CS 434 HW1

Tristan Gavin

April 20, 2021

1 Statistical Estimation

Q1. Maximum Likelihood Estimation of λ

1. Write out the log-likelihood function $logP(D|\lambda)$ assuming a poisson distribution.

$$l(\lambda) = P(D|\lambda) = \prod_{i=1}^{N} P(x_i|\lambda) \tag{1}$$

$$= \prod_{i=1}^{N} \frac{\lambda^{x_i \lambda} e^{-\lambda}}{x_i!} \tag{2}$$

$$ll(D|\lambda) = \sum_{i=1}^{N} log[\frac{e^{-\lambda}\lambda^{x_i}}{x_i!}] \tag{3}$$

$$= \sum_{i=1}^{N} [-\lambda + x_i log(\lambda) - log(x_i!)] \tag{4}$$

$$= -N\lambda + log(\lambda) \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} log(x_i!) \tag{5}$$

2. Take derivative of the log-likelihood with respect to the pramater λ

$$\frac{d}{d\lambda} ll(D|\lambda) = \frac{d}{d\lambda}(-N\lambda + log(\lambda) \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} log(x_i!)) \tag{6}$$

$$= -N + \frac{1}{\lambda} \sum_{i=1}^{N} x_i \tag{7}$$

3.Set the derivative equal to zero and solve for λ

$$-N + \frac{1}{\lambda} \sum_{i=1}^{N} x_i = 0 \tag{8}$$

$$N = \frac{1}{\lambda} \sum_{i=1}^{N} x_i \tag{9}$$

$$\frac{\sum_{i=1}^{N} x_i}{N} = \lambda \tag{10}$$

This answer makes intuitive sense. We are summing all values in our data set and divided them by the number of values in our data set to get an average.

Q2. Maximum A Posteriori Estimate of λ with a Gamma Prior

1. Write out the log-posterior $logP(\lambda|D)$ as proportional to $logP(D|\lambda) + logP(\lambda)$ (Assume λ is distributed according to a Gamma distribution)

$$logP(\lambda|D) = logP(D|\lambda) + logP(\lambda) \tag{11}$$
$$= [-N\lambda + log(\lambda)\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} log(x_i!)] + [log(\frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)})] \tag{12}$$
$$= [-N\lambda + log(\lambda)\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} log(x_i!)] + [\alpha log(\beta) + (\alpha-1)log(\lambda) - \beta\lambda - log(\Gamma(\alpha))] \tag{13}$$

2. Take the derivative of $logP(D|\lambda) + logP(\lambda)$ with respect to λ There are a lot of terms that are dropped because they are constants. this leaves us with a pretty simple derivative.

$$\frac{d}{d\lambda}[logP(D|\lambda) + logP(\lambda)] = \frac{d}{d\lambda}[(-N\lambda + log(\lambda)\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} log(x_i!)) \tag{14}$$
$$+ (\alpha log(\beta) + (\alpha-1)log(\lambda) - \beta\lambda - log(\Gamma(\alpha)))] \tag{15}$$
$$= \frac{1}{\lambda}(\alpha-1) - \beta - N + \frac{1}{\lambda}\sum_{i=1}^{N} x_i \tag{16}$$

3. Set the derivative to zero and solvbe for λ

$$\frac{1}{\lambda}(\alpha-1) - \beta - N + \frac{1}{\lambda}\sum_{i=1}^{N} x_i = 0 \tag{17}$$
$$\frac{1}{\lambda}((\alpha-1) + \sum_{i=1}^{N} x_i) = \beta + N \tag{18}$$
$$\lambda_{map} = \frac{\alpha - 1 + \sum_{i=1}^{N} x_i}{\beta + N} \tag{19}$$

Q3. Deriving the Posterior of A Poisson-Gammna Model

$$P(x_i|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{20}$$
$$P(\lambda) = \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)} \tag{21}$$
$$P(x_i|\lambda) \propto \lambda^{x_1+x_2+\cdots} e^{-N\lambda} \tag{22}$$
$$P(\lambda|x_i) \propto P(x_i|\lambda)P(\lambda) \tag{23}$$
$$= \lambda^{x_1+x_2+\cdots} e^{-N\lambda} \lambda^{\alpha-1} e^{-\beta\lambda\Gamma(\alpha)} \tag{24}$$

