John Kaufman

CS 434

Spring 2021

Homework 1

Q1.

1. Log-likelihood logP(D|λ) = $\log(\prod_{i=1}^{N} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}) = \frac{\log(\lambda)\sum_{i=1}^{N}(x_i) - \lambda N}{\log(\prod_{i=1}^{N} x_i!)}$

2. Derivative with respect to λ. Because the denominator is constant with respect to λ, it can be ignored.

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

3. Set the derivative equal to 0 and solve for lambda.

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

$$N\lambda = \sum_{i=1}^{N}(x_i)$$

$$\lambda_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

This makes sense, as λ is the average of the reported instances.

Q2.

1. Log-posterior: logP(λ|D) ∝ logP(D|λ) + logP(λ) = log($\prod_{i=1}^{N} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$) + log(

$$(\beta^{\alpha}\lambda^{(\alpha-1)}e^{(-\beta\lambda)}) \,/\, \Gamma(\alpha))$$

The denominators x! and $\Gamma(\alpha)$ are ignored because they are constant with respect to λ.

$$\propto (log(\lambda)\sum_{i}^{N} x_i \;-\; \lambda N) \;+\; (\alpha\, log(\beta) \;+\; (\alpha \;-\; 1)\, log(\lambda) \;-\; \beta\lambda)$$

2. Take the derivative with respect to λ:

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

3. Set the derivative equal to 0 and solve for λ:

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

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

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

Q3.

P(λ|D) = P(D|λ)P(λ) $= \frac{1}{x!}\lambda^{x}e^{-\lambda} * \frac{1}{\Gamma(\alpha)}\beta^{\alpha}\lambda^{(\alpha-1)}e^{(-\beta\lambda)}$

The denominators x! and $\Gamma(\alpha)$ are again ignored because they are constant with respect to λ.

$$= \lambda^{x}e^{-\lambda} * \beta^{\alpha}\lambda^{(\alpha-1)}e^{(-\beta\lambda)}$$

$$= \lambda^{x} * \beta^{\alpha}\lambda^{(\alpha-1)}e^{(-(\beta+1)\lambda)}$$

$$= \beta^{\alpha}\lambda^{(x+\alpha-1)}e^{(-(\beta+1)\lambda)}$$

Because β^{α} is constant with respect to lambda, we can say

$$\beta^{\alpha}\lambda^{(x+\alpha-1)}e^{(-(\beta+1)\lambda)} \propto \lambda^{(x+\alpha-1)}e^{(-(\beta+1)\lambda)}$$

This is of the same format as a Gamma function of the form:

Gamma($\Lambda = \lambda;\ \alpha + x;\ \beta + 1$)

Therefore, P(λ|D) $\propto$ Gamma($\Lambda = \lambda;\ \alpha + x;\ \beta + 1$)

Q4. With the one-hot encoding technique, the euclidean distance between points will be smaller, because the majority of attributes will share a value of 0. For instance, there are many countries, but a person is only from one country, so for every other country, the value will be 0. Therefore, many points will share a value of 0. If country of origin was treated as ordinal, the values would go from 1 to 40. The distance between two points, one from country 1 and the other from country 40, would be much greater than in the one-hot coding technique, where they would share many points of 0. This method would lead to problems, because there is no good way to assign ordinal values to countries.

Q5. 24.587% of the training data has an income >$50k. A model that said no individual had an income with >$50k would be more than 70% accurate. Therefore, the metric for a good model would be higher, perhaps around 85% accuracy. Each data point has 85 attributes ignoring the identifier and the class label.

Q6. The distance between a vector x and a vector z is $||x-z||_2 = \sqrt{\sum_{i=1}^{d} (x_i - z_i)^2}$,

where d is the number of attributes, and x_i and z_i are the values for those attributes.

Q7. See attached code file. Note: my implementation only works on python3.

Q8. See attached code file. Note: my implementation only works on python3.

Q9.

k value	Training Accuracy	Validation Accuracy (mean, variance)

1	0.986375	mean = 0.78775
		variance = 0.0002320

3	0.89075	mean = 0.808625
		variance = 0.0001485

5	0.8705	mean = 0.816875
		variance = 0.0000943

7	0.862125	mean = 0.818875
		variance = 0.0001553

9	0.855125	mean = 0.822125
		variance = 0.0000637

99	0.832375	mean = 0.8295
		variance = 0.00000124

999	0.82325	mean = 0.8205
		variance = 0.0000398

8000	0.754125	mean = 0.754125
		variance = 0.0000030

For 4-fold cross validation, the best number of neighbors appears to be 99 neighbors. When k = 1, training accuracy is slightly less than 1. This is because in the training set, there are data points where all attributes are the same except for id and classifier. The algorithm will sometimes pick a point with 0 distance, but a different classifier than the point in question. In the training set, increasing the number of neighbors takes the accuracy to the average of the whole set. In 4-fold cross validation, increasing k improves the accuracy up to a certain point, but then it starts decreasing, and accuracy also approaches the average of the whole set. When the number of neighbors is high, the model is underfitted, because it becomes the average of the set for every input. When the number of neighbors is low, the model is overfitted, because it is using the same points for the training set and the testing set.

Q10. On Kaggle, my name is John Campbell Kaufman. I ultimately settled on a regular knn implemented with 999 neighbors evaluated.

Debriefing:

1. I spent approximately 16 hours on this assignment.
2. I would rate this assignment as moderate.
3. I worked on this assignment alone.
4. I feel mostly good about the material it covers. I would say 85% overall understanding, but I am still a little uncertain about Maximum a Posteriori estimation.
5. Implementing knn was a fun challenge for me.