Homework Assignment #2

Due: Wednesday, October 14, 2015, 11:59 p.m.
Total marks: 110

Question 1. [15 marks]

Suppose that the number of accidents occurring daily in a certain plant has a Poisson distribution with an unknown mean λ. Based on previous experience in similar industrial plants, suppose that our initial feelings about the possible value of λ can be expressed by an exponential distribution with parameter $\theta = \frac{1}{2}$. That is, the prior density is

$$f(\lambda) = \theta e^{-\theta\lambda}$$

where $\lambda \in (0, \infty)$. Assume there are 79 accidents over the next 9 days.

(a) [5 marks] Determine the maximum likelihood estimate of λ.

Solution

Maximum Likelyhood by definition:

$$\lambda_{ML} = \underset{\lambda\in(0,\infty)}{\arg\max} \{p(\mathcal{D}|\lambda)\}.$$

We can write the likelihood function as

$$\begin{aligned} p(\mathcal{D}|\lambda) &= p(\{x_i\}_{i=1}^n |\lambda) \\ &= \prod_{i=1}^n p(x_i|\lambda) \\ &= \frac{\lambda^{\sum_{i=1}^n x_i} \cdot e^{-n\lambda}}{\prod_{i=1}^n x_i!}. \end{aligned}$$

To find λ that maximizes the likelihood, we will first take a logarithm (a monotonic function) to simplify the calculation, then find its first derivative with respect to λ, and finally equate it with zero to find the maximum.

Specifically, we express the log-likelihood ll(D,λ) = ln p(D|λ) as

$$ll(\mathcal{D}, \lambda) = \ln\lambda \sum_{i=1}^n x_i - n\lambda - \sum_{i=1}^n \ln(x_i!)$$

and proceed with the first derivative as

$$\begin{aligned} \frac{\partial ll(\mathcal{D}, \lambda)}{\partial\lambda} &= \frac{1}{\lambda}\sum_{i=1}^n x_i - n \\ &= 0. \end{aligned}$$

$$\lambda_{ML} = \frac{1}{n}\sum_{i=1}^n x_i$$

It is given in the question that 79 accidents happen over 9 days,

Therefore,

λ_{ML} = 79/9

(b) [5 marks] Determine the maximum a posteriori estimate of λ

Solution.

As explained in the notes example 9 of Chapter 2,
Given,

$$\lambda_{MAP} = \arg\max_{\lambda \in (0,\infty)} \{p(\mathcal{D}|\lambda)p(\lambda)\}.$$

Solving by taking log for P(D|λ) as in the first question, and multiplying it with the our new given Prior Distribution

$p(\lambda) = \theta e^{-\theta\lambda}$, we get

$$\log(p(D\backslash\lambda)p(\lambda) = \sum_{i=1}^{n} x_i \log(\lambda) - \lambda(n+\theta) + \log(\theta) - \sum_{i=1}^{n} \log(x_i!)$$

Maximizing the above equation, we get

$$0 = \frac{\sum_{i=1}^{n} x_i}{\lambda} - (n+\theta)$$

$$\lambda = \frac{\sum_{i=1}^{n1} x_i}{n+\theta}$$

$\lambda = 79/(9 + ½) =$ **158/19**

(c) [5 marks] Look at the plots of some exponential distributions to better understand the prior chosen on λ. Imagine that now new safety measures have been put in place and you believe that the number of accidents per day should sharply decrease. For example, maybe you now believe that 4 accidents per day would be a pretty high estimate. How might you change θ to better reflect this new belief about the number of accidents?

Solution

As solved in the above question, we now know that

$$\lambda = \frac{\sum_{i=1}^{n1} x_i}{n+\theta}$$

$$\Rightarrow \sum_{i=1}^{n1} x_i = (n+\theta)\lambda$$

Now, it is given to us in this question that accidents have decreased sharply such that

$$\sum_{i=1}^{n1} x_i < 4n$$

Substituting the value of $\sum_{i=1}^{n1} x_i$ in this equation we get

$(n+\theta)\lambda < 4n$ or **θ < 4 - λ**

Question 2. [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 $\mathsf{N}(\mu, \sigma^2)$ having a known mean μ and a known variance σ^2. What is the maximum a posteriori (MAP) estimate of θ?

