Homework Assignment # 2

By: Cody Rosevear
ccid: rosevear
Due: Sunday, October 29, 2017, 11:59 p.m.
Total marks: 100

Question 1. [25 MARKS]

Let $X_1, \ldots, X_n$ be i.i.d. Gaussian random variables, each having an unknown mean θ and known variance σ_0^2.

(a) [5 MARKS] Assume θ is itself selected from a normal distribution $\mathcal{N}(\mu, \sigma^2)$ having a known mean μ and a known variance σ^2. What is the maximum a posteriori (MAP) estimate of θ?

$$\begin{aligned}
p(\theta|D) &= \prod_{i=1}^{n} N(\theta, \sigma_0^2) N(\mu, \sigma^2) && (1)\\
\log p(\theta|D) &= \log \prod_{i=1}^{n} N(\theta, \sigma_0^2) N(\mu, \sigma^2) && (2)\\
&= \log \prod_{i=1}^{n} N(\theta, \sigma_0^2) + \log N(\mu, \sigma^2) && (3)\\
&= \sum_{i=1}^{n} (\log N(\theta, \sigma_0^2)) + \log N(\mu, \sigma^2) && (4)\\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-(x_i-\theta)^2}{(2\sigma_0^2)}\right) + \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(\theta-\mu)^2}{(2\sigma^2)}\right) && (5)\\
&= \sum_{i=1}^{n} \frac{-(x_i-\theta)^2}{(2\sigma_0^2)} - \log\sqrt{2\pi\sigma_0^2} - \frac{(\theta-\mu)^2}{(2\sigma^2)} - \log\sqrt{2\pi\sigma^2} && (6)\\
&= -\frac{1}{2\sigma_0^2} \sum_{i=1}^{n} (x_i-\theta)^2 - \log\sqrt{2\pi\sigma_0^2} - \frac{(\theta-\mu)^2}{(2\sigma^2)} - \log\sqrt{2\pi\sigma^2} && (7)\\
& && (8)
\end{aligned}$$

Then, differentiating with respect to θ yields:

$$\frac{1}{\sigma_0^2} \sum_{i=1}^{n} (x_j - \theta) - 0 - \frac{\theta-\mu}{\sigma^2} - 0 \qquad (9)$$

$$(10)$$

Setting to 0 and solving for θ yields:

$$\begin{aligned}
0 &= \frac{1}{\sigma_0^2} \sum_{i=1}^{n} (x_j - \theta) - \frac{\theta-\mu}{\sigma^2} && (11)\\
\frac{\theta-\mu}{\sigma^2} &= \frac{1}{\sigma_0^2} \sum_{i=1}^{n} (x_j - \theta) && (12)
\end{aligned}$$

$$\frac{\theta-\mu}{\sigma^2} = \frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j) - \frac{n\theta}{\sigma_0^2} \tag{13}$$

$$\frac{\theta-\mu}{\sigma^2} + \frac{n\theta}{\sigma_0^2} = \frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j) \tag{14}$$

$$\frac{\theta}{\sigma^2} - \frac{\mu}{\sigma^2} + \frac{n\theta}{\sigma_0^2} = \frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j) \tag{15}$$

$$\frac{\theta}{\sigma^2} + \frac{n\theta}{\sigma_0^2} = \frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j) + \frac{\mu}{\sigma^2} \tag{16}$$

$$\theta(\frac{1}{\sigma^2} + \frac{n}{\sigma_0^2}) = \frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j) + \frac{\mu}{\sigma^2} \tag{17}$$

$$\theta = \frac{\frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j) + \frac{\mu}{\sigma^2}}{\frac{1}{\sigma^2} + \frac{n}{\sigma_0^2}} \tag{18}$$

(b) [10 MARKS] Assume θ is itself selected from a Laplace distribution $\mathcal{L}(\mu, b)$ having a known mean (location) μ and a known scale (diversity) b. Recall that the pdf for a Laplace distribution is

$$p(x) = \frac{1}{2b}\exp\left(\frac{-|x-\mu|}{b}\right)$$

For simplicity, assume $\mu = 0$. What is the maximum a posteriori estimate of θ? If you cannot find a closed form solution, explain how you would use an iterative approach to obtain the solution.

