CMSC 35400 / STAT 37710

Spring 2020

Homework 3

1. Consider a sequence of independent coin tosses (i.i.d. Bernoulli random variables)

$$\boldsymbol{x} = [x_1, \ldots, x_N]^T$$

Let $x_n = 1$ denote "heads" and $x_n = 0$ "tails," and $\mathbb{P}(x_n = 1) = \theta$, $\mathbb{P}(x_n = 0) = 1 - \theta$. The likelihood function is

$$p(\boldsymbol{x}|\theta) \;=\; \theta^{s(\boldsymbol{x})}(1-\theta)^{N-s(\boldsymbol{x})}$$

where $s(\boldsymbol{x}) = \sum_{n=1}^{N} x_n$. Suppose we are interested in estimating θ given $\boldsymbol{x}$. Let's take a Bayesian approach. *A priori* we *believe* that $\theta \approx 1/2$ (*i.e.,* we're tossing a reasonably fair coin), and we know $0 \leq \theta \leq 1$.

a) Show that a beta density prior for θ reflects this prior information. The beta density is given by

$$p(\theta; \alpha) = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\theta^{\alpha-1}(1-\theta)^{\alpha-1}$$

where $\alpha \geq 1$ is a shape parameter to be specified by the user and Γ is the Euler gamma function $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$.

SOLUTION: The mean of $B(\alpha, \alpha)$, is $\frac{\alpha}{2\alpha} = \frac{1}{2}$. This reflects the fact that we expect $\theta \approx \frac{1}{2}$. In addition, the beta distribution is tightly concentrated around $1/2$.

b) Show that the beta density is a conjugate prior for the Bernoulli distribution. That is, show that the posterior density also has a beta form.

SOLUTION: Assume that $\theta \sim \text{Beta}(\alpha, \alpha)$. Note that

$$\begin{aligned}
p(x) &= \int_0^1 p(x|\theta)p(\theta)d\theta \\
&= \int_0^1 \theta^{s(x)}(1-\theta)^{N-s(x)}\frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\theta^{\alpha-1}(1-\theta)^{\alpha-1}d\theta \\
&= \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2}\int_0^1 \theta^{s(x)+\alpha-1}(1-\theta)^{N-s(x)+\alpha-1}d\theta \\
&= \frac{\Gamma(2\alpha)\Gamma(s(x)+\alpha)\Gamma(N-s(x)+\alpha)}{\Gamma(\alpha)^2\Gamma(N+2\alpha)}.
\end{aligned}$$

Using this we can compute the posterior,

$$\begin{aligned}
p(\theta|x) &= \frac{p(x|\theta)p(\theta)}{p(x)} \\
&= \frac{\Gamma(N+2\alpha)}{\Gamma(s(x)+\alpha)\Gamma(N-s(x)+\alpha)}\theta^{s(x)+\alpha-1}(1-\theta)^{N-s(x)+\alpha-1} \\
&\sim \text{Beta}(s(x)+\alpha, N-s(x)+\alpha)
\end{aligned}$$

This is a beta itself implying that the Beta distribution is a conjugate prior for the Bernoulli.

c) Find the MLE, posterior mean estimator $\mathbb{E}[\theta|X]$, and MAP estimator. How does α effect estimator performance? What is the asymptotic (large N) behavior of the estimator?
SOLUTION:

MLE:

$$\begin{aligned}\hat{\theta}_{\text{MLE}} &= \arg\max_{\theta} \theta^{s(x)}(1-\theta)^{N-s(x)} \\ &= \arg\max_{\theta} -\log(\theta^{s(x)}(1-\theta)^{N-s(x)}) \\ &= \arg\min_{\theta} s(x)\log(\theta) + (N-s(x))\log(1-\theta) \\ &= \frac{s(x)}{N}\end{aligned}$$

Note that $s(x) \sim \text{Binomial}(N, \theta)$, so we can compute:

$$\begin{aligned}\text{bias}(\hat{\theta}_{\text{MLE}}) &= 0, \\ \text{var}(\theta) &= N\theta(1-\theta), \\ \text{MSE}(\hat{\theta}) &= N\theta(1-\theta)\end{aligned}$$

