Assignment 2

Delaney Gomen

10/3/2019

1) Provide confusion matrix as an evaluation for your classifier on both datasets.

For the probabilistic generative classifier on the crab test data:

Actual/Predicted	1	0
1	34	0
0	0	32

For the probabilistic generative classifier on the 10d test data:

Actual/Predicted	1	0
1	163	8
0	6	153

2) You should encounter problems when working with the crab dataset when using probabilistic generative model, what is your solution for the problem? Explain why your solution works. (Note: There are more than one solution.)

We encounter a problem with the crab dataset because the last two features (binary, Female, Male) are directly correlated with one another. in linear regression, we would call this issue multicolinearity. What happens here during our model training when two columns are the same is that we begin to have issues with taking the inverse of our matrices. This is because the matrix is not full rank, and when it's full rank it is singular, and when it's singular we can't compute the covariance matrix properly. However, we need this matrix when computing our prior probability to estimate our posterior:

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{1/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1}(\mathbf{x}-\mu_k)\right\} \tag{1}$$

What can we do to solve this? We can simply remove the last column. It's redundant. We know that if a crab has `Female == 0` that that crab is a male, so we do not need an additional column that says `Male == 1` anyway.

3) When training KNN classifier, what happens as you vary k from small to large? Why?

As your k gets larger, you will start having worse results when you implement your trained model on the test data. As k increases, your distance function will have to reach "farther" away from the point that's currently finding its neighbors. That gives it more of a chance to "grab" points that are not in its classification-making the decision boundary more restrictive and making the generalist boundaries less clear. You will begin to overfit because you are just adding more information to the classifier for each point and making it more difficult for the model to generalize new data.

4) For 10d dataset, KNN classifier should give worse performance than probabilistic generative classifier, why is that and how can you improve it?

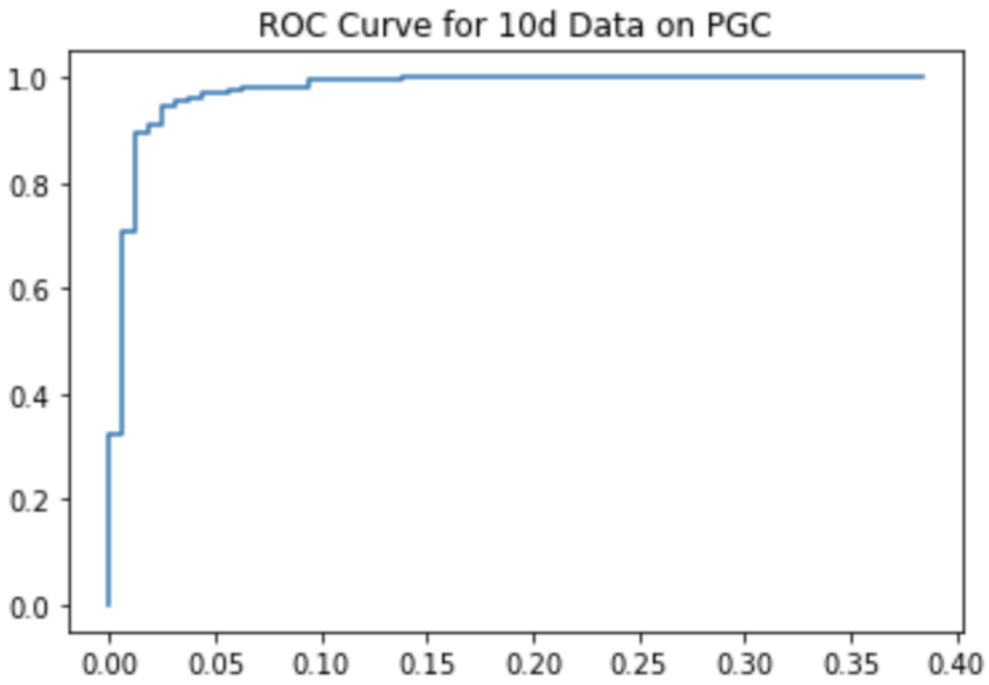
Yes, we were getting around 50% accuracy on the KNN for the 10d data. That is like guessing when you consider that there was about the same number of class 1 as class 2 in the 10d data. We are getting these results because the distance functions we are using-uniform and weighted-are not capturing the relationship between the distribution features. Looking at the scatter plot of the 4th and 5th features, we can see that there is quite a bit of overlap-a KNN Euclidean distance nightmare! We can improve the outcome of our KNN classifier by creating a model that takes advantage of the nature of the features in its distance function. Since the features are random variables, maybe we could create a distance function that utilizies density rather than distance (like in the example in class, using angles instead of distance when spectra alters the underlying result).