Solution. Again, we'll proceed in a similar way to calculate MAP estimate of θ with the new given functions. Here,

$$p(x_i|\theta) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(x_i-\theta)^2}{2\sigma_0^2}}$$

$$p(\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta-\mu)^2}{2\sigma^2}}.$$

Applying the MAP formula and taking log

$$log(p(\mathcal{D}|\theta)p(\theta)) = log\left(\frac{1}{(2\pi)^{(n+1)/2}\sigma_0^n\sigma}\right) - \frac{\sum_{i=1}^{n}(x_i-\theta)^2}{2\sigma_0^2} - \frac{(\theta-\mu)^2}{2\sigma^2}$$

Maximizing,

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

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

$$\frac{\sum_{i=1}^{n}(x_i)}{\sigma_0^2} - \frac{\mu}{\sigma^2} = \theta\left(\frac{n}{\sigma_0^2} - \frac{1}{\sigma^2}\right)$$

$$\frac{\sum_{i=1}^{n}(x_i)}{\sigma_0^2} - \frac{\mu}{\sigma^2} = \theta\left(\frac{n}{\sigma_0^2} - \frac{1}{\sigma^2}\right)$$

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

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

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

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.

Solution

Again solving for MAP, by taking log and maximizing we get

$$\frac{\sum_{i=1}^{n}(x_i - \theta)}{\sigma_0^2} = -\frac{(\theta - \mu)}{|\theta - \mu| b}$$

As μ can be assumed to zero, we can say

$$\sum_{i=1}^{n}(x_i) - n\theta = \left(\frac{(sign\ of\ \theta)\sigma_0^2}{b}\right)$$

$$n\theta = \left(\frac{sign\ of\ \theta}{b}\right)\sigma_0^2 + \sum_{i=1}^{n}(x_i)$$

$$\theta = \frac{\left(\frac{sign\ of\ \theta}{b}\right)\sigma_0^2 + \sum_{i=1}^{n}(x_i)}{n}$$

As sign of θ cannot be definite, we cannot have a closed form solution by this approach.

But, we can solve this problem by using some iterative algorithm because we can approach the minimum value from both the positive or negative side by using an iterative algorithm. We can iterate the algorithm and make sure the f(x) value is decreasing and reaching to the minimum. Using Gradient Descent would be a good choice for this problem.

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

Solution.

This needs to be solved in the similar way as 2(a), it just need more matrix manipulations.

I am not repeating the MAP part and directly implementing the Matrix calculations

$$\frac{d}{d\Theta} = \frac{-d}{2d\Theta} [\, \sigma^2 \Theta^T\Theta + \sum (X^T X - X^T\Theta) - (\Theta^T X + \Theta^T\Theta)]$$

$$0 = (-1/2) [\, 2 \sigma^2 \Theta + \sum (0 - X_i - X_i + 2\Theta)]$$

$$0 = (-1/2) [2 \sigma^2 \Theta + 2 \sum X_i + 2n\Theta]$$

$$0 = \sum X_i - (n + \sigma^2) \Theta$$

$$\Theta_{MAP} = \frac{\sum X_i}{(n + \sigma^2)}$$

Question 3. [10 marks]

Program the two iterative algorithms, gradient descent and Newton's method, to find the minimum of a function of two-dimensional variable $\mathbf{x} = (x_1, x_2)$

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Set the step length to $\eta = 1$ and try two different starting points$^{0)} = (1.2,\ 1.2)$ and a more difficult $x^{(0)} = (-1.2,\ 1)$. Reduce the step size to some $\eta < 1$ and repeat the minimization. Show all your work and discuss the optimization processes you tested.

Solution3.

I started using the gradient descent function with learning rate of 1 and observed

1. The function f(x) value was increasing rather than decreasing.
2. The gradient and derivative of x1 and x2 values were oscillating between positive and negative with no definite path which suggests that I was missing the minimum value.
3. I started decreasing my learning rate and observed that f(x) value kept on increasing till learning rate 0.003.
4. At learning rate .0025, the gradient decent function may get stuck and follow a zig zag path and started oscillating
5. I inferred that I had to again decrease the learning rate to approach the solution.
6. But even decreasing the learning rate to very small, it started oscillating which led me to conclude that it is quite a sharp point at minimum
7. At learning rate of 0.0001 after more than 300,000 iterations I conclude the minimum value is approximately **x={1,1)**
8. Further, I solved using the Newton method using the Hessian and found out the solution as **x={1,1).**
9. Then I also solved with starting values (-1.2,1) and reached the same solution but I took more number of iterations to solve for (-1.2,1)
10. I have attached my python scripts

Question 4. [60 marks]

In this question, you will implement linear regression and Poisson regression. An initial script in python has been given to you, called script_classify.py, and associated python files. You will be running on a blog dataset, with 230 features and 60,000 samples. Note that the first 50 features are constants for each sample, giving information about the features, and so are removed when the data is loaded. 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.

We will be examining some of the practical aspects of implementing regression. As a suggestion, you should start or complete Question 3 before doing this question.

