 1 corresponds to "success". Determine the maximum likelihood estimator of $\theta$ as a function of observations (realizations) $x_1, \ldots, x_n$ of $X_1, \ldots, X_n$.

**(b)** Calculate the Hessian of the log-likelihood function and verify that the estimator you found is indeed the unique maximizer of the log-likelihood function.

## 2. Naïve Bayes for Document Classification: Cars or Motorcycles?

In this problem, you will implement a Naive Bayes classifier on the Newsgroup20 dataset. The documents in the data set are messages sourced from 20 newsgroups (an online message forum). You will build a classifier to determine whether a message originated from a newsgroup about cars or motorcycles.

You should implement this algorithm yourself. Do not use existing implementations. The following section provides background for the Multinomial Naive Bayes model you will be implementing.

**Naïve Bayes Background**

For each sample in the data set, we have a $d$-dimensional feature vector $\boldsymbol{x}$, and a binary label $y \in \{0, 1\}$. Each feature $x_i$ is a count of the number of times word $j$ occurred in the $i$th document.

For a given feature vector $\boldsymbol{x} = [x_1 \ \cdots \ x_d]^T$, Naïve Bayes makes a classification by estimating quantities in the following formula for the Bayes classifier:

$$\widehat{y} = \arg\max_{k \in \{0,1\}} \pi_k \prod_{j=1}^{d} \Pr(X_j = x_j | Y = k).$$

Here, $X_j$ is the $j$th feature, viewed as a random variable, $Y$ is the label, and $\Pr(X_j = \ell | Y = k)$ is the probability that word $j$ occurred $\ell$ times in document $i$, given the document belongs to class $k$ (bag of words model).

For *multinomial Naïve Bayes*, we will model this class-conditional probability as

$$\Pr(X_j = \ell | Y = k) = (p_{kj})^{\ell}$$

using the following estimate for $p_{kj}$:

$$\widehat{p}_{kj} = \frac{n_{kj} + \alpha}{n_k + \alpha d},$$

Where $n_k = \sum_{i:y_i=k} \sum_j x_{ij}$ is the total number of words in label $k$ documents, and $n_{kj} = \sum_{i:y_i=k} x_{ij}$ is the number of times word $j$ appears in documents with label $k$, with smoothing parameter $\alpha = 1$ to ensure $\widehat{p}_{kj} > 0$.

$\pi_k$ denotes the class-$k$ priors, which are estimated as $\widehat{\pi}_k = \frac{m_k}{n}$, where $m_k = |\{i : y_i = k\}|$ is the number of label k documents.

Notice that in the classification rule, we take the product over probabilities, and we exponentiate a probability in the class-conditional estimate. In some situations, multiplying many numbers less than one can result in *underflow*, as the product becomes very small. The resulting numerical error can worsen performance of the classifier. Thankfully, underflow can be mitigated by doing operations in log-space, and converting back via exponentiation when needed.

In this problem you will implement the classifier

$$\widehat{y} = \arg\max_{k \in \{0,1\}} \log \left( \widehat{\pi}_k \prod_{j=1}^{d} (\widehat{p}_{kj})^{x_j} \right). \tag{1}$$

The log does not change the maximizer since it is a strictly increasing function. Here and throught out the course, logarithms are natural unless a different base is specified.

(a) Intuitively, $p_{kj}$ is the probability that feature $j$ appears once in a class-$k$ document. What additional assumption about the data, in addition to the naïve Bayes assumption, makes this intuition valid?

(b) By applying properties of the natural logarithm, simplify the expression on the right-hand side of (1), substituting the formula for $\widehat{p}_{kj}$. Your final expression should not contain logs of products or divisors.

(c) Download the training and test datasets from Canvas under Files → Homework → HW2 → hw2p2_data.zip. `hw2p2_train_x.npy`, `hw2p2_train_y.npy` are the training features and labels respectively, `hw2p2_test_x.npy` and `hw2p2_test_y.npy` are the test data. The data can be loaded with `numpy.load(filename)`. For example:

```
train_x = np.load("hw2p2_train_x.npy")
train_y = np.load("hw2p2_train_y.npy")
```

Here, `train_x` has dimensions $n_{train} \times d$, and `train_y` is an $n_{train}$-vector of 0s and 1s, denoting a message from the cars group or motorcycles group, respectively. The training set consists of $n_{train} = 1192$ samples, and the test set has $n_{test} = 792$ samples. There are $d = 1000$ features, where each feature denotes the number of times a particular word appeared in a document.

(i) Use the training data to estimate $\log p_{kj}$ for each class $k = 0, 1$ and for each feature $j = 1, \ldots, d$.

(ii) Also estimate the log-priors $\log \pi_k$ for each class.

Report your answers to $(ii)$.

(d) Use the estimates for $\log p_{kj}$ and $\log \pi_k$ to predict labels for the testing data. That is, apply the decision rule with the samples in `test_x.npy` and `test_y.npy`. Report the test error.

(e) What would the test error be if we always predicted the same class, namely, the majority class from the training data? (Use this result as a sanity check for the error in part (d))

(f) Submit all code used for (c), (d), and (e) as a single .py file. Submit this file to the corresponding assignment on Canvas. In addition, please also submit a .pdf version of your code (just print the .py file to pdf) and upload to gradescope for part (f). This will make it easier for graders who want to take a quick look at your code without running it. Your code should follow best coding practices including the use of comments, indentation, and descriptive variable names.

