CSCI 391 PA2 Write Up

Case ID: qm12

1) A k-means class is created, in which there ar ethods to simulate the k_means clustering algorithm

k_means(k): k - number of clusters

Fields: x: data set

n_samples: # of samples

f_features: # of features

k_clusters: # of clusters

r: (dim nxk): assign parameter arr

mu (dim kxf): coordinates of centroids

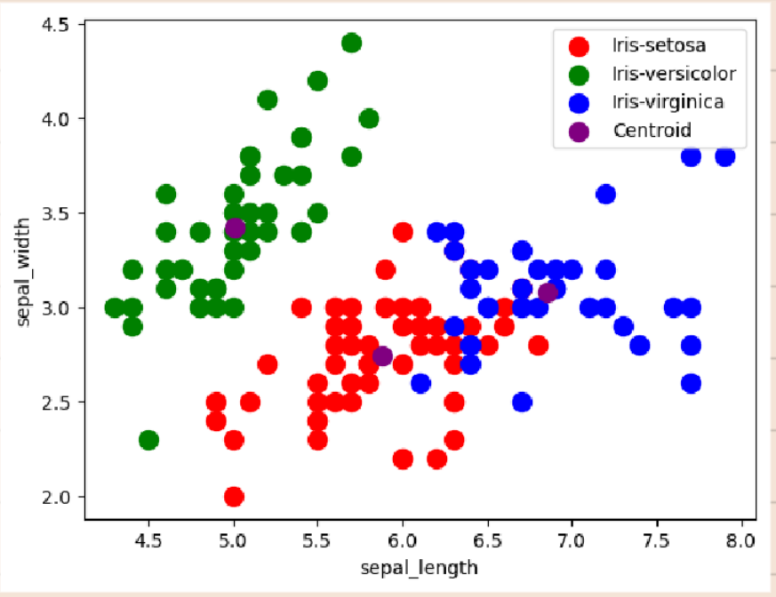
distortion_arr: store changes in Distortion values

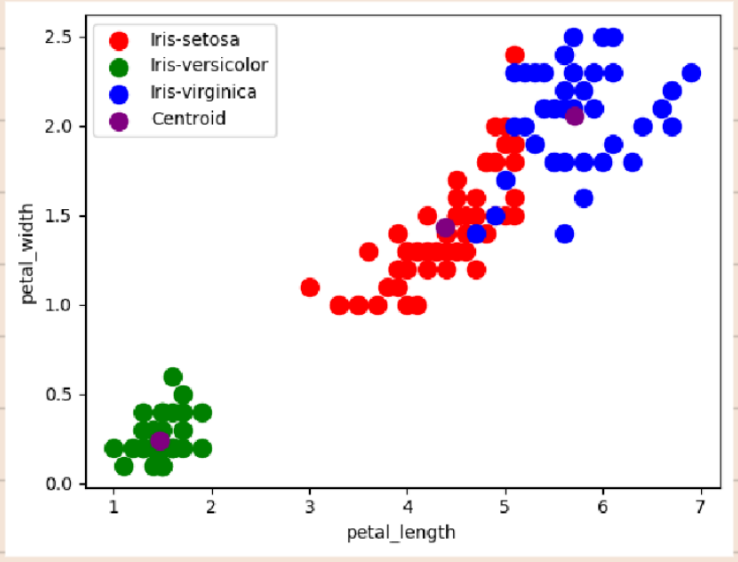
mu_arr: changes in centroids assignment

1a) plot_clustering()