(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 $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Increase the number of selected features (including all the way to including all the features). What do you find? How can this be remedied?

Solution. My Findings are as follows –

- As I started increasing the number of features, the Accuracy value altered with increase or decrease with no standard pattern.
- My observations of data is as follows –

```
sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
[0, 1, 2]
```

Accuracy for features: 3 FS: 556.293743969
[0, 1, 2, 3, 4]
Accuracy for features: 5 FS: 881.007534525
[0, 1, 2, 3, 4, 5, 6]
Accuracy for features: 7 FS: 770.77512472
[0, 1, 2, 3, 4, 5, 6, 7, 8]
Accuracy for features: 9 FS: 699.410940435
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Accuracy for features: 11 FS: 1026.99457323

- The reason for this should be that the accuracy value should become better with increase in features and our predictor become better, but more features may also lead to overfitting and accuracy value increase
- Further increasing the lambda value to around 15 started giving error. The error message is because of Singular Matrix that the inverse of matrix could not be calculated
- We can remedy this scenario by Singular Value decomposition. In practise, datasets include large number of features which are sometimes identical, similar, or nearly linearly dependent. This causes XTX to no longer be invertible and produces solutions that are sensitive to perturbations in y and X. As specified in Sec 4.5 of the class notes, this can be done using $X = U\Sigma V^T$ and solving for it.

(b) [10 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 results for different regularization parameter λ values. Modify the code to report error averaged over multiple splits of the data (at least 10 splits).

Solution

To implement Ridge regression we have to add $\lambda \mathbf{w}^2$ for optimization
Solving for this the weight matrix can be changed to

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}.$$

I created another function in the algorithms.py script for Ridge Regression and observed following values for different datasets

sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
Running learner = RidgeLinearRegression
Accuracy for RidgeLinearRegression: 999.436920753
Running learner = FSLinearRegression
Accuracy for FSLinearRegression: 2871.78416882
Running learner = Random
Accuracy for Random: 48889.6882209
Running learner = Mean
Accuracy for Mean: 1268.46721111

sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
Running learner = RidgeLinearRegression
Accuracy for RidgeLinearRegression: 1768.99623863
Running learner = FSLinearRegression
Accuracy for FSLinearRegression: 1928.10740339
Running learner = Random
Accuracy for Random: 6951.60154009
Running learner = Mean
Accuracy for Mean: 2203.0116

sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows

Running learner = RidgeLinearRegression
Accuracy for RidgeLinearRegression: 345.005157553
Running learner = FSLinearRegression
Accuracy for FSLinearRegression: 881.845094548
Running learner = Random
Accuracy for Random: 10133.922496
Running learner = Mean
Accuracy for Mean: 317.823511111

sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
Running learner = RidgeLinearRegression
Accuracy for RidgeLinearRegression: 774.30073132
Running learner = FSLinearRegression
Accuracy for FSLinearRegression: 2399117.44798
Running learner = Random
Accuracy for Random: 22233.5520104
Running learner = Mean
Accuracy for Mean: 841.525111111

sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
Running learner = RidgeLinearRegression
Accuracy for RidgeLinearRegression: 179.231003782
Running learner = FSLinearRegression
Accuracy for FSLinearRegression: 9349.6850972
Running learner = Random
Accuracy for Random: 55875.207374
Running learner = Mean
Accuracy for Mean: 300.0516

From this, I infer that RidgeLinearRegression always provides better solution than FSLinearRegression. Also, for some particular cases I observed Mean outperformed RidgeLinearRegression

Then, I ran the RidgeLinearRegression for different values of lamda, (0.0001, 0.01, 1, 100, 10000)
I am pasting down two different kind of observations below –
sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
Running learner = RidgeLinearRegression
Accuracy for lamda: 0.0001 RidgeLinearRegression: 506.765033696
Running learner = RidgeLinearRegression
Accuracy for lamda: 0.01 RidgeLinearRegression: 506.763039656
Running learner = RidgeLinearRegression
Accuracy for lamda: 1 RidgeLinearRegression: 506.569704608
Running learner = RidgeLinearRegression
Accuracy for lamda: 100 RidgeLinearRegression: 502.425143157
Running learner = RidgeLinearRegression
Accuracy for lamda: 10000 RidgeLinearRegression: 496.282708682

sohil@master:~/Downloads/ML/regression$ python script_classify.py
Split 5000 rows into train=300 and test=100 rows
Running learner = RidgeLinearRegression
Accuracy for lamda: 0.0001 RidgeLinearRegression: 146.931448927
Running learner = RidgeLinearRegression
Accuracy for lamda: 0.01 RidgeLinearRegression: 146.931209975
Running learner = RidgeLinearRegression
Accuracy for lamda: 1 RidgeLinearRegression: 146.913495069
Running learner = RidgeLinearRegression
Accuracy for lamda: 100 RidgeLinearRegression: 146.610703879
Running learner = RidgeLinearRegression
Accuracy for lamda: 10000 RidgeLinearRegression: 146.727799951

The first experiment show that by adding regularization and bigger value of lambda the accuracy value decreased which is good. But as observant in the second experiment adding very high values of lamda can also increase the accuracy value.