## 3. Logistic regression objective function

Consider the regularized logistic regression objective (penalized negative log-likelihood)

$$J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$$

where

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[ y_i \log \left( \frac{1}{1 + e^{-\boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i}} \right) + (1 - y_i) \log \left( \frac{e^{-\boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i}}{1 + e^{-\boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i}} \right) \right].$$

Here $\boldsymbol{\theta} = [b \; w_1 \; \cdots \; w_d]^T$, $\tilde{\boldsymbol{x}}_i = [1 \; x_{i1} \; \cdots \; x_{id}]^T$ and $y_i \in \{0, 1\}$.

**(a)** Show that if we change the label convention in logistic regression from $y \in \{0, 1\}$ to $y \in \{-1, 1\}$, then

$$-\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left( 1 + \exp \left( -y_i \boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i \right) \right).$$

Introduce the notation $\phi(t) = \log(1 + \exp(-t))$ so that, by the previous problem, the logistic regression regularized negative log-likelihood may be written

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \phi(y_i \boldsymbol{\theta}^T \tilde{\boldsymbol{x}}_i) + \lambda \|\boldsymbol{\theta}\|^2. \tag{2}$$

**(b)** Calculate the gradient of $J(\boldsymbol{\theta})$ using (2). Your answer may be in the form of a summation. Simplify your result so that it involves only the full vectors $\boldsymbol{x}_i$ (or $\tilde{\boldsymbol{x}}_i$) and not their components.

**(c)** Calculate the Hessian of $J(\boldsymbol{\theta})$. This will be a $(d+1) \times (d+1)$ matrix. Again your result may be in the form of a summation. Simply your result so that it involves only the full vectors $\boldsymbol{x}_i$ (or $\tilde{\boldsymbol{x}}_i$) and not their components.

**(d)** Use the previous result to argue that $J$ is a convex function of $\boldsymbol{\theta}$ when $\lambda \geq 0$, and strictly convex when $\lambda > 0$.

## 4. Logistic Regression for Fashion Classification

Download the files `fashion_mnist_images.npy`, `fashion_mnist_labels.npy` from Canvas under Files → Homework → HW2 → `hw2p4_data.zip`. This is a subset of the "Fashion MNIST" dataset, which contains images of different articles of clothing. This subset contains examples of coats and dresses.

The data file contains variables `x` and `y`, with the former containing images and the latter labels. The images are stored as column vectors. To visualize an image, in Python run

```
import numpy as np
import matplotlib.pyplot as plt

x = np.load("fashion_mnist_images.npy")
y = np.load("fashion_mnist_labels.npy")
d, n= x.shape

i = 0 #Index of the image to be visualized
plt.imshow(np.reshape(x[:,i], (int(np.sqrt(d)),int(np.sqrt(d)))), cmap="Greys")
plt.show()
```

The above code can be found in the file `loadfmnist.py`.

Newton's method finds a critical point of an objective function $J(\boldsymbol{\theta})$ by iterating

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - (\nabla^2 J(\boldsymbol{\theta}_t))^{-1} \nabla J(\boldsymbol{\theta}_t).$$

Implement Newton's method (a.k.a. Newton-Raphson) to find a minimizer of the regularized negative log likelihood $J(\boldsymbol{\theta})$ from the previous problem. Set $\lambda = 1$. Use the first 5000 examples as training data, and the last 1000 as test data. As a termination criterion, let $i$ be the final iteration as soon as $|J(\boldsymbol{\theta}_i) - J(\boldsymbol{\theta}_{i-1})|/J(\boldsymbol{\theta}_{i-1}) \leq \epsilon := 10^{-6}$. Let the initial iterate $\boldsymbol{\theta}_0$ be the zero vector.

(a) Report the test error, the number of iterations run, and the value of the objective function after convergence.

(b) Generate a figure displaying 20 images in a 4 x 5 array. These images should be the 20 misclassified images for which the logistic regression classifier was most confident about its prediction (you will have to define a notion of confidence in a reasonable way – explain what this is). In the title of each subplot, indicate the true label of the image. What you should expect to see is some dresses that look kind of like coats and coats that look kind of like dresses.

(c) Submit your code as a single .py file to the corresponding assignment on *Canvas* designated for this purpose. In addition, please also submit a .pdf version of your code (just print the .py file to pdf) and upload to gradescope. This will make it easier for graders who want to take a quick look at your code without running it. Your code should follow best coding practices including the use of comments, indentation, and descriptive variable names.

Some additional remarks:

- Helpful Python commands and libraries: numpy.log, numpy.exp, numpy.sum, numpy.sign, numpy.zeros, numpy.repeat, str(), matplotlib.pyplot.
- Note that the labels in the data are $\pm 1$, whereas the notes (at times) assume that the labels are 0 and 1.
- It is possible to "vectorize" Newton's method, that is, implement it without any additional loops besides the main loop. If you would prefer to do this, please consult the book by Hastie, Tibshirani, and Friedman and look for the iterative reweighted least squares (IRLS) implementation. Do note that they may have different notation than what was presented in class. That said, if you just use an additional loop to calculate the Hessian at each iteration, and it shouldn't take too long.