Kyle Robinson
kyle.robinson@uwaterloo.ca
#20540871

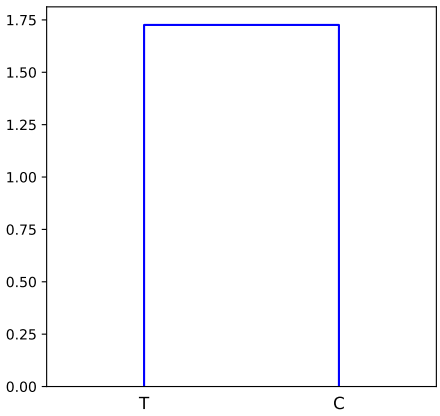
Homework Assignment 4
CS 480/680 Winter 2019

2019-03-21

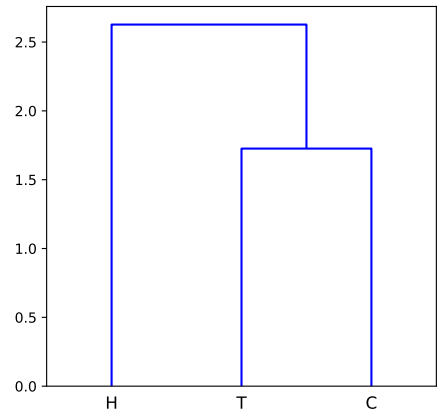
1 K-Means and Hierarchical Clustering

1.1 (a)

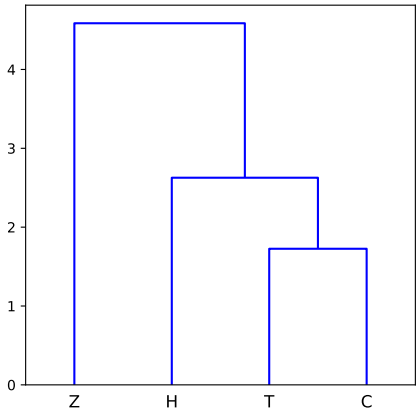
	Z	H	T	L	D	C
Z	0.0	6.934	4.586	9.51	10.543	4.88
H	6.934	0.0	2.627	7.377	4.951	3.11
T	4.586	2.627	0.0	7.375	6.451	1.726
L	9.51	7.377	7.375	0.0	5.076	5.807
D	10.543	4.951	6.451	5.076	0.0	5.707
C	4.88	3.11	1.726	5.807	5.707	0.0



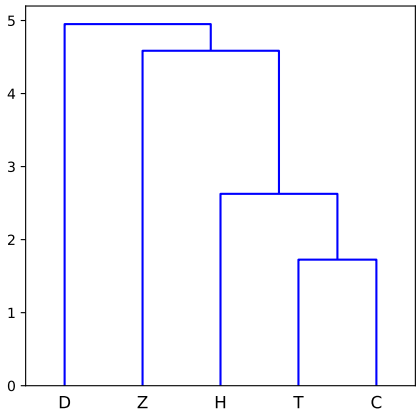
	Z	H	(T,C)	L	D
Z	0.0	6.934	4.586	9.51	10.543
H	6.934	0.0	2.627	7.377	4.951
(T,C)	4.586	2.627	0.0	5.807	5.707
L	9.51	7.377	5.807	0.0	5.076
D	10.543	4.951	5.707	5.076	0.0



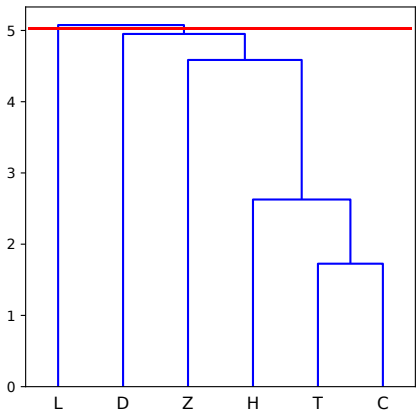
	Z	(T,C,H)	L	D
Z	0.0	4.586	9.51	10.543
(T,C,H)	4.586	0.0	5.807	4.951
L	9.51	5.807	0.0	5.076
D	10.543	4.951	5.076	0.0



	(T,C,H,Z)	L	D
(T,C,H,Z)	0.0	5.807	4.951
L	5.807	0.0	5.076
D	4.951	5.076	0.0



	(T,C,H,Z,D)	L
(T,C,H,Z,D)	0.0	5.076
L	5.076	0.0



The two clusters created consist of (Toronto, Chicago, Hong Kong, Zurich, Delhi) and (Lisbon). The distance between the clusters is 5.076.

