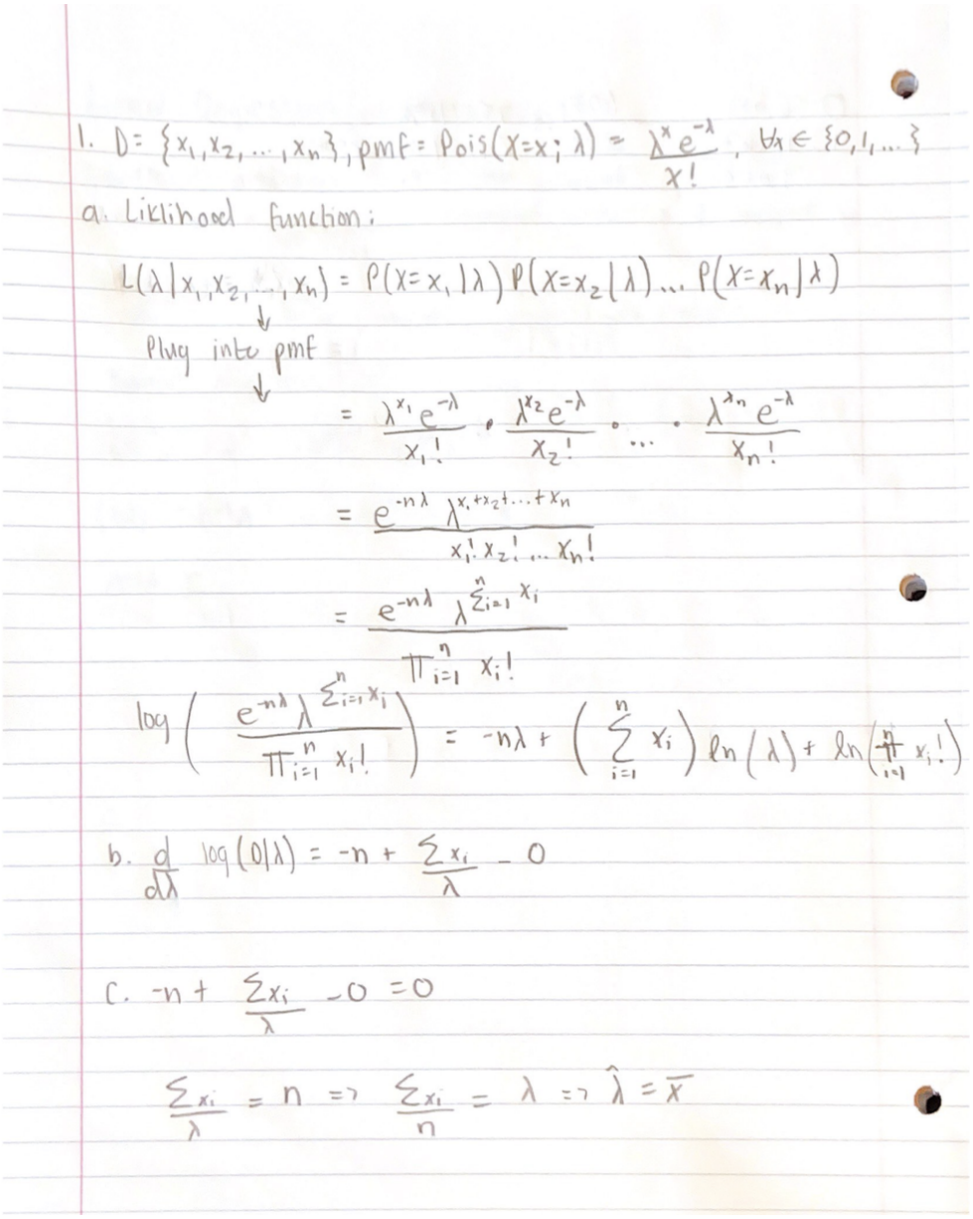
Kevin Pfeil
CS 434
Dr. Lee
Homework #1

1.



The above calculations show I found that $\lambda = \bar{x}$ and this makes sense. This shows that λ gives the average number of occurrences and is the maximizing value, or MLE.

2. Did not have time to get to this question

3.

3. $p(\lambda|D) \propto \text{Gamma}(\lambda; \alpha_p, \beta_p)$

$$f(x_1, x_2, \ldots, x_n|\lambda) = \prod_{i=1}^{n} \left(\frac{e^{-\lambda} \cdot \lambda^{x_i}}{x_i!} \right) = \frac{e^{-n\lambda} \lambda^{\sum_{i=1}^{n} x_i}}{\left(\prod_{i=1}^{n} x_i! \right)}$$

we have the prior $p(\lambda) = \frac{e^{-\beta\lambda} \cdot \lambda^{\alpha-1} \beta^{\alpha}}{\Gamma(\alpha)}, \quad \{\forall \lambda | \lambda > 0\}$

$$p(\lambda|D) = p(\lambda|x_1, x_2, \ldots, x_n)$$

$$= f(x_1, x_2, \ldots, x_n|\lambda) p(\lambda)$$

$$= \int_0^{\infty} f(x_1, x_2, \ldots, x_n|\lambda) p(\lambda) d\lambda$$

$$p(\lambda|D) \propto f(x_1, x_2, \ldots x_n|\lambda) p(\lambda)$$

$$= \frac{e^{-n\lambda} \cdot \lambda^{\sum_{i=1}^{n} x_i}}{\left(\prod_{i=1}^{n} x_i! \right)} \cdot \frac{e^{-\beta\lambda} \cdot \lambda^{\alpha-1} \beta^{\alpha}}{\Gamma(\alpha)} \propto e^{-\lambda(\beta+n)} \cdot \lambda^{\left(\sum_{i=1}^{n} x_i + \alpha - 1 \right)}$$

this shows that $p(\lambda|D) \propto \Gamma(\alpha_p, \beta_p)$ when

$$\alpha_p = \sum_{i=1}^{n} x_i + \alpha - 1 \quad \& \quad \beta_p = \beta + n$$