Posterior Mean:

$$\begin{aligned}\hat{\theta}_{PM} &= \int_0^1 \theta p(\theta|x)d\theta \\ &= \frac{\Gamma(N+2\alpha)}{\Gamma(s(x)+\alpha)\Gamma(N-s(x)+\alpha)} \frac{\Gamma(s(x)+\alpha+1)\Gamma(N-s(x)+\alpha)}{\Gamma(N+2\alpha+1)} \\ &= \frac{s(x)+\alpha+1}{N+2\alpha+1}\end{aligned}$$

and

$$\begin{aligned}\text{bias}(\hat{\theta}_{\text{MLE}}) &= \frac{N\theta+\alpha+1}{N+2\alpha+1}, \\ \text{var}(\theta) &= \frac{1}{(N+2\alpha+1)^2} N\theta(1-\theta), \\ \text{MSE}(\hat{\theta}) &= \left(\frac{N\theta+\alpha+1}{N+2\alpha+1}\right)^2 + \frac{N\theta(1-\theta)}{(N+2\alpha+1)^2}\end{aligned}$$

MAP:

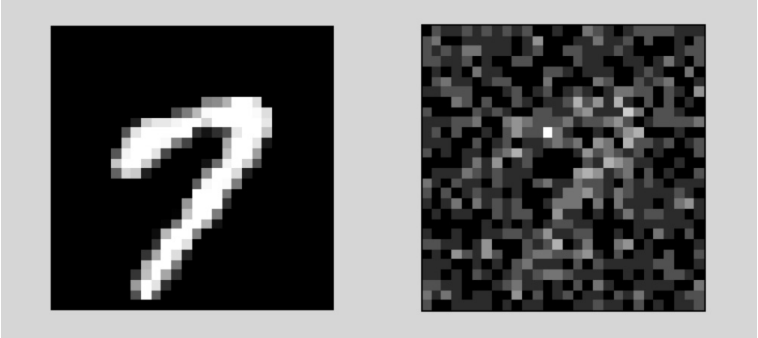
$$\begin{aligned}\hat{\theta}_{MAP} &= \arg\max_{\theta} p(\theta|y) \\ &= \arg\max_{\theta} \frac{\Gamma(N+2\alpha)}{\Gamma(s(x)+\alpha)\Gamma(N-s(x)+\alpha)} \theta^{s(x)+\alpha-1}(1-\theta)^{N-s(x)+\alpha-1} \\ &= \frac{s(x)+\alpha-1}{N+\alpha-1}\end{aligned}$$

$$\text{bias}(\hat{\theta}_{MAP}) = \frac{N\theta + \alpha - 1}{N + 2\alpha - 1},$$
$$\text{var}(\theta) = \frac{N\theta(1-\theta)}{(N + 2\alpha - 1)^2},$$
$$\text{MSE}(\hat{\theta}) = \left(\frac{N\theta + \alpha - 1}{N + 2\alpha - 1}\right)^2 + \frac{N\theta(1-\theta)}{(N + 2\alpha - 1)^2}$$

Note that α acts as a parameter that biases the mean.

2. Suppose that $y_i \sim \text{Poisson}(\mu_i)$, $i = 1, \ldots, n$. Derive a GLM for this model (i.e., the means are related by a nonlinear transformation of a linear model $X\theta$, where $X \in \mathbb{R}^{n \times p}$ is known and $\theta \in \mathbb{R}^p$ is an unknown parameter).

Next consider the images below. The image on the left shows a handwritten digit. The image on the right is a simulation of a low-photon count observation of this image (as though we took a picture of the digit in a very dark room). Comparing the two images, you might be able to make out the digits pattern in the image of photon counts, but alone the image on the right probably isn't recognizable. In this problem, we will use the GLM developed above to estimate the original image from the count data.



The Matlab file `sevens.mat` contains 1000 images of handwritten versions of the number 7. The file contains a two dimensional array $d \in \mathbb{R}^{784 \times 1000}$. Each column of d is a vectorized version of a 28×28 image of a digits. To view the ith image you can extract and reshape the ith column of d and display it as follows:

```
%matlab
imagesc(reshape(d(:,i),28,28)); colormap(gray)+
```

```
#python
data = sio.loadmat('sevens.mat')['d']
data = data.reshape(28,28,-1,order='F')
plt.imshow(data[:,:,0],cmap = 'gray')
```

a) Use the dataset to find a low-dimensional subspace of $\mathbb{R}^{784}$ that most of the digits (approximately) belong to. It should be possible to find decent subspace that has $r = 25$ and $r = 50$ dimensions. Let $X \in \mathbb{R}^{784 \times r}$ denote an orthobasis for this subspace.

The Matlab file `mean.mat` contains a vector `mu` $\in \mathbb{R}^{784}$, which is one ideal digit image. Generate Poisson distributed count data using the command `y = poissrnd(mu)`. If you reshape and display `mu` and `y`, then you should see images that look like the two above.