In (24) we can drop the $\frac{\beta^\alpha}{\Gamma(\alpha)}$ of the gamma prior becuase it is not in terms of *lambda*. Combining like terms we get

$$\lambda^{(x_1+x_2+\ldots)+\alpha-1} e^{-(\beta+N)\lambda} = gamma(\sum_{i=1}^{N} x_i + \alpha, \beta + N) \tag{25}$$

2 k-Nearest Neighbor

Q4. Encoding Distances

Increasing integer encoding would give greater weights to certain nominal values than others. Having different weights would make some values increase the distance more than other values of the same type. If we do not want nominal values to have an effect on our distance then it is important to use one-hot encoding. One hot encoding ensures that all nominal values will have the same weight of 1 and therefore will not effect our distance calculation.

Q5 Looking at Data

Adding up all the data entries that make > \$50k and dividing those by the total data sets ($\frac{1967}{8000}$) we see that only 24.6% of our data points have an income >\$50k. A model that achieved 70% accuracy given this data set would not be good. We would do better to just say that every data point earned >\$50k a year. In this case would be accurate 76.4% of the time which is better than our model.
There are 85 different columns in the data set including the binary codes for the one-hot encoding so each data point has 85 dimension.

Q6 Norms and Distance

$$\|x\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2} \tag{26}$$

$$x = x - 0 \tag{27}$$

$$\|x - 0\|_2 = \sqrt{\sum_{i=1}^{d} (x_i - 0)^2} \tag{28}$$

$$\tag{29}$$

The norm of x is the distance between x and the 0 vector. we can now show that the distance between x and a new point z can be written in the same form.

$$\text{let } y = x - z \tag{30}$$

$$\|y\|_2 = \sqrt{\sum_{i=1}^{d} (x_i - z)^2} \tag{31}$$

Q7 & Q8 Code

Code is in this zip file. Some lines need to be uncommented to run properly. I didn't do a good job planning out with psuedo code at the begining so I have some pretty janky parameter passing but everything works and I did my best to comment diligently.

Q9 Accuracy

K	Train on Train Accuracy	4-Fold Accuracy	Variance(4-fold)
1	.987	.785	6.9E-5
3	.890	.809	3.8E-5
5	.870	.817	4.1E-5
7	.861	.820	7.5E-6
9	.855	.823	1.1E-5
99	.833	.828	5.3E-5
999	.823	.821	5.4E-5
8000	.754	.754	3.7E-5

The best number of K-neighbors for the 4-fold validation was the K=99 test. This produced an average accuracy of .827 over all 4 folds. However this was not the best choice of k for the test data according to kaggle.
When k=1 there is still some training error (98.7% accuracy). I think the only way that the training test with k=1 would not be zero is if there are other data points in the dataset with the same x values but have a different y value. That is duplicate data points with different outputs. Otherwise it seems that k=1 would always produce a training error of 0% because it would just always pick itself as the closest point.
As K increases our train on train accuracy starts to decrease. This decrease in accuracy correlates to underfitting the data. The model is taking in data points that could be pretty far away from the one we are trying to test, and therefore not have similar outputs. We see this to an extreme when k=8000. At this point our model is only 75% accurate because it so general.
On the otherside, we see when k is smaller, closer to 1, our model overfits the training data. When k=1 we see almost an 100% accuracy. This is because the model just looks at the closest point, which is likely the point itself in the training data, and chooses the output of that one. We can see that k=1 does really poorly with test data by looking at the accuracy of our 4-fold validation test (only 78% accurate).
Something that was interesting to me was that the hyperparameter of k that I found to be the most accurate in the 4-fold validation test did not perform very well with the actual test data. In the 4-fold validation I found k=99 to be the best performing value for K but when using this hyperparameter with the real test data I only got an accuracy of 79%.

3 Debriefing

1. 15 hrs
2. Moderate
3. Mostly alone
4. 80%, I think this gave really good insight on what is going on under the hood of KNN. However, I do not understand a lot of the statistics still.
5. Really fun project. got me excited for the rest of the term!