$$P(\theta|D) = \prod_{i=1}^{n} N(\theta, \sigma_0^2)L(\mu, b) \tag{19}$$

$$\log P(\theta|D) = \log(\prod_{i=1}^{n} N(\theta, \sigma_0^2)L(\mu, b)) \tag{20}$$

$$= \log\prod_{i=1}^{n} N(\theta, \sigma_0^2) + \log L(\mu, b) \tag{21}$$

$$= \sum_{i=1}^{n} \log N(\theta, \sigma_0^2) + \log L(\mu, b) \tag{22}$$

$$= \sum_{i=1}^{n}(\log\frac{1}{\sqrt{2\pi\sigma_0}}\exp\left(\frac{-1}{2}(\frac{x_j-\theta}{\sigma_0})^2\right)) + \log(\frac{1}{2b}\exp\left(\frac{-|\theta-0|}{b}\right)) \tag{23}$$

$$= \sum_{i=1}^{n}(\frac{-1}{2}(\frac{x_j-\theta}{\sigma_0})^2 - \log\sqrt{2\pi\sigma_0}) - \frac{|\theta|}{b} - \log 2b \tag{24}$$

Then, differentiating with respect to θ yields:

$$\frac{1}{\sigma_0^2}\sum_{i=1}^{n}(x_j-\theta) - 0 - \frac{|\theta|}{b\theta} - 0 \tag{25}$$

$$\tag{26}$$

Setting this equation to 0 and finding a closed form solution would be irksome due to the presence of absolute values, which suggests the existence of non differentiable points in the original cost function. One option to tackle this problem would be to use a proximal method in conjunction with gradient descent. The gradient descent algorithm would search through the function along those points where it is in fact differentiable, while relying on a proximal operator that sets the grdient to 0 for values that pass a certin threshold with respect to how close to 0 they are.

(c) [10 MARKS] Now assume that we have **multivariate** i.i.d. Gaussian random variables, $\mathbf{X}_1, \ldots, \mathbf{X}_n$ with each $\mathbf{X}_i \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma}_0)$ for some unknown mean $\boldsymbol{\theta} \in \mathbb{R}^d$ and known $\boldsymbol{\Sigma}_0 = \mathbf{I} \in \mathbb{R}^{d \times d}$, where $\mathbf{I}$ is the identity matrix. Assume $\boldsymbol{\theta} \in \mathbb{R}^d$ is selected from a zero-mean multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = \sigma^2\mathbf{I})$ and a known variance parameter σ^2 on the diagonal. What is the MAP estimate of $\boldsymbol{\theta}$?

$$p(\theta|D) = \prod_{i=1}^{n} N(\theta, \Sigma_0^2) N(\mu, \Sigma^2) \tag{27}$$

$$\log p(\theta|D) = \log \prod_{i=1}^{n} N(\theta, \Sigma_0^2) N(\mu, \Sigma^2) \tag{28}$$

$$= \log \prod_{i=1}^{n} N(\theta, \Sigma_0^2) + \log N(\mu, \Sigma^2) \tag{29}$$

$$= \sum_{i=1}^{n} (\log N(\theta, \Sigma_0^2)) + \log N(\mu, \Sigma^2) \tag{30}$$

$$= \sum_{i=1}^{n} \log \frac{1}{det(2\pi\Sigma_0)} \exp\left(-\frac{1}{2}(x_i - \theta)'\Sigma_0^{-1}(x_i - \theta)\right) + \log \frac{1}{det(2\pi\Sigma)} \exp\left(-\frac{1}{2}(\theta)'\Sigma^{-1}(\theta)\right) \tag{31}$$

$$= \sum_{i=1}^{n} -\frac{1}{2}(x_i - \theta)'\Sigma_0^{-1}(x_j - \theta) - \log det(2\pi\Sigma_0) - \frac{1}{2}(\theta)'\Sigma_0^{-1}(\theta) - \log det(2\pi\Sigma) \tag{32}$$

