COS 598 – Machine Learning - Homework #2

Due: 5:00PM 02/26/2021

Homework Submission: Homeworks must be submitted via Brightspace 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 (if possible), at least use an app like CamScanner that provides some correction for shading and projective transformations.

In problems 2 do NOT use "sklearn" library to import NB classifier.

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 $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n$ 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 $x_1,\dots,x_n$ from Gamma distribution with pdf

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

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

(b) (10 pts) Let $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n$ 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 Brightspace under HW2, and issue the following commands to load the data. 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 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?

Note: In class you may learn the Laplace Smoothing technique but in this problem you **don't** need to implement this technique.

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