1.2 (b)

$$x_1 = 2, x_2 = 4, x_3 = 5, x_4 = 12, x_5 = 18, x_6 = 20$$

1.2.1 (i)

$$c_1 = 0, c_2 = 19$$

(1)

	Distance to c_1	Distance to c_2
x_1	2	17
x_2	4	15
x_3	5	14
x_4	12	7
x_5	18	1
x_6	20	1

$$c_1 = \frac{x_1 + x_2 + x_3}{3} = \frac{2 + 4 + 5}{3} = 3.7$$

$$c_1 = \frac{x_4 + x_5 + x_6}{3} = \frac{12 + 18 + 20}{3} = 16.7$$

$$energy = 1.7^2 + 0.3^2 + 1.3^2 + 4.7^2 + 1.3^2 + 3.3^2 = 39.34$$

(2)

	Distance to c_1	Distance to c_2
x_1	1.7	14.7
x_2	0.3	12.7
x_3	1.3	11.7
x_4	8.3	4.7
x_5	14.3	1.3
x_6	16.3	3.3

$$c_1 = \frac{x_1 + x_2 + x_3}{3} = \frac{2 + 4 + 5}{3} = 3.7$$

$$c_2 = \frac{x_4 + x_5 + x_6}{3} = \frac{12 + 18 + 20}{3} = 16.7$$

$$energy = 1.7^2 + 0.3^2 + 1.3^2 + 4.7^2 + 1.3^2 + 3.3^2 = 39.34$$

The clusters are unchanged from the first iteration and so the algorithm has converged.

1.2.2 (ii)

$$c_1 = 18, c_2 = 19$$

(1)

	Distance to c_1	Distance to c_2
x_1	16	17
x_2	14	15
x_3	13	14
x_4	6	7
x_5	0	1
x_6	2	1

$$c_1 = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{5} = \frac{2 + 4 + 5 + 12 + 18}{5} = 8.2$$

$$c_2 = x_6 = 20$$

$$energy = 6.2^2 + 4.2^2 + 3.2^2 + 3.8^2 + 2^2 + 0 = 84.76$$

(2)

	Distance to c_1	Distance to c_2
x_1	6.2	18
x_2	4.2	16
x_3	3.2	15
x_4	3.8	8
x_5	9.8	2
x_6	11.8	0

$$c_1 = \frac{x_1 + x_2 + x_3 + x_4}{4} = \frac{2 + 4 + 5 + 12}{4} = 5.75$$

$$c_2 = \frac{x_5 + x_6}{2} = \frac{18 + 20}{2} = 19$$

$$energy = 3.75^2 + 1.75^2 + 0.75^2 + 6.25^2 + 1^2 + 1^2 = 58.75$$

(3)

	Distance to c_1	Distance to c_2
x_1	3.75	17
x_2	1.75	15
x_3	0.75	14
x_4	6.25	7
x_5	12.25	1
x_6	14.25	1

$$c_1 = \frac{x_1 + x_2 + x_3 + x_4}{4} = \frac{2 + 4 + 5 + 12}{4} = 5.75$$

$$c_2 = \frac{x_5 + x_6}{2} = \frac{18 + 20}{2} = 19$$

$$energy = 3.75^2 + 1.75^2 + 0.75^2 + 6.25^2 + 1^2 + 1^2 = 58.75$$

The clusters are unchanged from the last iteration, and so the algorithm has converged (to a local minimum). Of the two K-Means solutions, the first is better as it has a lower energy. It also achieved convergence in less iterations than the second solution. This shows a simple example of how the random initialization of cluster points can impact the k-means algorithm.