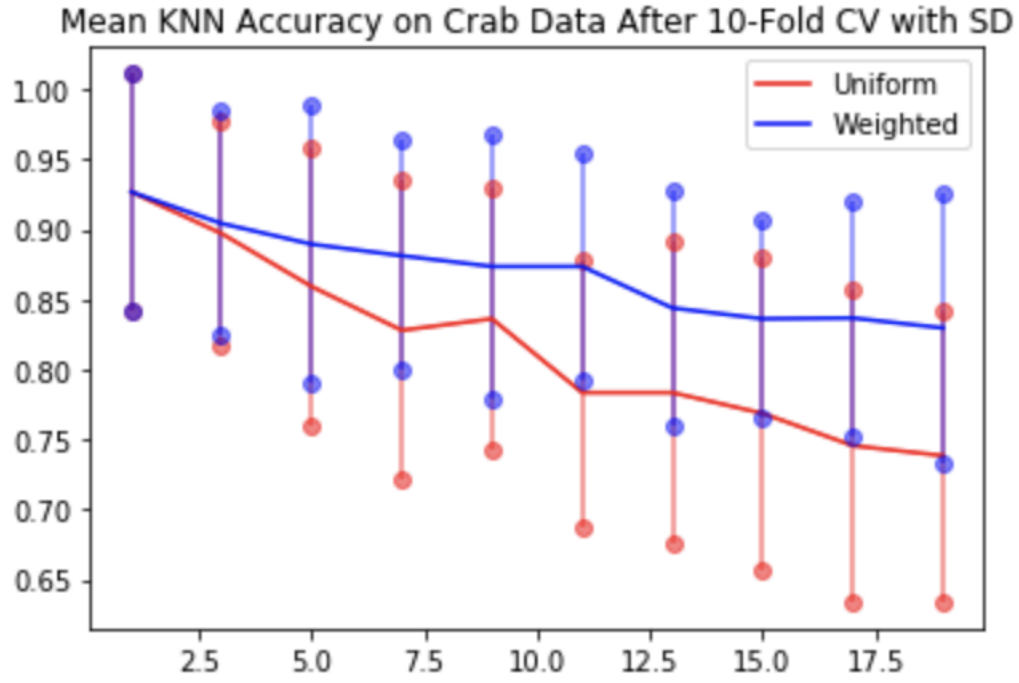
5) Determine which classifier(s) you would use for each data set and give an explanation of your reasoning. Hint: This should incorporate some discussion based on results from cross-validation.

