CSE 546 HW1

DAVID FLEMING

Contents

Question 1

1.1. The expectation value for $max(X_1, X_2)$ is given by

$$E[X] = \int max(X_1, X_2) f(x) dx_1 dx_2 \tag{1}$$

where $f(x) = 1$ is the PDF for the uniform random variates. We consider two regions, one below the line defined by $X = X_1 = X_2$ and one above said line for our integrations. This gives us the following integral:

Date: October 14th, 2016.

$$\begin{aligned}
E[X] &= \int_0^1 \int_{x_2}^1 x_1 dx_1 dx_2 + \int_0^1 \int_{x_1}^1 x_2 dx_2 dx_1 \\
&= \int_0^1 (\frac{1}{2} - \frac{1}{2}x_2^2) dx_2 + \int_0^1 (\frac{1}{2} - \frac{1}{2}x_1^2) dx_1 \\
&= (\frac{1}{2}x_2 - \frac{1}{6}x_2^2)|_0^1 + (\frac{1}{2}x_1 - \frac{1}{6}x_1^2)|_0^1 \\
&= \frac{2}{3}
\end{aligned} \tag{2}$$

1.2. Since

$$Var[X] = E[X^2] - E[X]^2 \tag{3}$$

and we now know $E[X] = 2/3$, we solve the integral presented above but with the integrand squared as follows

$$\begin{aligned}
E[X^2] &= \int_0^1 \int_{x_2}^1 x_1^2 dx_1 dx_2 + \int_0^1 \int_{x_1}^1 x_2^2 dx_2 dx_1 \\
&= \int_0^1 (\frac{1}{3} - \frac{1}{3}x_2^3) dx_2 + \int_0^1 (\frac{1}{3} - \frac{1}{3}x_1^3) dx_1 \\
&= (\frac{1}{3}x_2 - \frac{1}{12}x_2^4)|_0^1 + (\frac{1}{3}x_1 - \frac{1}{12}x_1^4)|_0^1 \\
&= \frac{1}{2}
\end{aligned} \tag{4}$$

which when combined with the definition for $Var[X]$ gives

$$Var[X] = \frac{1}{2} - \left(\frac{2}{3}\right)^2 = \frac{1}{18} \tag{5}$$

1.3. Since

$$Cov[X, X_1] = E[XX_1] - E[X]E[X_1] \tag{6}$$

and we know $E[X]$ from 1.1 and $E[X_1] = 1/2$ is the trivial result for the uniform distribution from $[0, 1]$, we need only calculate the first term as follows:

$$\begin{aligned}
E[XX_1] &= \int_0^1 \int_{x_2}^1 x_1^2 dx_1 dx_2 + \int_0^1 \int_{x_1}^1 x_2 x_1 dx_2 dx_1 \\
&= \int_0^1 (\frac{1}{3} - \frac{1}{3}x_2^3) dx_2 + \int_0^1 x_1(\frac{1}{2} - \frac{1}{2}x_1^2) dx_1 \\
&= (\frac{1}{3}x_2 - \frac{1}{12}x_2^4)|_0^1 + (\frac{1}{4}x_1^2 - \frac{1}{8}x_1^4)|_0^1 \\
&= \frac{3}{8}
\end{aligned} \tag{7}$$

giving us

$$Cov[X, X_1] = E[XX_1] - E[X]E[X_1] = \frac{3}{8} - \frac{2}{3}\frac{1}{2} = \frac{1}{24} \tag{8}$$

Note: For this question, I collaborated with Matt Wilde.

Question 2

2.1. For the log-likelihood of G given λ and i.i.d. samples, we have

$$\begin{aligned} LL &= \log\left(P(G|\theta)\right) \\ &= \Pi_i^n P(G_i|\theta) \\ &= \log\left(\frac{\lambda^{\Sigma_i^n k_i} e^{-\lambda n}}{\Pi_i^n k_i!}\right) \\ &= \Sigma_i^n k_i \log \lambda - \lambda n - \log \Pi_i^n k_i! \end{aligned} \tag{9}$$

2.2. To compute the MLE for λ in general,

$$\frac{\partial}{\partial \lambda}[LL] = 0 \tag{10}$$

which yields

$$\begin{aligned} 0 &= \frac{\partial}{\partial \lambda}\left(\Sigma_i^n k_i \log \lambda - \lambda n - \log \Pi_i^n k_i!\right) \\ &= \frac{\Sigma_i^n k_i}{\lambda} - n \\ \hat{\lambda}_{MLE} &= \frac{\Sigma_i^n k_i}{n} \end{aligned} \tag{11}$$

2.3. For the observed set G, I use Eqn. 11 to compute λ_{MLE} as

$$\hat{\lambda}_{MLE} = \frac{4+1+3+5+5+1+3+8}{8} = 3.75 \tag{12}$$

Note: For this question, I collaborated with Matt Wilde.

Question 3

TODO

Question 4

Note: Both both subquestions 1 and 2, I assume that the too small or too large λs bias the model to a too complex or too simple model relative to the optimum model complexity, respectively.

4.1: Too Small a λ.

4.1.a. For both LASSO and Ridge Regression (RR), the error on the training set would decrease as the penalty term for both regression techniques would be effectively negligible. With small λ, the regularization penalty is also small causing both regression techniques to tend towards least squares regression (LSR) and allow for more complex models which overfit the training data and hence leading to smaller training set error. For example, too small of a λ could push both LASSO and RR to favor a high-order polynomial model when the underlying data is linear.