Then, differentiating with respect to θ yields:

$$\sum_{i=1}^{n} \Sigma_0^{-1}(x_i - \theta) - 0 - \Sigma^{-1}\theta - 0 \tag{33}$$

Setting to 0 and solving for θ yields:

$$0 = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i - \theta) - \Sigma^{-1}\theta \tag{34}$$

$$\Sigma^{-1}\theta = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i - \theta) \tag{35}$$

$$\Sigma^{-1}\theta = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i) - (\Sigma_0^{-1} n\theta) \tag{36}$$

$$\Sigma^{-1}\theta + \Sigma_0^{-1} n\theta = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i) \tag{37}$$

$$\theta(\Sigma^{-1} + \Sigma_0^{-1} n) = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i) \tag{38}$$

$$\theta(\Sigma^{-1} + \Sigma_0^{-1} n) = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i) \tag{39}$$

$$\theta = \sum_{i=1}^{n} \Sigma_0^{-1}(x_i)(\Sigma^{-1} + \Sigma_0^{-1} n)^{-1} \tag{40}$$

Question 2. [75 MARKS]

In this question, you will implement variants of linear regression. We will be examining some of the practical aspects of implementing regression, including for a large number of features and samples. An initial script in python has been given to you, called `script_regression.py`, and associated python files. You will be running on a UCI dataset for CT slices[1], with 385 features and 53,500 samples. Baseline algorithms, including mean and random predictions, are used to serve as sanity checks. We should be able to outperform random predictions, and the mean value of the target in the training set.

(a) [5 MARKS] The main linear regression class is `FSLinearRegression`. The FS stands for FeatureSelect. The provided implementation has subselected features and then simply explicitly solved for $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$. Increase the number of selected features (up to all the features). What do you find? How can this be remedied?

Attempting to run linear regression with the full feature set resulted in failure due to the singularity of $(\mathbf{X}^\top \mathbf{X})$. The singular nature of the matrix is either likely to due to the presence of redundant features, which results in a lack of linear independence between the columns of the feature matrix, and/or because of near linear dependence due to small singular values. This can be addressed by adding regularization to the regression algorithm, which will shift or truncate the small singular values, as revealed by singular value decomposition, that are creating numerical instability in the solution, or one can use the pseudoinverse to solve the problem in closed form.

(b) [5 MARKS] The current code averages the error over multiple training, test sets, subsampled from the data. Modify the code to additionally report the standard error over these multiple runs (i.e., the sample standard deviation divided by the square root of the number of runs).

(c) [5 MARKS] Now implement Ridge Regression, where a ridge regularizer $\lambda \|\mathbf{w}\|_2^2$ is added to the optimization. Run this algorithm on all the features. How does the result differ from (a)? Discuss the result in a couple of sentences, for one regularization parameter, $\lambda = 0.01$.

Now that regualrization has been added there is no longer a singular matrix error occurring when one attempts to use all of them features for learning.

(d) [20 MARKS] Now imagine that you want to try a feature selection method and you've heard all about this amazing and mysterious Lasso. Lasso can often be described as an algorithm, or otherwise as an objective with a least-squares loss and ℓ_1 regularizer. It is more suitably thought of as the objective, rather than an algorithm, as there are many algorithms to solve the Lasso.

[1] `https://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis`

Implement an iterative solution approach that uses the soft thresholding operator (also called the shrinkage operator), described in the chapter on advanced optimization techniques.

(e) [20 MARKS] Implement a stochastic gradient descent approach to obtaining the linear regression solution (see the chapter on advanced optimization techniques). Report the error, for a step-size of 0.01 and 1000 epochs.

(f) [20 MARKS] Implement batch gradient descent for linear regression, using line search. Compare stochastic gradient descent to batch gradient descent, in terms of the number of times the entire training set is processed. Set the step-size to 0.01 for stochastic gradient descent. Report the error versus epochs, where one epoch involves processing the training set once. Report the error versus runtime.