(c) [10 marks] Imagine that the dataset size continues to grow, which causes the matrix $\boldsymbol{X} \in \mathrm{R}^{n \times d}$ for n samples and d features to become quite large. One option is to go back to subselecting features. In FSLinearRegression, add an approach to select 10 features and explain your choice. How does accuracy change compared to the original FSLinearRegression and RidgeRegression? What happens to the accuracy of the solution if you add ridge regularization with this subset of features?

Solution

I could think of a few possible solutions for doing feature selection –

The best approach I would like to use for feature selection is to apply Ridge Regression and find out the weights of all the features. Then select the features with maximum weight. The reason for this will be because they have the most important weight in the function and ignoring others with lower weights will not impact the function very much.

Another approach could be using variance on all the features and checking which feature has the maximum variance, because the maximum feature will have more effect on the function.

Filter methods perform feature selection as a preprocessing step, independently of the learning algorithm used for model construction. An example of such a mechanism is variable ranking, using e.g. correlation coefficients between each feature and the dependent variable. Another filter approach selects features based on a linear model (this corresponds to the filter preprocessing step), and then constructs a non-linear model using the selected features

Wrapper methods are characterized as being a subset selection approach. The main idea of wrapper methods is to assess subsets of variables according to their usefulness to a given learning algorithm. Here, the learning algorithm is treated as a black box, and the best subset of features is determined according to the performance of the particular algorithm applied to build a regression model (e.g. linear or non-linear regression, neural networks, support vector machines, among others). Wrapper methods need a criterion to compare the performance of different feature subsets

We can also use the Least mean square to check if our features have predicted correct value, and use available algorithms to select features which have minimum LMS value and select the best 10 out this way.

(d) [15 marks] Now imagine that your dataset has gotten even larger, to the point where dropping features is not enough. Instead of removing features, implement a stochastic optimization approach to obtaining the linear regression solution (see Section 4.5.3). Explain your implementation choices.

Solution

To implement Stochastic Batch Gradient Descent, I would write an iterative gradient descent algorithm. And iterate it to a descent choice of number of iterations, say about 1500 could be a good choice.

I will start with any weight matrix of zero weights and iterative improvise it.

I am copying the code snippet I implemented -

```
num_of_iterations=1500
alpha = 0.001

self.weights = np.zeros(num_of_inputs)
for i in range(num_of_iterations):
        self.weights = self.weights - alpha/len(ytrain) * (np.dot(Xless.T,np.dot(Xless, self.weights) - ytrain))
         print self.weights
```

Now, as our number of examples will increase, it will take very long time to minimize the function.

We have to therefore start with a very high value of step size and then keep on decreasing it.

Selecting alpha = 1, and making it alpha/$i^{-1/2}$ at every step will be a good choice.

(e) [20 marks] Next you notice that the target is always positive, with many zeros and small numbers and a few large values. These target values look a little like they could come from a Poisson distribution. You recall that generalized linear models allow non-linear transfers on the linear solution, and that conveniently the Poisson distribution happens to have a nice link function. Implement Poisson regression on this data. Hint: using an exponential transfer can cause numerical instabilities. For training, you might consider scaling down the target so that there are not such large values. Moreover, the approach can be somewhat sensitive to features being collinear. You could consider subselecting some number of features once again. Explain the choices you use. You should be able to obtain better performance than linear regression.

Solution

We need to implement Generalized Linear model in this question. Generalized linear models (GLMs) extend ordinary least-squares regression beyond Gaussian probability distributions and linear dependencies between the features and the target.

We shall first revisit the main points of the ordinary least-squares regres-sion. There, we assumed that a set of i.i.d. data points with their targets D = {(xi, yi)} were drawn according to some distribution p(x, y). We also assumed that an underlying relationship between the features and the target was linear, i.e.

$$Y = \sum_{j=0}^{k} \omega_j X_j + \varepsilon,$$

We can use an approach can be somewhat sensitive to features being collinear.
As the collinear features increase, correlation between them will also increase

Discussed solutions with Anirudh and Rohit