2 graphs: feature 1 and feature 0

feature 3 and feature 2

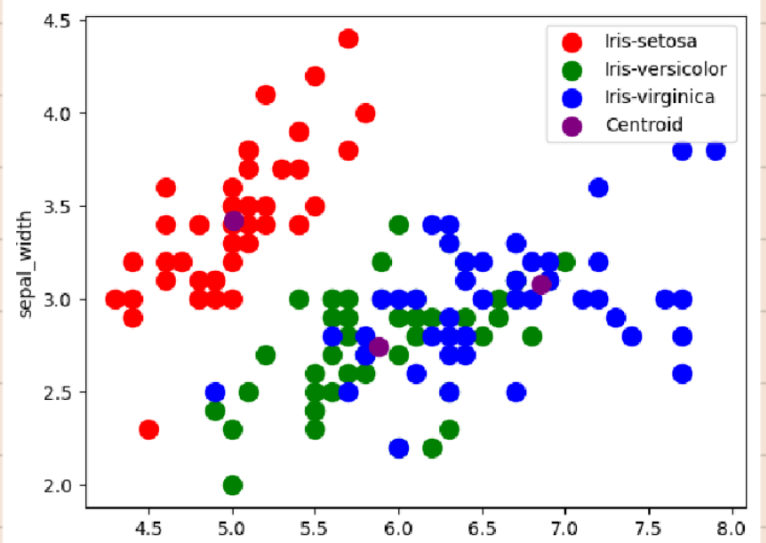


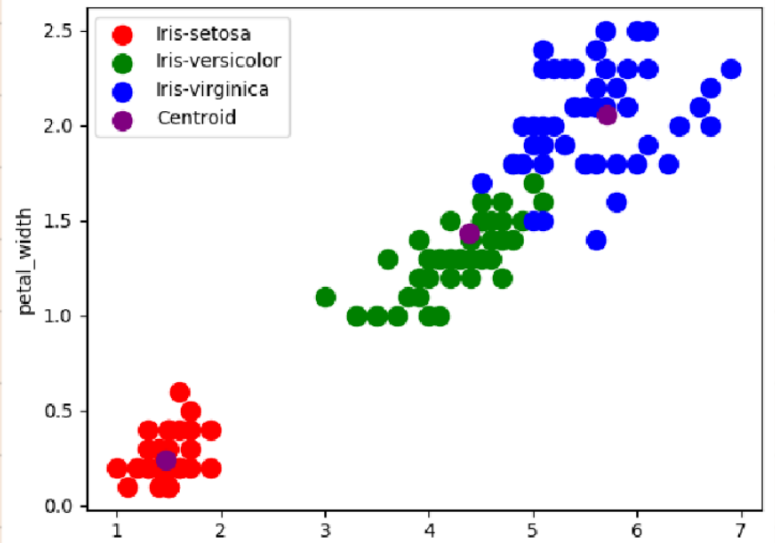


plot_centroids_on_iris_clustering()

plot the cluster centers found by k_means on the original classification.

(read explanation in code)

