Brandon Withington

CS434 HW1 Writeup

April 19, 2021

Part 1 Statistical Estimation

Question 1.)

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

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

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

$$= l(\lambda|x_1, \dots \dots x_n) = \sum_{i=1}^{N} log \left[\frac{\lambda^{x_i} e^{-\lambda}}{x!}\right]$$

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

$$l(\lambda|x_1, \dots \dots x_n) \propto \sum_{i=1}^{N} [-\lambda + x_i \, log\lambda]$$

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

$$-N\lambda + \sum_{i=1}^{N} x_i \, log\lambda$$

$$-N + \frac{\sum x_i}{\lambda}$$

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

$$-N + \frac{\sum x_i}{\lambda} = 0$$

$$\frac{\sum x_i}{\lambda} = N$$

$$\frac{\sum x_i}{N} = \lambda$$

Question 2.) Maximum posteriori estimate of λ with Gamma prior

a.) Write out the log posterior $logP(\lambda|\mathrm{D})$ as proportional to $logP(D|\lambda) + \log\, P(\lambda)$

$$logP(\lambda|\mathrm{D}) \propto \, logP(D|\lambda) + \log\, P(\lambda)$$

b. Take the derivative of $logP(D|\lambda) + \log\, P(\lambda)$ with respect to λ There are a lot of terms that are dropped as they are constants.

$$\frac{d}{d\lambda}[logP(D|\lambda) + \log P(\lambda)]$$

$$\frac{d}{d\lambda}\left[-N\lambda + \log(\lambda)\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \log(x_i!) + (\alpha\log(\beta) + (\alpha-1)\log(\lambda) - \beta\lambda\right]$$

$= \frac{1}{\lambda}(\alpha - 1) - \beta - N + \frac{1}{\lambda}\sum_{i=1}^{N} x_i$

c. Set derivative to zero and solve for lambda

$\frac{1}{\lambda}(\alpha - 1) - \beta - N + \frac{1}{\lambda}\sum_{i=1}^{N} x_i = 0$

$$\lambda_{map} = \frac{(\alpha - 1) + \sum_{i=1}^{N} x_i}{N + \beta}$$

Question 3.)

$$P(\lambda\,|D) \propto \prod_{i=1}^{N} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \qquad \frac{\beta^{\alpha}\lambda^{\alpha} e^{-\beta\lambda}}{\Gamma(\alpha)}$$

$$\frac{\lambda^{\sum x_i} e^{-N\lambda}}{\prod_{i=1}^{N} x_i!} \qquad \frac{\beta^{\alpha}\lambda^{\alpha} e^{-\beta\lambda}}{\Gamma(\alpha)}$$

$$\propto \lambda^{\sum x_i + \lambda - 1} e^{-N\lambda} e^{-(\beta - N)\lambda}$$

$$\lambda^{\sum x_i + \lambda - 1} e^{-N\lambda} e^{-(\beta - N)\lambda} \propto \text{Gamma}(\lambda, \alpha, \beta)$$

PART 2.) K-NEAREST NEIGHBOR

Question 4.) Encodings and Distance

When looking at the binarization of the fields, if we were to increase the ratios of the way we encode integers, it could possibly lend to giving larger weights to differing values, therefore potentially increasing distance from one point to another in a major way. To potentially eliminate nominal values one-hot encoding could be used, which allows us to apply the same flat weight to each value therefore not impacting the way we calculate distance.

Question 5.)

1967 people out of 8000 people have an income that is >$50k which translates into about 24.587% of our data. If we were to implement a model that achieved 70% accuracy I do not believe it would be sufficient in both fully capturing and describing the data. If we were to go with a model based off the stat of how many people make >$50k It would achieve a maximum accuracy of about 75.413% which would be slightly better but not by much. Considering there are 86 different columns and we are only looking at 85; every data point has 85 dimensions.

Question 6.)

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

The normal of x is = x – 0.

Given a new vector z, show that the Euclidean distance between x and z can be written as L_2 norm.

Let B = x – z

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

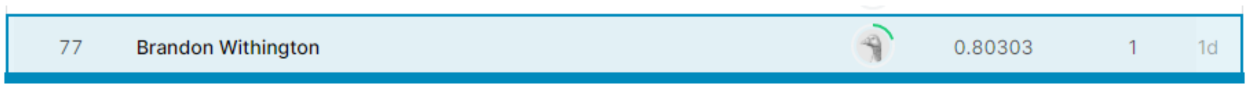
Question 7 & Question 8 are Code in the submitted ZIP file. If needed, the script produces a CSV file as well as a results text file that stores the training on training data accuracy, four fold accuracy, and variance as the number of K neighbors changes.

Q9.)

K-nearest Neighbor #	Training on itself accuracy	Four folds accuracy	Four Folds Variance
1	0.986875	0.7940416666666666	0.0013230254629629638
3	0.89025	0.805375	0.0001933958333333335
5	0.869375	0.8103333333333333	0.0000633518518518138
7	0.863625	0.812125	0.0000470624999999375
9	0.85575	0.8145833333333334	0.0000349166666666156
99	0.832875	0.822125	0.0000003587962962962851
999	0.82525	0.754125	0.000004581018518518487
8000	0.754125	0.754125	0.000004581018518518487

Discussion on results :

From my tests, I believe that the best number of K-neighbors rests between 9 and 99. The highest four folds accuracy rating was achieved with K at 99, however, there does appear to be diminishing returns up to a certain point as the four folds accuracy appears to plumet between 99 and 999. With K set to 99, it hit 82.2125% accuracy over four folds. If we set K to 1 my training error hits around 98.6875% accuracy. I believe it is not 100% because there is potential for duplicate data to reappear in the test sets. I would intuitively believe that the accuracy of the training set on itself should be equivalent to 100% if that were not the case. Having K set to 1 would just let each individual point pick its closest neighbor. Looking at the chart, as we increase the number of neighbors each point looks at we can see the training accuracy decreases gradually over small changes of K. The largest jump is seen between 999 and 8000 K nearest neighbors with its accuracy dipping by roughly 7%. After that, we can see that the jump between 9 and 99 results in a 2% dip in overall accuracy within the training set. One interesting thing about K = 999 neighbors and K = 8000 neighbors is that the four folds variance appears to have been maxed out, or rather minimized. When training accuracy decreases, it corresponds to an underfitting of our data. Our model is looking at data points that are more likely to be further from the point we are at. This is exemplified in the last set, K = 8000, where the training accuracy is 75.4125%, indicating that that model is way too general. If we were to solely go off on training accuracy on itself, K = 1 would allow it to hit around 98.6875% accurate but the four folds accuracy is pretty weak compared to others within the same column. For my Kaggle submission I simply used K=24 mostly because I like the number 24. Here is the result of that :



77	Brandon Withington	0.80303	1	1d

Debrief :

1. 20 hours.
2. Difficult.
3. Alone.
4. 75%. I really liked implementing KNN in code, and creating a method of which my script both produces a CSV file and a result output text file for the four folds accuracy.
5. The first couple questions had me pretty stressed for a long while. I found that the additional bonus office hours for this homework helped me a lot!