2 EM Algorithm

2.1 (a)

Let k_1, k_2 be the number of heads observed over $n_1, n - n_1$ coin tosses for C_1, C_2 respectively.
In general:

$$P_p(k) = k \ln(p) + (n - k) \ln(1 - p)$$

$$0 = \frac{d}{dp} P_p(k) = \frac{k}{p} - \frac{n - k}{1 - p} \implies p = \frac{k}{n}$$

For p_1:

$$p_1 = \frac{k_1}{n_1}$$

For p_2:

$$p_2 = \frac{k_2}{n - n_1}$$

2.2 (b)

Let x be the set of observed coin flips, and $\theta = (p_1, p_2, \lambda)$ then the expected log likelihood for the data if you can not tell the coins apart is:

$$\sum_C p_2(C|x, \theta) \log p_1(x, C|\theta)$$

2.3 (c)

Derivations found and used from here.

2.3.1 E-step

Let $\hat{\lambda} = 0.5, \hat{p_1} = 0.7, \hat{p_2} = 0.3$, and $x = [1, 0, 1, 1]$ represent the binary results of a coin toss, then:

$$E_i = P(C_i = 1|x_i, \theta)$$

$$E_i = \frac{\lambda(\hat{p_1})^{x_i}(1-\hat{p_1})^{1-x_i}}{0.5\hat{p_1}^{x_i}(1-\hat{p_1})^{1-x_i} + (1-\lambda)\hat{p_2}^{x_i}(1-\hat{p_2})^{1-x_i}}$$

Substituting values we get:

$$E_1 = \frac{0.5(0.7)^1(1-0.7)^{1-1}}{0.50.7^1(1-0.7)^{1-1} + (1-0.5)0.3^1(1-0.3)^{1-1}} = 0.7$$

$$E_2 = \frac{0.5(0.7)^0(1-0.7)^{1-0}}{0.50.7^0(1-0.7)^{1-0} + (1-0.5)0.3^0(1-0.3)^{1-0}} = 0.3$$

$$E_3 = \frac{0.5(0.7)^1(1-0.7)^{1-1}}{0.50.7^1(1-0.7)^{1-1} + (1-0.5)0.3^1(1-0.3)^{1-1}} = 0.7$$

$$E_4 = \frac{0.5(0.7)^1(1-0.7)^{1-1}}{0.50.7^1(1-0.7)^{1-1} + (1-0.5)0.3^1(1-0.3)^{1-1}} = 0.7$$

2.3.2 M-step

$$\lambda^{next} = \frac{1}{n}\sum_i E_i$$

$$\hat{p_1}^{next} = \frac{\sum_i E_i x_i}{\sum_i E_i}$$

$$\hat{p_2}^{next} = \frac{\sum_i (1-E_i) x_i}{\sum_i (1-E_i)}$$

Substituting values we get:

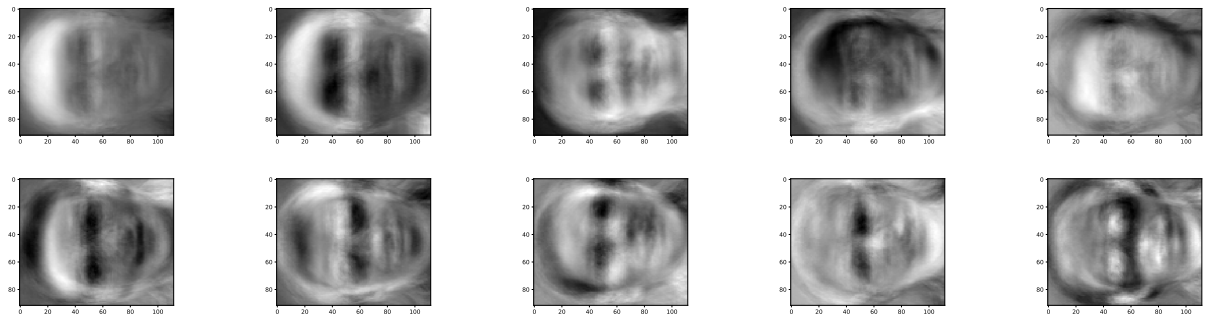
$$\lambda^{next} = \frac{1}{4}(0.7 + 0.3 + 0.7 + 0.7) = 0.6$$