4.1.b. For both LASSO and RR, too small a λ leads to overfitting on the training set. With a model overfit on the training set, it will do a poor job of generalizing to new data and hence will poorly fit the testing set leading to larger testing error.

4.1.c. For both LASSO and RR, too small a λ yields too small of a complexity penalty causing both algorithms to tend towards the LSR solution. In this case, $\hat{\omega}$ could get large via overfitting. RR, however, will likely predict larger $\hat{\omega}$ than LASSO as RR's l_2 norm penalty primarily seeks to control the magnitude of the weight vector, hence biasing towards larger values in this case as the penalty decreases, while LASSO seeks a sparse solution.

4.1.d. With LASSO, too small of a λ would yield more non-zero parameters since a smaller regularization penalty will prevent many of the elements of $\hat{\omega}$ from being set to 0 under LASSO's sharp l_1 norm penalty which tries to make $\hat{\omega}$ sparse. For RR, too small of a λ would have little effect on the number of non-zero parameters as RR's regularization penalty deals with constraining the magnitude of $\hat{\omega}$ as opposed to forcing it to be sparse.

4.2: Too Large a λ.

4.2.a. For both regression techniques, too large of a λ will yield larger training set error as the regularization penalty will select for simpler models that could poorly fit the data. For example, too large of a λ could push both LASSO and RR to favor a linear model when the underlying data is quadratic.

4.2.b. For both LASSO and RR, too large of a λ will yield larger testing set error since the models again will be biased towards lower-than-optimal model complexity and hence will likely yield a poor fit to future data similar to why the error on the training set is also larger in this case.

4.2.c. For both LASSO and RR, too large of a λ will yield smaller $\hat{\omega}$ as the larger regularization penalty restricts the weight vector. RR, however, will likely predict smaller $\hat{\omega}$ than LASSO as RR's l_2 norm penalty primarily seeks to control the magnitude of the weight vector, hence biasing towards smaller values in this case, while LASSO seeks a sparse solution.

4.2.d. For LASSO, a larger λ will yield a sparser solution via its l_1 norm regularization penalty and hence will have fewer nonzero elements in $\hat{\omega}$ than RR. In the large λ limit, RR will also yield fewer nonzero elements in $\hat{\omega}$ only because its large regularization penalty makes coefficients in its weight vector small and hence potentially pushing some towards 0.

Note: For this question, I collaborated with Matt Wilde.

Question 5

TODO

Question 6

Note: The following scripts (attached) include all code used to solve this question: `hw1_6.py, mnist_utils.py, regression_utils.py, ridge_utils.py`.

Here, the I build a linear classifier by minimizing the following Ridge Regression Loss:

$$\min_{\omega} = \frac{1}{N}\Sigma_{i=1}^{N}(y_i - \omega \cdot x_i)^2 + \lambda ||\omega||^2. \tag{13}$$

In general, the RR regression weight vector can be solved for analytically via the following

$$\hat{\omega_{ridge}} = (\lambda I_D + X^T X)^{-1} X^T y \tag{14}$$

however the matrix inversion required is not only computationally expensive but also can be numerically unstable. Instead, I minimize Equation 13 by following the formulae in the textbook (see Murphy Section 7.5.2) which involves centering the data and augmenting X with the Cholesky decomposition of the precision matrix and y with a vector of 0s. This yields the MAP estimate

$$\hat{\omega_{ridge}} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T \tilde{y}. \tag{15}$$

6.1. I adopt a regularization constant $\lambda = 1$. For the purpose of classification, I label a digit as 2 if $\omega \cdot x > c$. To compute the threshold c, I first take the training labels and set all instances of 2 to 1 and all other digits to 0. Then, I mask the training labels to get a vector that encodes which rows of the training data yield a 2. Next, I fit the training set with my ridge regression model via Equation 15 and compute my predictions over the training set $\hat{y} = \omega_0 + X_{train}\hat{\omega}$. Then, using my previously computed mask, I find the entries of my predictions $\hat{y}$ that should be labeled as 2s according to the true training labels. I take the median of these values to find an approximate threshold $c = 0.589$. When I apply this model to the testing set, and predicted value with $\omega \cdot x > c$ will be set to 1 and 0 otherwise.

After training my model, I find a 0/1 and a square loss of 3059 on the training set. These values are the same because my predictions are either 0 or 1 and the true labels are either 0 or 1. Note that this loss does not seem terribly high for my heuristic classifier given that there are of order 6,000 2's in the MNIST training set. However, even a naive classifier that says everything is not a 2 would be correct roughly 90% of the time, therefore I expect my model that does a simple prediction with a poor linear Ridge Regression to do slightly better.

6.2. On the testing set, I evaluated my model from fitting the training set and apply it to the testing input images to find a 0/1 and a square loss of 545. Note that it appears that my error on the testing set is actually less than that of the training set! This, however, is not the case since the training set contains 60,000 samples while the testing set contains 10,000. If the size of the set over which the model is evaluated is taken into account, my model performs worse on the testing set and hence the testing error is larger as is expected for classifying with a linear regression model.

6.3. Linear regression is a poor idea for classification since the slope of the predicted line decides which input vector is labeled a 0 and which is labeled a 1. Instead of ascribing a probability to each estimate, it instead checks to see if it's above or below a given threshold. This model does not successfully generalize to new data at all since new data points since the slope of the classifying line is so dependent on the training data and new points could dramatically shift the slope of that line, changing which class a given point belongs to.

6.4. TODO

Note: For this question, I collaborated with Matt Wilde.