Clustering

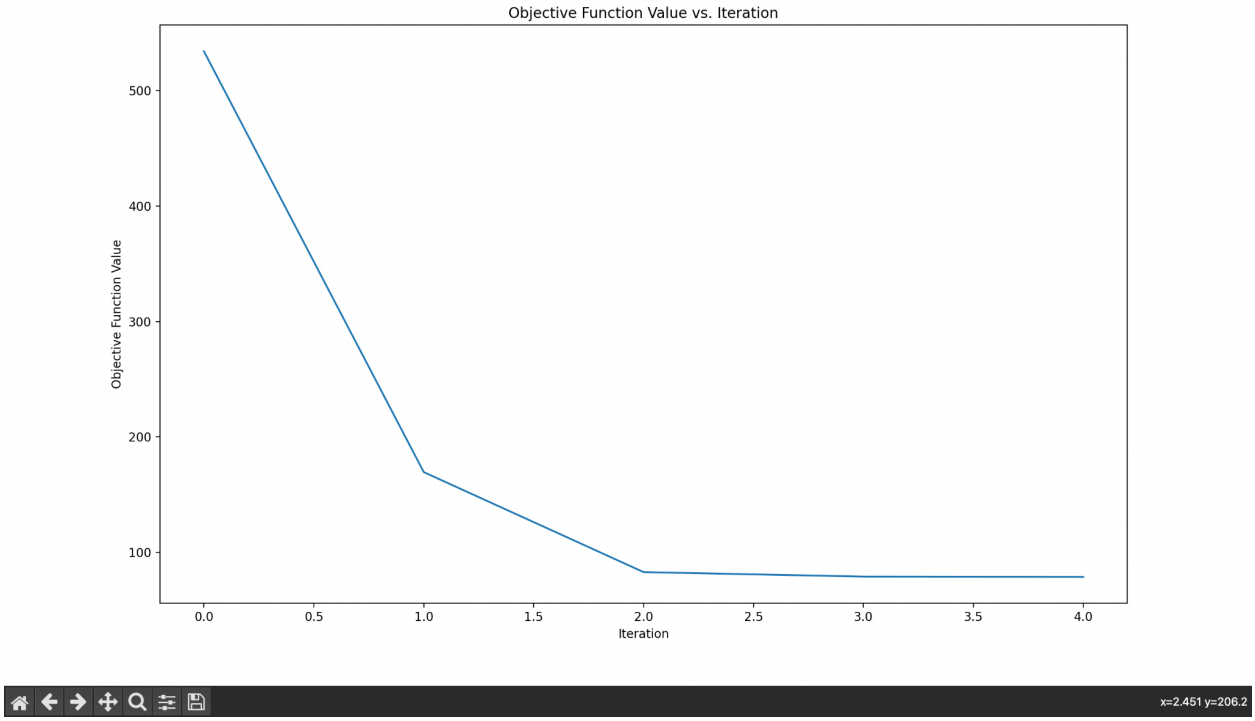
a) Write a program that implements the k-means clustering algorithm on the iris data set. You should use the objective function and learning rule that you will derive in W4 Q1 (and your implementation will help you uncover any errors, so it is helpful to do this step before you turn in W4.) (15 marks)

We import necessary libraries (numpy, matplotlib, scikit learn). The function initializes centroids as mu in random fashion and assigns each point to its centroid using the euclidean distance formula. `mu = data[np.random.choice(N, K, replace=False)]`