$$\hat{p_1}^{next} = \frac{(0.7*1) + (0.3*0) + (0.7*1) + (0.7*1)}{0.7 + 0.3 + 0.7 + 0.7} = 0.875$$

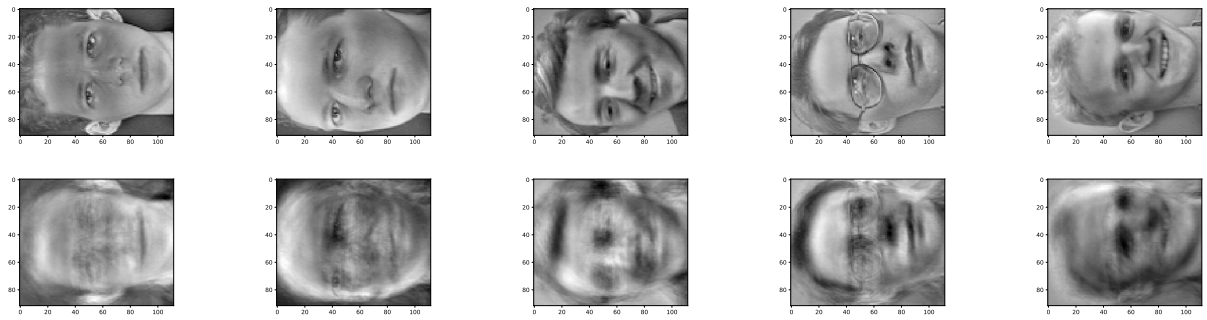
$$\hat{p_2}^{next} = \frac{1(1-0.7) + 0(1-0.3) + 1(1-0.7) + 1(1-0.7)}{(1-0.7) + (1-0.3) + (1-0.7) + (1-0.7)} = 0.5625$$

3 PCA and Eigenfaces

3.1 (a)



3.2 (b)