SOLUTION:

```
%matlab
load('sevens.mat');
load('mean.mat');
[U,D,V] = svd(d);
X = d*V(:,1:50);
y = poissrnd(mu)
```

```
#python
d = sio.loadmat('PS2_data_files/sevens.mat')['d']
mu = sio.loadmat('PS2_data_files/mean.mat')['mu']
d = d.reshape(28,28,-1).transpose(1,0,2).reshape(28 * 28, -1)
[U,D,Vt] = np.linalg.svd(d)
V = Vt.T
X = d @ V[:,1:50]
y = np.random.poisson(mu)
```

b) Consider modeling the natural parameter of the Poisson distribution with a linear model $X\theta$, where $\theta \in \mathbb{R}^r$. The idea is that the natural parameter lies in the subspace you determined in (a). Use the function `glmfit` in Matlab (or another software package) to compute the MLE of θ.

SOLUTION:

```
%matlab
 w = glmfit(X, y(:), 'poisson');
 yhat = glmval(w, X, 'log');
```

```
#python
w = sm.GLM(y.reshape(-1), X, family=sm.families.Poisson
(link=sm.families.links.log))
w_result = w.fit()
yhat = w.predict(w_result.params,linear = False)
#equal to: yhat = np.exp(X @ w_result.params)
```

c) Compare the estimated image you obtain from the GLM (don't forget to transform from

the natural parameter to the mean) to the image obtained from the solution to

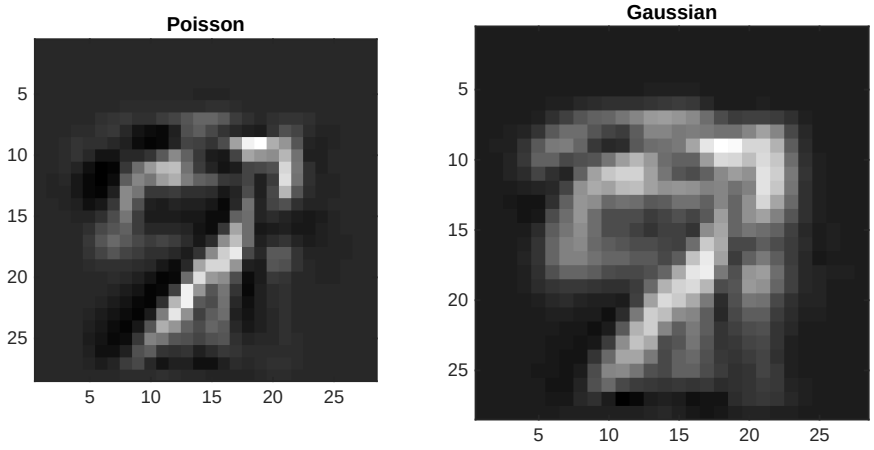
$$\min_{\theta} \|y - X\theta\|_2^2 \ .$$

This is the solution we would obtain if we modeled our data as Gaussian instead of Poisson.

SOLUTION:

```
%matlab
figure; imagesc(reshape(yhat,28,28)); colormap(gray)
yhatg = X*(X'*X)^(-1)*X'*y(:)
figure; imagesc(reshape(yhatg,28,28)); colormap(gray)
```

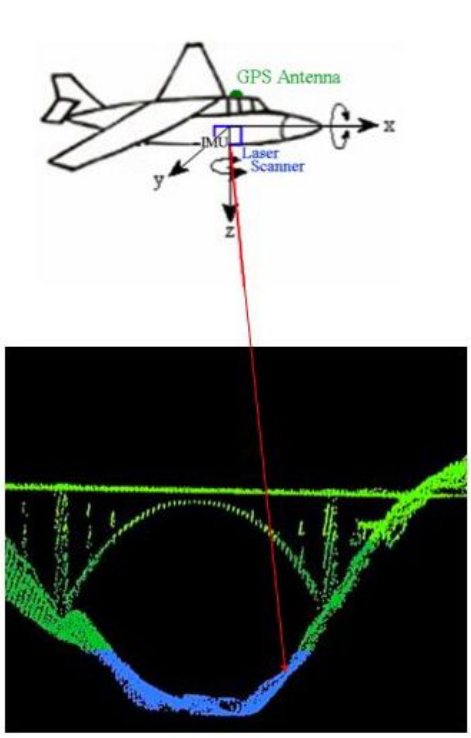
```
#python
yhatg = X @ (X.T @ X).T @ X.T @ y.reshape(-1)
plt.imshow(np.reshape(yhatg,(28,28)),cmap = 'gray')
plt.savefig('yhatg')
```