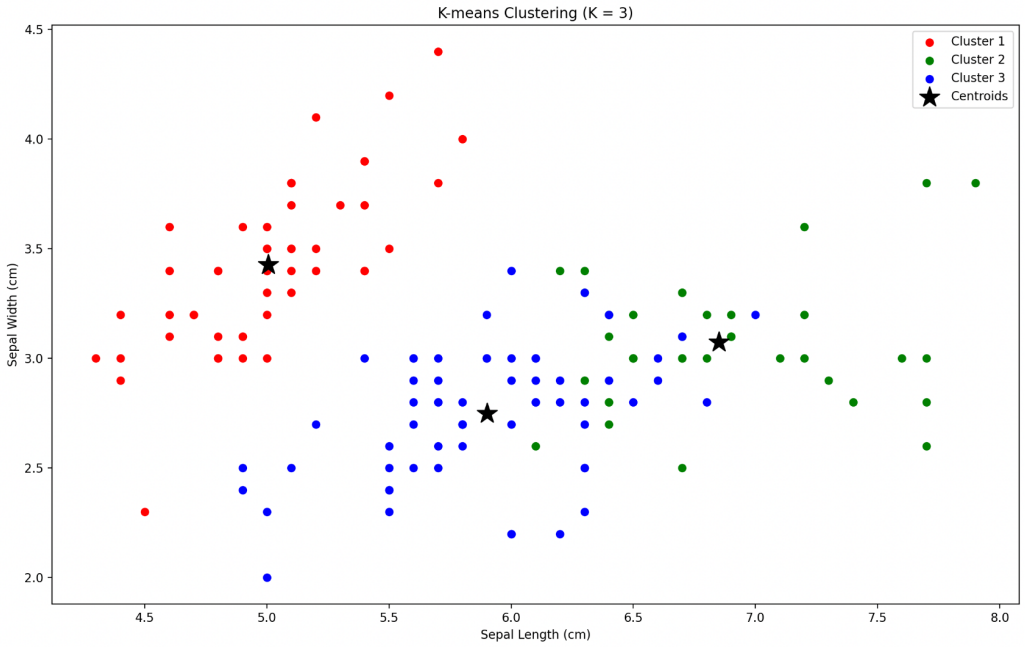
The function updates the centroids by calculating means for all data points on each centroid. All centroids are either converged or the iterations are maxed out. Mu is then returned and data points are assigned to their clusters.

```
for iter in range(maxiter):
        for n in range(N):
            distances = np.sum((data[n, :] - mu)**2, axis=1)
            k = np.argmin(distances)
            r[n, :] = 0
            r[n, k] = 1

        munew = np.zeros((K, D))
        for k in range(K):
            munew[k, :] = np.mean(data[r[:, k] == 1, :], axis=0)

        objectivefuncvalue = np.sum(
            [np.sum((data[r[:, k] == 1, :] - mu[k, :])**2) for k in range(K)])
        objective_func_values.append(objectivefuncvalue)

        if np.allclose(mu, munew, atol=1e-4):
            break

        mu = munew.copy()
```

b) Plot the value of the objective function as a function of the iteration $D = X_{N\ n=1} X_{K\ k=1} r_{n,k} || x_n - \mu_k ||_2$ *to show that your learning rule and implementation minimizes this expression. (5 marks)*


Objective Function Value vs. Iteration
Objective Function Value
Iteration
500
400
300
200
100
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x=2.451 y=206.2

c) Plot the results of the learning process by showing the initial, intermediate, and converged cluster centers overlaid on the data for k = 2 and k = 3. (5 marks)