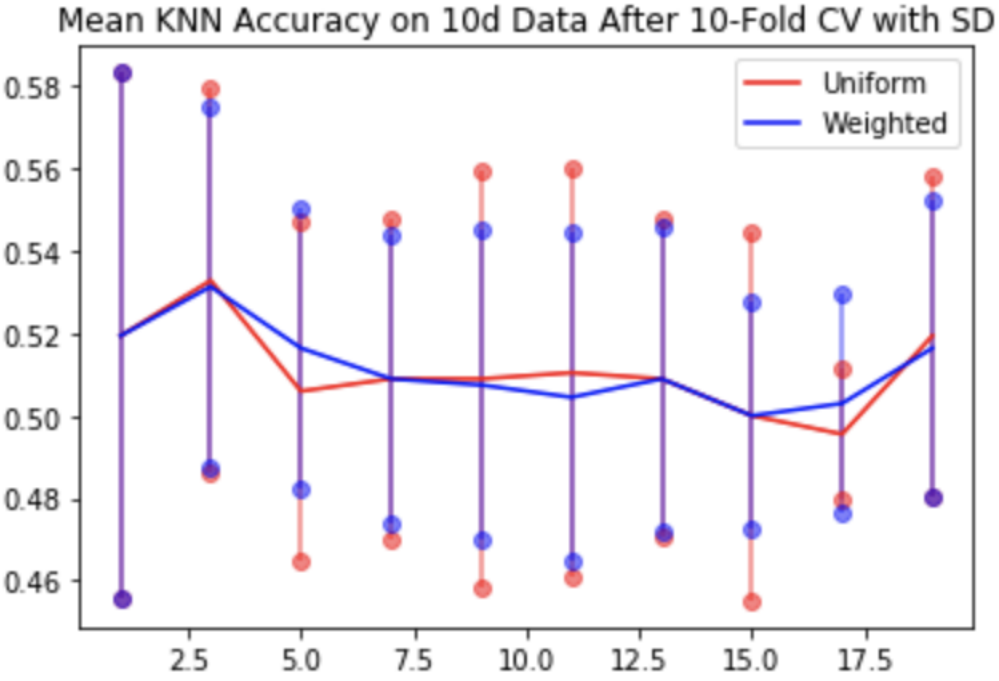
Both datasets performed well under cross-validation using the training data with the probabilistic generative model. I used 10-fold training sets. However, the crab dataset still worries me that it is overfitting despite performing well in cross-validation. The data is very small and leads me to believe that finding neighbors is a better robust mechanism than finding distributions. The KNN classifier at small k performed much better with the crab data than the 10d data, too, which has me believe that the KNN classifier (weighted) could be appropriate for the crab data. However, it's easy to see that the probabilisitic generative classifier performed much better on the 10d data. This makes sense. The central limit theorem is on our side for the training function-we take the mean across the features and the standard deviation. Then we assume that the two classes follow multivariate Gaussian distributions. Each μ_k will be the sample mean of a test statistic, which the probability distribution of a sample mean (no matter the underlying distribution) follows a Gaussian. So, there's clearly some relationship going on here where the probabilistic generation and classification based on random distributions provides an advantage over distance and n-nearest-neighbors.

6) Proofs on last page of PDF.

1. Find the MLE of the mean of a Poisson distribution
2. Find the MAP estimates of the parameters given a Poisson data likelihood and Gamma posterior




Mean KNN Accuracy on Crab Data After 10-Fold CV with SD
Uniform
Weighted
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Mean KNN Accuracy on 10d Data After 10-Fold CV with SD
Uniform
Weighted
0.58
0.56
0.54
0.52
0.50
0.48
0.46
2.5
5.0
7.5
10.0
12.5
15.0
17.5