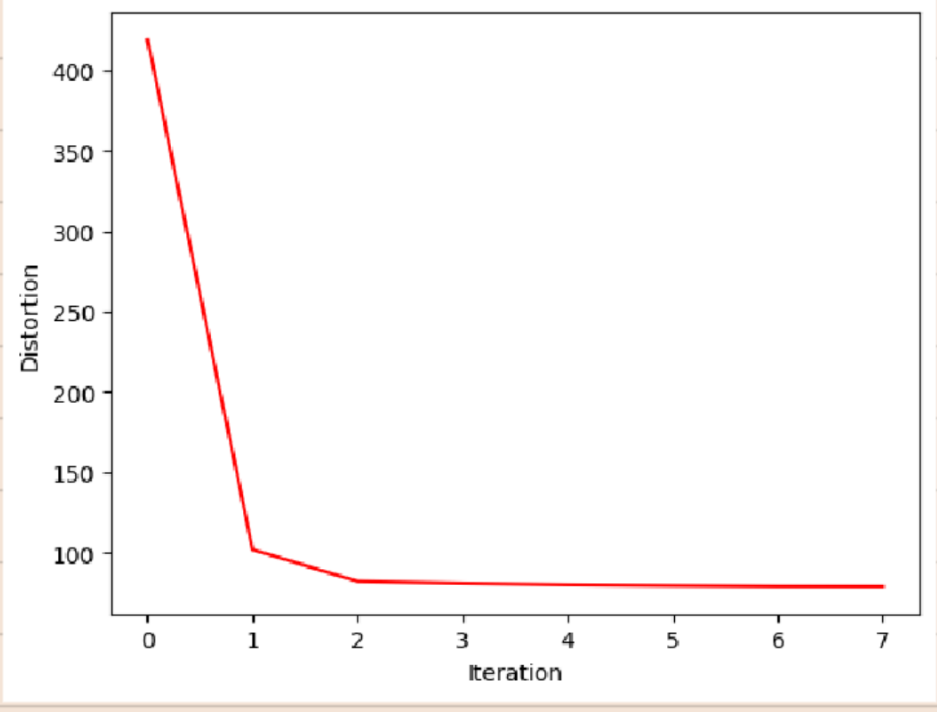




sepal_length

petal_length

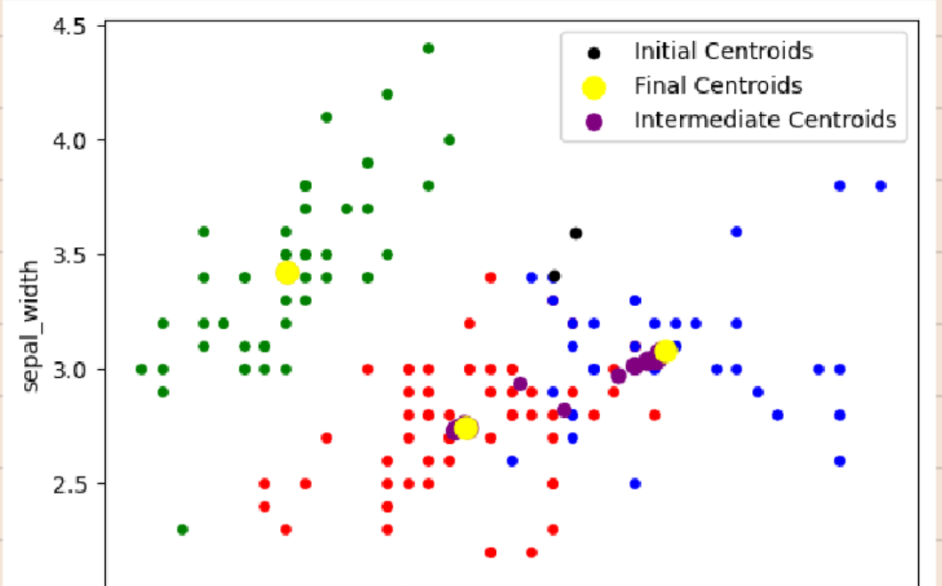
1b) plot_distortion(): using the distortion_arr field from k_means



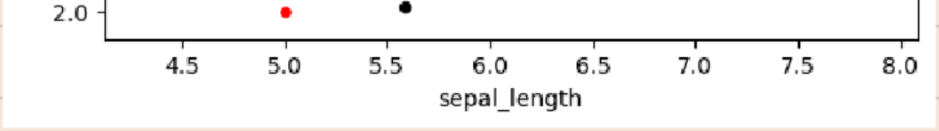
1c) plot_learning_process()

k = 3

use mu_arr field from k_means

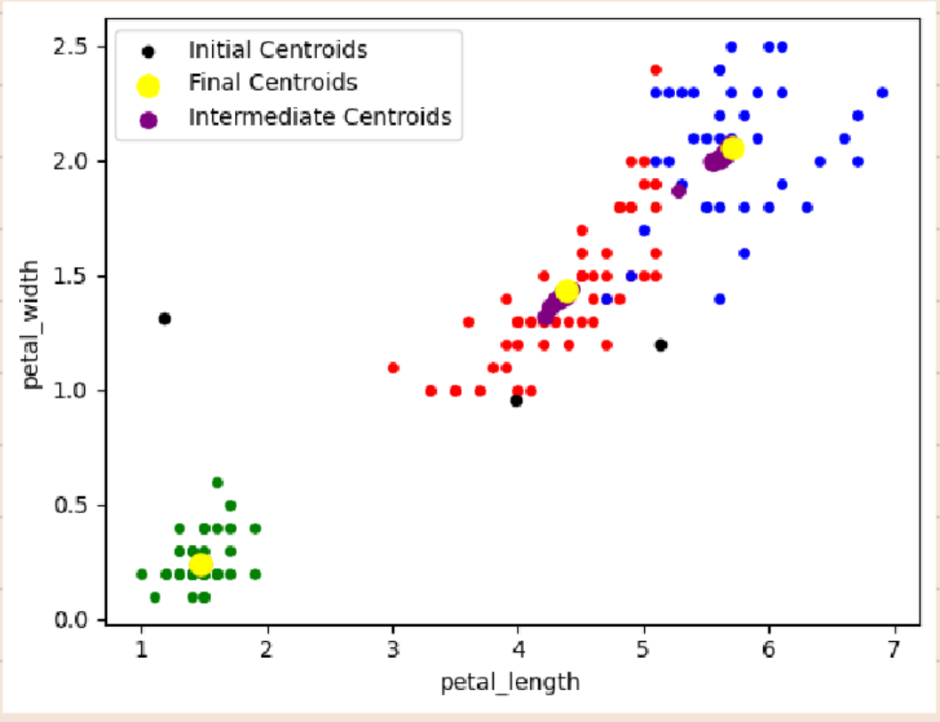


Initial centroids are black, smallest



smallest

k = 2



Final centroids are yellow, biggest centroid's size get bigger after each iteration

1c) Method to graph decision boundary is: Create a grid, then graph each point in the grid according to the cluster centroid that is closest to it.

plot_decision_boundaries(feature_x, feature_y)

↓

integer location of feature

(Graph at the end) column

2) class. single_neuron_NN():
represent a one neuron neural network.

