EECS 545 – Machine Learning - Homework #2

Due: 5:00PM 10/04/2018

Homework Submission: Homeworks must be submitted via Gradescope (https://gradescope.com/) as pdf files. This includes your code when appropriate. Please use a high quality scanner if possible, as found at the library or your departmental copy room. If you must use your phone, please don't just take photos, at least use an app like CamScanner that provides some correction for shading and projective transformations. The gradescope entry code for this course is MG2XEE, use this code to add the course to your account. If you have never used gradescope before, please familiarize yourself with it well before the homework deadline. They offer tutorials for students, and the best way to learn is to upload a draft of your solutions well before the deadline.

1) **Maximum Likelihood Estimation (15 pts).**

Consider a random variable $\mathbf{X}$ (possibly a vector) whose distribution (pdf or pmf) belongs to a parametric family. The density or mass function may be written as $f(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is called the parameter, and can be either a scalar or vector. For example, in the univariate Gaussian distribution, $\boldsymbol{\theta}$ can be a two dimensional vector consisting of the mean and the 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 observations of $\boldsymbol{x}$.

Maximum likelihood estimation is one of the most important parameter estimation techniques. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be i.i.d. (independent and identically distributed) realizations drawn from $f(\boldsymbol{x}, \boldsymbol{\theta})$. By independence, the joint distribution of the observations is the product

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

Viewed as a function of $\boldsymbol{\theta}$, this quantity is called the *likelihood* of $\boldsymbol{\theta}$. It is often more convenient to work with the *log-likelihood*,

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

A maximum likelihood estimate (MLE) of $\boldsymbol{\theta}$ is any parameter

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

If the maximizer is unique, $\hat{\boldsymbol{\theta}}$ is called *the* maximum likelihood estimate of $\boldsymbol{\theta}$.

(a) (5 pts) Consider i.i.d random variables $x_1, \ldots, x_n$ from Gamma distribution with pdf

$$f(x; \alpha, \lambda) = \frac{1}{\Gamma(\alpha)} \lambda^{\alpha} x^{\alpha-1} \exp\big(-\lambda x\big), \quad x \geq 0.$$

Suppose the parameter $\alpha > 0$ is known, find the maximum likelihood estimate of λ.

(b) (10 pts) Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be i.i.d. d-dimensional Gaussian random variables distributed according to $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. That is,

$$f(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right).$$

Showing your work, derive the maximum likelihood estimate of the mean vector $\boldsymbol{\mu}$. You can assume the covariance matrix Σ is known.

2) **Bayesian spam filtering (15 pts).**

In this problem you will apply the naïve Bayes classifier to the problem of spam detection, using a benchmark database assembled by researchers at Hewlett-Packard. Download the file `spambase.data` from Canvas under Files → Data and Helper Functions, and issue the following commands to load the data. In Matlab:

```
z = dlmread('spambase.data',',');
rng(0); % initialize the random number generator
rp = randperm(size(z,1)); % random permutation of indices
z = z(rp,:); % shuffle the rows of a
x = z(:,1:end-1);
y = z(:,end);
```

In Python:

```
import numpy as np
z = np.genfromtxt('spambase.data', dtype=float, delimiter=',')
np.random.seed(0) #Seed the random number generator
rp = np.random.permutation(z.shape[0]) #random permutation of indices
z = z[rp,:] #shuffle the rows of z
x = z[:,:-1]
y = z[:,-1]
```

Note: If you copy and paste the above, you may need to delete and retype the single quote to avoid an error. This has to do with the optical character recognition of pdfs on some systems.

Here `x` is $n \times d$, where $n = 4601$ and $d = 57$. The different features correspond to different properties of an email, such as the frequency with which certain characters appear. `y` is a vector of labels indicating spam or not spam. For a detailed description of the dataset, visit the UCI Machine Learning Repository, or Google 'spambase'.

To evaluate the method, treat the first 2000 examples as training data, and the rest as test data. Fit the naïve Bayes model using the training data (i.e., estimate the class-conditional marginals), and compute the misclassification rate (i.e., the test error) on the test data. The code above randomly permutes the data, so that the proportion of each class is about the same in both training and test data.

Note: On the spam detection problem, please note that you will get a different test error depending on how you quantize values that are **equal** *to the median. It makes a difference whether you quantize values equal to the median to 1 or 2. You should quantize all medians the same way – I'm not suggesting that you try all* 2^d *combinations. So just make sure you try both options, and report the one that works better.*

(a) (10 pts) Quantize each variable to one of two values, say 1 and 2, so that values below the median map to 1, and those above map to 2.

To get the median in Matlab, use the `median` command. In Python, use `np.median(a,axis=0)`.

Report the test error. As a sanity check, what would be the test error if you always predicted the same class, namely, the majority class from the training data?