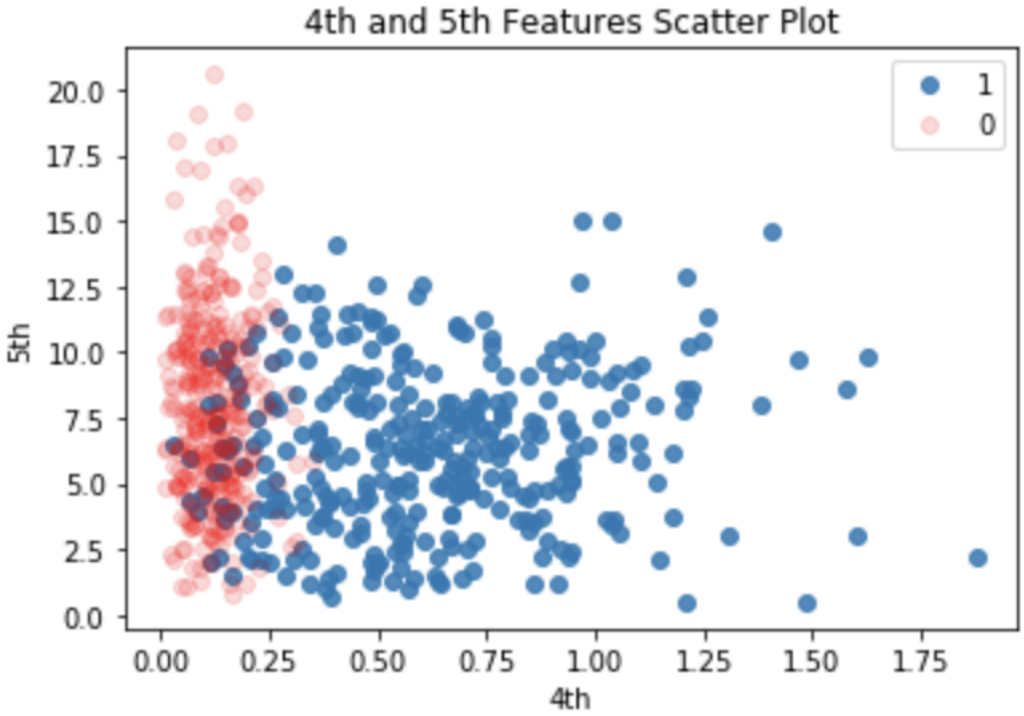


Figure 1: 10d Data

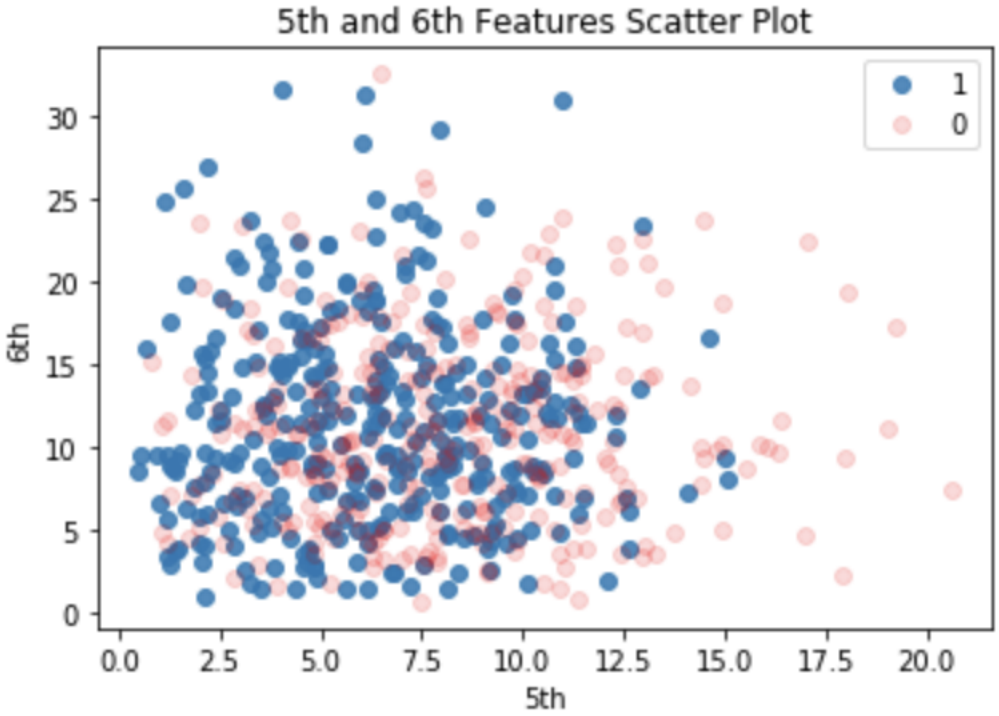


Figure 2: 10d Data

step one : find likelihood function

step two: maximize it w/r/t λ once you take the derivative

step three: solve for λ

1. Find MLE of a Poisson mean (λ)

$$p(\mathbf{p} \mid \lambda) = \prod_{n=1}^{N} p(x_n \mid \lambda)$$

$$= \prod_{n=1}^{N} \frac{\lambda^{x_n} e^{-\lambda}}{x_n!}$$

* use log trick

$$\mathcal{L} = \ln\left(\prod_{n=1}^{N} \frac{\lambda^{x_n} e^{-\lambda}}{x_n!}\right)$$

$$= \sum_{n=1}^{N} \ln(\lambda^{x_n} e^{-\lambda} / x_n!)$$

$$= \sum_{n=1}^{N} \ln(\lambda^{x_n}) - \ln(x!) + \ln(e^{-\lambda})$$

$$= \sum_{n=1}^{N} -\lambda + x_n \ln(\lambda) - \ln(x_n!)$$

$$= -\lambda N + \sum_{n=1}^{N} x_n \ln(\lambda) - \sum_{n=1}^{N} \ln(x_n!)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 = -N + \frac{1}{\lambda} \sum_{n=1}^{N} x_n - 0$$

$$N = \frac{1}{\lambda} \sum_{n=1}^{N} x_n \rightarrow \lambda_{MLE} = \sum_{n=1}^{N} x_n / N$$

2. Find the MAP estimates of the parameters given Poisson likelihood + gamma prior

posterior $\propto$ likelihood $\times$ prior

$$p(\lambda|x) \propto p(x|\lambda)\,p(\lambda)$$

Poisson pdf / likelihood $\prod_{n=1}^{N} \frac{e^{-\lambda}\lambda^{x_n}}{x_n!}$

Where $p(x|\lambda) \sim$ poisson

$p(\lambda) \sim \text{gamma}(\alpha, \beta)$