2a) compute_mean_squared(data, parameters, pattern_classes):
- pass the arguments into a single_neuron_NN object, then call calculate_mse()
b) Attempted to use method from 1d to graph decision boundaries, but ran into a problem.
No matter how I initialize the weights (all zeros or random),

the error remains high and the outputs all seem to be $> \frac{1}{2}$ or $\leq \frac{1}{2}$ (so effectively not helpful in classifying)

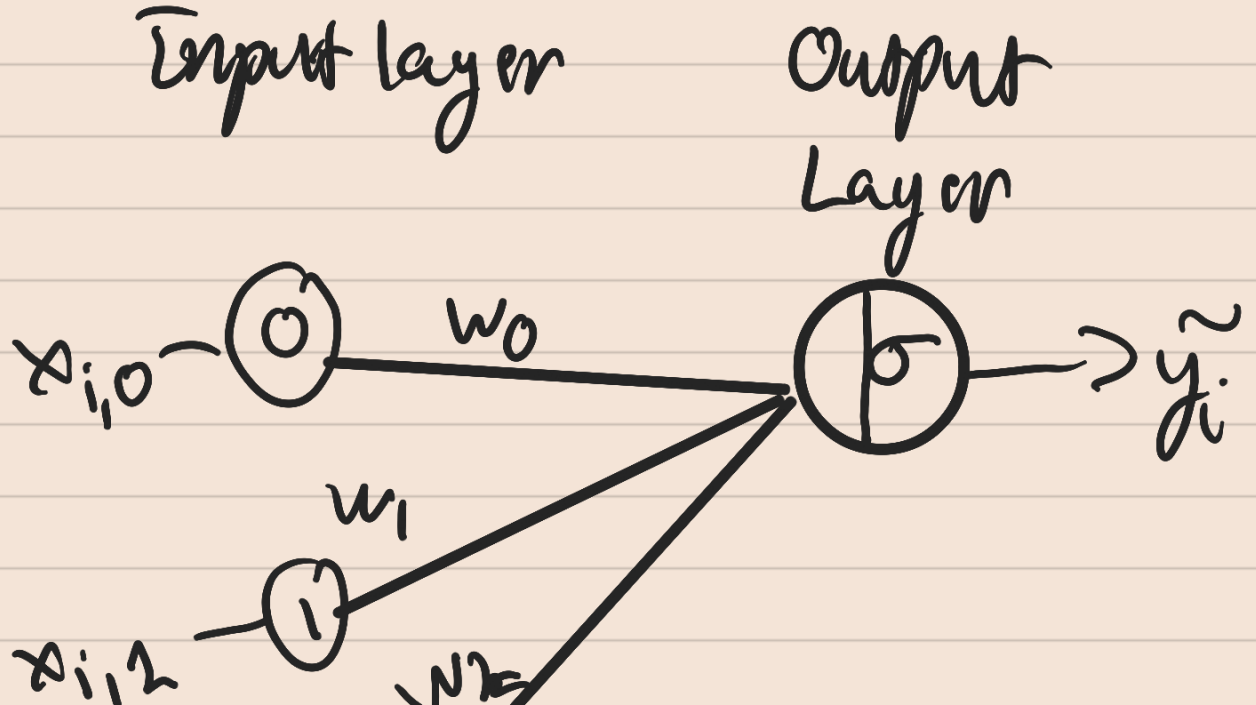
c) Objective function:

$$D = \sum_{i=1}^{N} (y_i - \tilde{y_i})^2$$

$$\tilde{y_i} = \sigma\left(\sum_{j=0}^{F} w_j \cdot x_{i,j}\right)$$

(w_0 is bias, $x_{i,j} = 1 \; \forall i$)

Neural network:



$x_{i,2}$ — O_2 /

$$\tilde{y_i} = \frac{1}{1+e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$