4.
These two encodings would affect how kNN runs greatly. In our case, we are simply using 0s and 1s to say whether or not this attribute is met. This means that we have many, many columns and each of the examples has many zeros and just a few ones to indicate which of the specific categories it falls into. In the other case, there would be fewer columns and each example would have a value. This value would run until the options for the attribute have been exhausted. This really becomes interesting when dealing with Euclidean distance. Zero and one are points in our *num_dimensions* dimensional space, and in the other encoding system all of the values are also distances that need to be calculated. These larger numbers turn into larger calculations, thus changing how the distances used to calculate kNN would be completed. The other encoding system would also have less dimensions.

5.
$Number\ of\ points\ with\ income\ <\ 50k\ =\ 6,033$
$Number\ of\ points\ with\ income\ >\ 50k\ =\ 1,967$
$\frac{6,033}{8,000}\ =\ 75.41\%,\ \frac{1,967}{8,000}\ =\ 24.59\%$
24.59% of the examples have an income greater than 50 thousand dollars. This could cause the model to lean more in the direction of categorizing the query as income of less than 50 thousand dollars. If we are simply using the majority of the neighbors around the new point to categorize it, then more points that fall into a certain category would logically cause that category to occur more frequently.

I would say that a model that has 70% accuracy is doing fairly well if it has not trained on much data. I would expect that model to get better as it trains on more data. Each data point has 85 dimensions (87 total, subtracting id and class label). Choosing the one-hot encoding strategy causes this number to be so high.

6.

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

let $v = (v_1, v_2, \ldots v_d)^T$ be vector v

distance between v & zero vector

$= \sqrt{(v_1-0)^2 + (v_2-0)^2 + \ldots + (v_d-0)^2}$

$= \|v\|_2 = \sqrt{\sum_{i=1}^{d} v_i^2}$

now, z vector $= (z_1, z_2, \ldots, z_d)^T$

distance between vectors z and v =

$\sqrt{(v_1-z_1)^2 + (v_2-z_2)^2 + \ldots + (v_d-z_d)^2}$

$= \sqrt{\sum_{i=1}^{d} (v_i-z_i)^2}$

$= \|x-z\|_2$

This distance from vector v to z can be written as an L_2 norm.

7. Code attached
8. Code attached

9.

```
Performing 4-fold cross validation
k =     1 -- train acc = 98.66%  val acc = 78.68% (0.0020)            [exe_time = 58.09]
k =     3 -- train acc = 89.01%  val acc = 80.34% (0.0021)            [exe_time = 60.61]
k =     5 -- train acc = 87.06%  val acc = 81.48% (0.0049)            [exe_time = 60.33]
k =     7 -- train acc = 86.28%  val acc = 81.51% (0.0050)            [exe_time = 60.61]
k =     9 -- train acc = 85.47%  val acc = 81.80% (0.0042)            [exe_time = 57.73]
k =    99 -- train acc = 83.25%  val acc = 82.62% (0.0032)            [exe_time = 63.01]
k =   999 -- train acc = 82.31%  val acc = 81.96% (0.0054)            [exe_time = 72.38]
k =  8000 -- train acc = 75.41%  val acc = 75.41% (0.0031)            [exe_time = 153.39]
PS C:\Users\Kevin\Desktop\Coursework\CS434>
```

k = 99 gives me the best validation accuracy of the various k's that are tested. At k = 1, training error is not 0% but it is fairly close at just over 1.3%. The source of training error when k = 1 comes from duplicate points in the data. Each point is its own nearest neighbor, but if there are duplicates then this training accuracy can fall below 100%. The trends that I notice are that the training accuracy goes down as k increases and the validation accuracy goes up and then comes down as k gets very large. This relates to underfitting and overfitting because we see these at both ends of the data. In the smallest case where we simply check the 1 nearest neighbor the data is underfitted and the validation accuracy is comparatively low. At the opposite end we see that checking the 8,000 nearest neighbors sends the validation accuracy back down as a result of overfitting the data.

10.

The only hyperparameter that I messed around with was k, and found that my best k was 99. This was a result of just looking at the various accuracies that were outputted by the program and then trying some other values. I tried k = 150, and this lowered my accuracy by .1%. A similar thing happened when I tried k = 80.

Debriefing

1. I spent approximately 20 hours on this assignment.
2. I would rate this assignment as difficult. Definitely scared for the rest of the term.
3. I spent a lot of time working with others. This was in the form of just being on a call in discord and talking every so often. There was not much in the form of collaboration, more just struggling together.
4. I feel that I deeply understand about 80% of this. If I had more time to understand the homework I would spend it on figuring out the math questions at the beginning of the homework. I know those are not worth as many points, but I feel much more confident in my code than my answers to those questions.
5. It was really interesting and cool, just very hard!