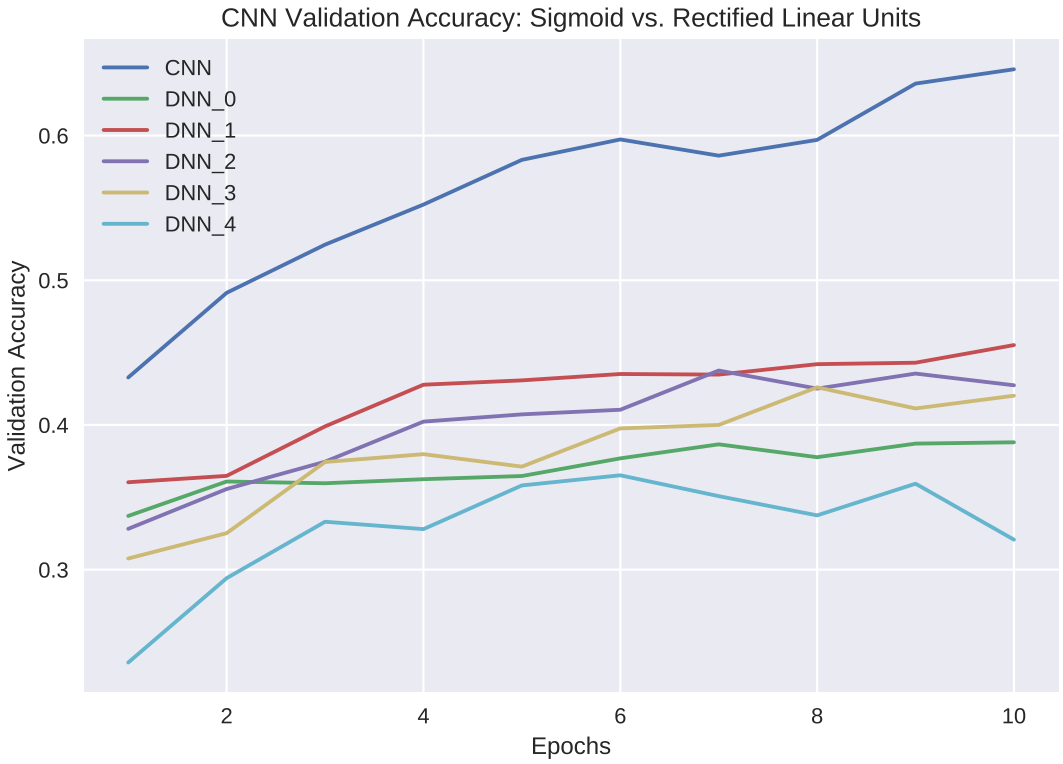


3.3 (c)

Data was loaded into a Pandas dataframe within a Jupyter Notebook Python 3 environment which can be accessed here. Using a NearestNeighborClassifier from the SKLearn package with n_neighbors set to 1, a classification accuracy of 0.95 was achieved on the test data.

4 Neural Network

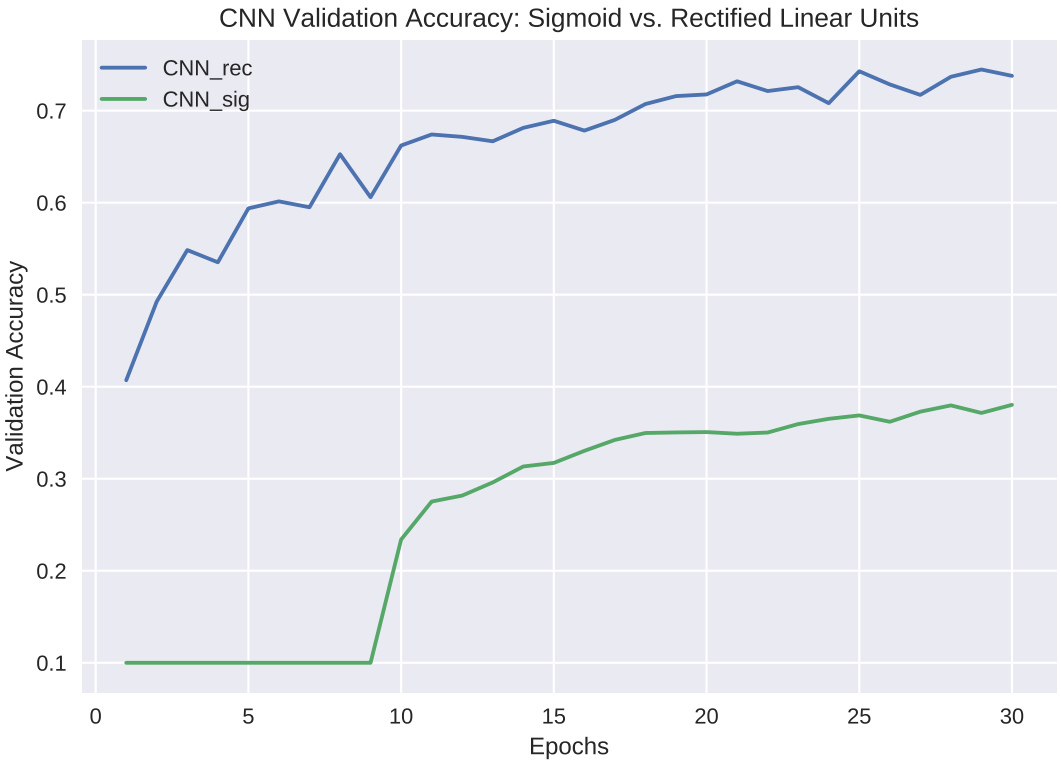
4.0.1 (a)