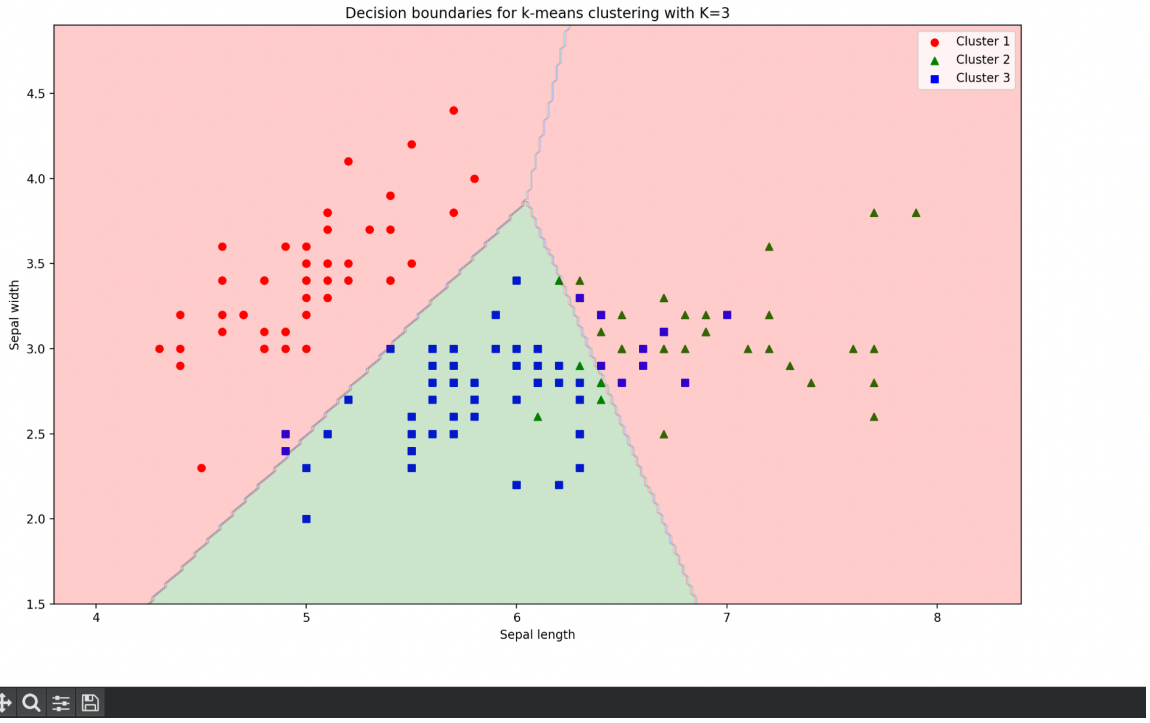
d) Devise a method to plot the decision boundaries for this dataset using the optimized parameters. Explain your approach and plot your results. (10 marks)

In the function, I used a meshgrid to create a decision boundary for all the clusters generated from the kmeansclustering function. The grid contains points spanning the entire range of values for sepal length and sepal width. They are separated with step size 0.02 `h = 0.02` *in a 2d array. Each point's distance to the cluster centers is found with the np.linalg.norm function.* `dist = np.linalg.norm(x - mu[:, :2], axis=1)`

The index of the center with minimum distance assigns a cluster label to the respective point.

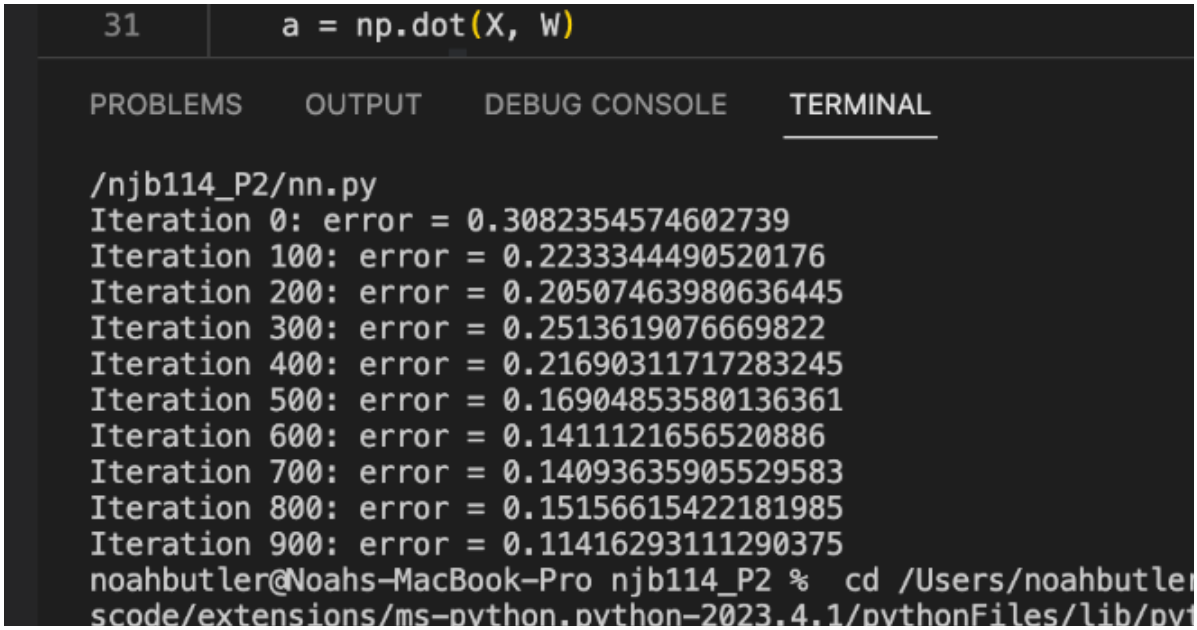
```
plt.scatter(data[r[:, k] == 1, 0], data[r[:, k] == 1, 1],
            c=colors[k], marker=markers[k], label=f'Cluster {k + 1}')
```