Gamma prior $\frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}, \quad \lambda > 0$

$$p(\lambda|x) \propto \left[\prod_{n=1}^{N} \frac{e^{-\lambda}\lambda^{x_n}}{x_n!}\right]\left[\frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}\right]$$

$$= \frac{e^{-\sum^{N}\lambda}\,\lambda^{\sum^{N} x_n}}{\left(\prod^{N} x_n!\right)} \cdot \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)}\right) \cdot \lambda^{\alpha-1} e^{-\beta\lambda}$$

$$= e^{-N\lambda} e^{-\beta\lambda}\, \lambda^{\sum^{N} x_n}\, \lambda^{\alpha-1} \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)\prod x_n!}\right)$$

$$= \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)\prod x_n!}\right) e^{-\lambda(N+\beta)}\, \lambda^{\sum^{N} x_n + \alpha - 1}$$

$\text{gamma}(\sum x_n + \alpha,\ \frac{\propto}{N+\beta})$

Now use the log trick to maximize λ

$$\frac{\partial f}{\partial \lambda} = \frac{\partial f}{\partial \lambda}\left(\ln(c) + \ln\left(e^{-\lambda(N+\beta)}\right) + \ln\left(\lambda^{\sum^{N} x_n + \alpha - 1}\right)\right)$$

$$= \frac{\partial f}{\partial \lambda}\left(\ln(c) + (-\lambda)(N+\beta) + \left(\sum^{N} x_n + \alpha - 1\right)\ln(\lambda)\right)$$

$$0 = 0 + (-1)(N+\beta) + \left(\sum^{N} x_n + \alpha - 1\right)\frac{1}{\lambda}$$

$$\lambda_{MAP} = \left(\frac{\sum^{N} x_n + \alpha - 1}{N+\beta}\right)$$