The dense neural networks were implemented using one Flatten input layer, and one Dense softmax output layer. Dense layers were included in between using rectified linear units and a dropout value of 0.5.
The figure above displays two trends. First, the CNN developed for this image classification task preforms (unsurprisingly) very well, while the Neural Networks comprised of only Dense hidden layers performed substantially worse. This result was expected, and can be explained by the ability for hidden convolutions and pooling layers to pick up on more abstract patterns within images through the use of many convolutions and semi-connected layers which are eventually fed into a fully connected (dense) layer before final classification. The second trend which may have become more apparent if deeper neural networks were tested was the drop off in accuracy with deep and dense networks. The worst performing network was the one consisting of 4 dense layers. More testing with deeper networks and longer convolutions would be needed in order see if the

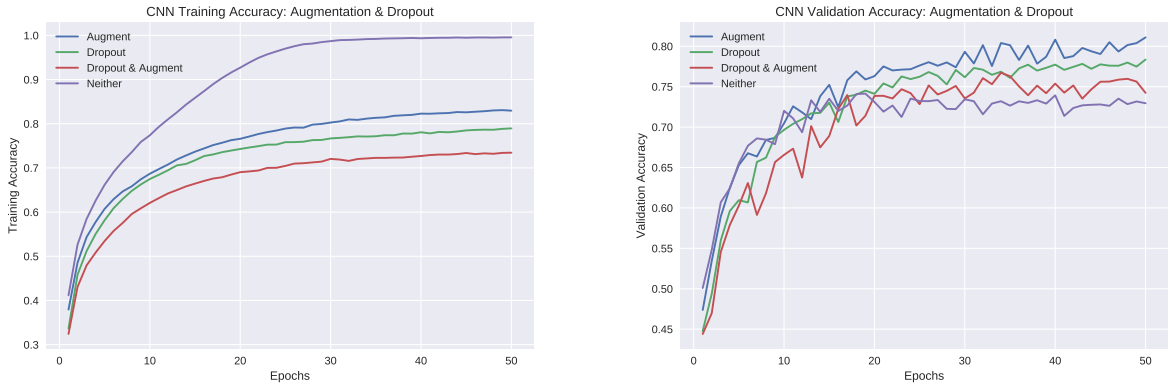
trend continues. A couple possible reasons for this result could be overfitting to the training data (unlikely over just 10 convolutions), underfitting due to decay in the back-propagation through the greater depth, or simply not giving it enough time to approach convergence.

4.0.2 (b)



The results from this question can be seen in the above graph, and the results speak for themselves. The Neural Network trained using sigmoid units preformed abysmally, whereas the one using rectified linear units preformed very well. More interestingly, until around the 9th epoch the sigmoid model remained fixed at an accuracy rate of 0.1. This is likely due to the ability of the RELU activation function to reduce the effect of vanishing gradient which is especially problematic in deeper neural networks with more hidden layers. Another reason is the RELU activation function's sparse output, where any value less than or equal to 0 is set to 0. The sigmoid function is more likely to produce a non-zero value due to it's inherent shape.

4.0.3 (c)



The results from the first 50 epochs of the test can be seen above. Results were generated earlier to 100 epochs, however were lost. The results past 50 epochs mirror the trends sen here. This trend may not continue past 100 epochs.
For training accuracy, the model with neither dropout nor augmentation preformed the best by a substantial margin, whereas the other three configurations preformed progressively worse. The worst performer on the

training data was the configuration with both dropout and augmentation. It is clear, especially when comparing against the testing acuracy, that the configuration with neither dropout nor augmentation overfit to the training data.
The testing acuracy for all models was relatively high, however the configuration with neither dropout nor augmentation preformed worse (due to overfitting). For this number of epochs, it is interesting that augmentation had a positive effect on testing accuracy as we did not even come close to using the entire training data set. Dropout has clear advantages as it essentially mimics ensemble methods by combining many smaller not fully connected networks into one. It would be interesting to see if, and where these results change. I would expect for the combination of augmentation and dropout to preform best if trained for a very long time, as it would provide more data in general, and prevent over fitting on the data which differs only slightly.