For each ith sample:

$$\frac{dL_i}{dw_j} = \frac{d(y-\tilde{y}_i)^2}{dw_j}$$

$$= \frac{d(y_i-\tilde{y_i})^2}{d\tilde{y_i}} \cdot \frac{d\tilde{y_i}}{dw_j} \quad (1^{st} \text{ chain rule})$$

$$= 2(y_i-\tilde{y_i}).(-1).\frac{d\tilde{y_i}}{dw_j} \quad (1)$$

$$g_i = \sum_{j=0}^{F} x_{i,j} \cdot w_j$$

$$\Rightarrow \frac{d\tilde{y_i}}{dg_i} = \frac{d\left(\frac{1}{1+e^{-g_i}}\right)}{dg_i}$$

$$= \frac{-1}{(1+e^{-g_i})^2}(-e^{-g_i}) \text{ (chain rule)}$$

$$= \frac{e^{-g_i}}{(1+e^{-g_i})^2} = \frac{1}{1+e^{-g_i}}\left(1 - \frac{1}{1+e^{-g_i}}\right)$$

$$= \tilde{y}_i \cdot (1 - \tilde{y}_i)$$

Continue
From (1):

$$\frac{dL_i}{dw_j} = -2(y - \tilde{y}_i) \cdot \frac{d\tilde{y}_i}{dg_i} \cdot \frac{dg_i}{w_j}$$

$$= -2(y - \tilde{y}_i) \cdot \tilde{y}_i \cdot (1 - \tilde{y}_i) \cdot \frac{d(w_0 x_{i,0} + w_1 x_{i,1} + w_2 x_{i,2})}{dw_j}$$

$$= -\underbrace{2(y - \tilde{y}_i) \cdot \tilde{y}_i \cdot (1 - \tilde{y}_i) \cdot x_{i,j}}_{N}$$

Scalar Form

$$\frac{dL}{dw_j} = \sum_{i=1}^{N} \frac{dL_i}{dw_j}$$

d) Vector Form:

#: element-wise multiplication

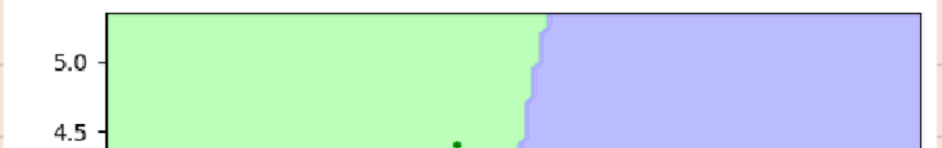
$$\nabla_w L$$

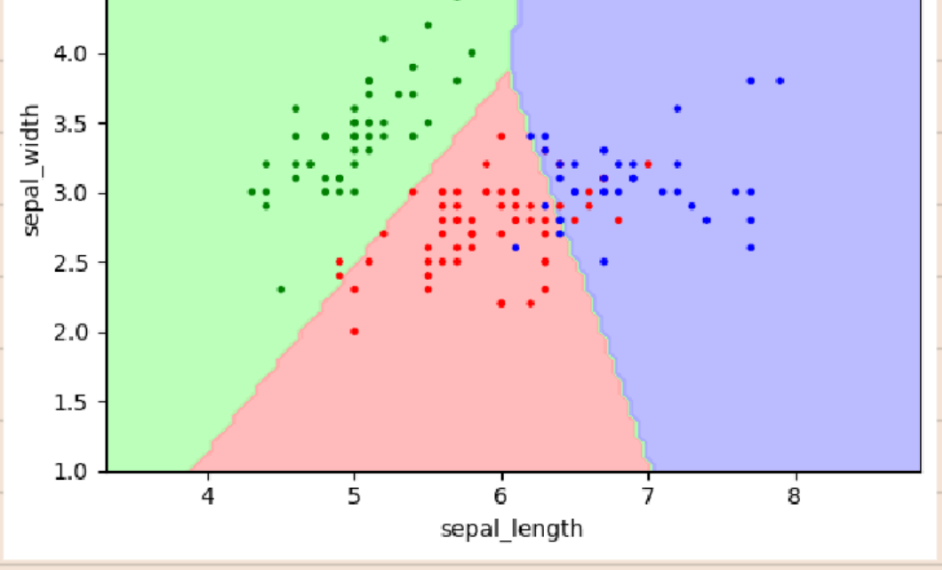
$= (-2)\left[(\vec{y} - \tilde{y}) \# \tilde{y} \# (1 - \tilde{y})\right]$

$\vec{x}$

dot product

e) The code is in method back_propagation() in class single_neuron_NN, which implement the vector-form formula in part d. However, when I run the train() method the loss function seems to be bouncing back and forth, and it doesn't really converge (i.e: weights always fluctuate)




sepal_width
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8
sepal_length