(b) (5 pts) Submit your code via gradescope (concise, well-organized and clearly commented, as usual).

3) **Logistic regression Hessian (15 pts).**

Determine a formula for the gradient and the Hessian of the regularized logistic regression objective function. Argue that the objective function

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

is convex when $\lambda \geq 0$, and that for $\lambda > 0$, the objective function is strictly convex.

Hints: The following conventions and properties regarding vector differentiation may be useful. The properties can be easily verified from definitions. Try to avoid long, tedious calculations.

- If $f : \mathbb{R}^n \to \mathbb{R}$, then we adopt the convention

$$\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} := \nabla f(\mathbf{z}).$$

- If $f : \mathbb{R}^n \to \mathbb{R}^m$, where $f(\mathbf{z}) = [f_1(\mathbf{z}) \cdots f_m(\mathbf{z})]^T$, then we adopt the convention

$$\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} := \begin{bmatrix} \frac{\partial f_1(\mathbf{z})}{\partial z_1} & \frac{\partial f_1(\mathbf{z})}{\partial z_2} \cdots & \\ \frac{\partial f_2(\mathbf{z})}{\partial z_1} & \frac{\partial f_2(\mathbf{z})}{\partial z_2} \cdots & \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} \nabla f_1(\mathbf{z})^T \\ \nabla f_2(\mathbf{z})^T \\ \vdots \end{bmatrix} \in \mathbb{R}^{m\times n}.$$

- If $f : \mathbb{R}^n \to \mathbb{R}^m$, adopt the convention

$$\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}^T} := \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right)^T.$$

- Given these conventions, it follows that the Hessian $\mathbf{H}$ of J is

$$\mathbf{H} = \frac{\partial}{\partial \boldsymbol{\theta}^T}\left(\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right),$$

  which is often denoted more concisely as

$$\frac{\partial^2 J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}.$$

- (One form of a multivariable chain rule): If $f(\mathbf{z}) = g(h(\mathbf{z}))$ where $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}$, then

$$\nabla f(\mathbf{z}) = \nabla h(\mathbf{z}) \cdot g'(h(\mathbf{z})).$$

4) **Handwritten digit classification with logistic regression (15 pts).**

Download the file `mnist_49_3000.mat` from Canvas under Files → Data and Helper Functions. This is a subset of the MNIST handwritten digit database, which is a well-known benchmark database for classification algorithms. This subset contains examples of the digits 4 and 9.

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

```
load mnist_49_3000
[d,n] = size(x);
i = 1; % index of image to be visualized
imagesc(reshape(x(:,i),[sqrt(d),sqrt(d)])') % notice the transpose
```

In Python type

```
import numpy as np
import scipy.io as sio
import matplotlib.pyplot as plt

mnist_49_3000 = sio.loadmat('mnist_49_3000.mat')
x = mnist_49_3000['x']
y = mnist_49_3000['y']
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)))))
plt.show()
```

Note: If you copy and paste the above, you may need to delete and retype the single quote to avoid an error. This has to do with the optical character recognition of pdfs on some systems.

Implement Newton's method (a.k.a. Newton-Raphson) to find a minimizer of the regularized negative log likelihood $J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}\|^2$. Regularization and Newton's method are described at the end of the LR notes. Try setting $\lambda = 10$. Use the first 2000 examples as training data, and the last 1000 as test data.

(a) (5 pts) Report the test error, your termination criterion (multiple options here), and the value of the objective function at the optimum.

(b) (5 pts) In addition, 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 a bunch of 4s that look kind of like 9s and 9s that look kind of like 4s.

(c) (5 pts) Upload your well-organized, clearly commented code to gradescope.

Helpful Matlab commands: `log, exp, .*, ./, sum, min, find, sign, zeros, ones, repmat, figure, subplot, title, num2str`.

Helpful Python commands and libraries: numpy.log, numpy.exp, numpy.sum, numpy.sign, numpy.zeros, numpy.repeat, str(), matplotlib.pyplot.

Some additional remarks:

- Note that the labels in the data are ± 1, whereas the notes (at times) assume that the labels are 0 and 1.
- Please initialize with the zero vector.
- 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 doesn't take too long.

5) **Weighted Least Squares Regression (10 pts).**

Consider linear regression and let $c_1, \ldots, c_n > 0$ be known weights. Determine the solution of

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} c_i(y_i - \mathbf{w}^T \boldsymbol{x}_i - b)^2.$$

Express your solution in terms of the matrix $C = \operatorname{diag}(c_1, \ldots, c_n)$ and an appropriate data matrix X.

Hints: Try to reduce the problem to ordinary least squares, and then follow the "alternate" solution of OLS described in the notes. As a sanity check, your solution should reduce to $(X^T X)^{-1} X^T \tilde{\mathbf{y}}$ when C is the identity matrix.