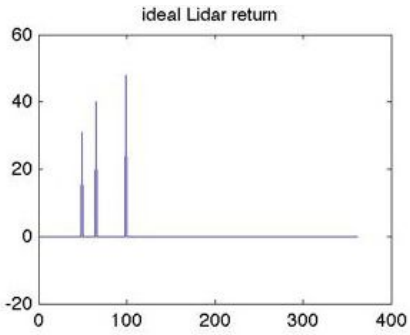


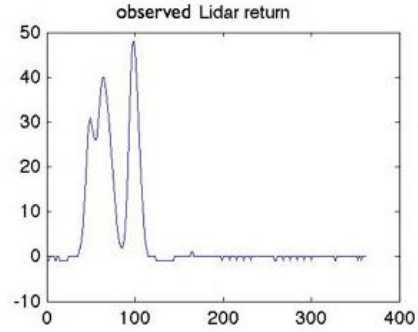
As we can see, the image that used the poisson link function is much better.

3. LIDAR uses the time-delays of reflections of a laser pulse to determine the distance of objects, as shown in the figure below. The laser pulse can be modeled as an ideal impulse convolved

with a Gaussian bump. The LIDAR "return" signal is an set of impulses (at locations commesurate with the distances of reflecting objects) convolved with a Gaussian impulse response function plus additive Gaussian noise. The goal is to deconvolve the LIDAR return to obtain the "ideal" return, consisting of a few non-zero impulses. Let θ denote the ideal LIDAR return, X denote the Gaussian blurring operation, and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Then we can express the observed LIDAR return as $y = X\theta + \epsilon$.







Simulate the LIDAR return in Matlab or Python as follows:

a) Generate an ideal LIDAR return consisting of a few impulses at random locations. This should be a sparse vector θ of dimension 256×1. Then simulate the noisy and blurred return using the matrix X included in the file `lidar_blur.mat` and use $\sigma = 10^{-2}$.

SOLUTION: a) We use the following matlab code:

```
%matlab
load('lidar_blur.mat');
theta = (rand(256,1)>.9);
sigma = 10^-2;
Sigma = sigma^2*eye(256);
y = X*theta+ mvnrnd(zeros(256,1), Sigma)';
```

```
#python
X = sio.loadmat('PS2_data_files/lidar_blur.mat')['X']
theta = (np.random.rand(256)>0.9).astype(np.float32)
```

```
sigma = 10**-2
Sigma = sigma**2 * np.eye(256)
y = X @ theta + np.random.multivariate_normal(np.zeros(256),
Sigma)
```

b) Pose the deconvolution as a Maximum Likelihood Estimation.

SOLUTION: The MLE is given by

$$\hat{\theta}_{\text{MLE}} = \arg\min_{\theta} ||X\theta - y||^2$$

If X^TX was invertible, a solution this would be given by $\hat{\theta} = (X^TX)^{-1}X^Ty$. However, in this case X^TX is not full rank! As a result, there are infinitely many possible values and it's not clear what to choose for θ. A natural choice is to choose a θ with small norm. This is exactly the result of taking a Bayesian prior through regularization.

c) Use a Gaussian prior $\theta \sim \mathcal{N}(0, \gamma^2 I)$ to devise a Bayesian estimation procedure. Experiment with different values of γ. Compare the Bayesian estimates to the MLE in multiple simulations.

SOLUTION: From class and the Gauss-Markov theorem, using the Gaussian prior, $\theta \sim N(0, \gamma^2 I)$, gives us ridge regression,

$$\hat{\theta} = (X^TX + \frac{\sigma^2}{\gamma^2} I)^{-1}X^Ty$$

In general the MLE solution in Matlab gives us nonsense, with values of θ on the order of 10^{30}. By using ridge regression, we get much more reasonable values of θ and smaller values of error.

```
%matlab
for t=1:10
    gamma = rand()*10;
    thetat = (X'*X+sigma^2/gamma^2*eye(256))^(-1)*(X'*y);
    norm(thetat-theta);
end
```

```
#python
for t in range(10):
    gamma = np.random.rand()*10
    thetat = np.linalg.inv((X.T @ X + sigma**2 / gamma**2 *
    np.eye(256))) @ (X.T @ y)
    np.linalg.norm(thetat - theta)
```