All this information is stored in the Z array. The data is plotted with color coding for each cluster and showing the correct contour lines.

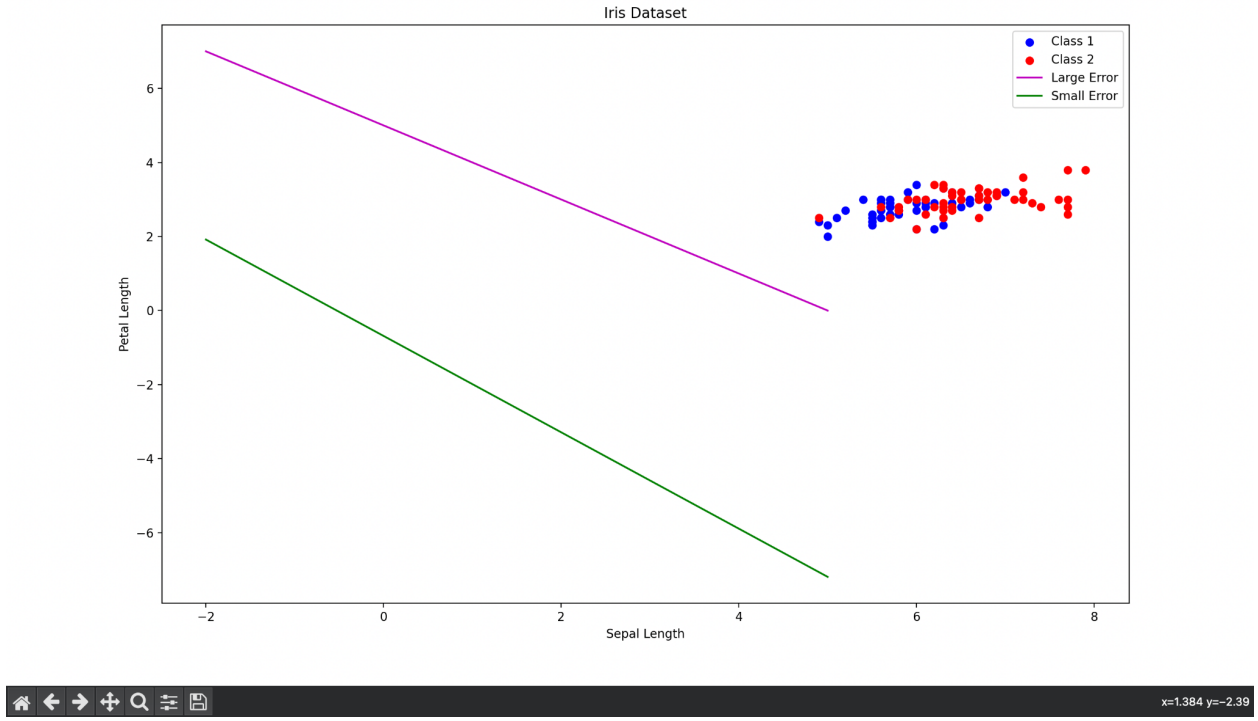


Neural networks

a) Write a program that calculates the mean-squared error of the iris data for simple one-layer neural network using a sigmoid non-linearity (linear classification with sigmoid non-linearity). The function should take three arguments: the data vectors, the parameters defining the neural network, and the pattern classes. (10 marks)



b) Compute the mean squared error for two different settings of the weights (i.e. two different decision boundaries). Select these by hand and choose settings that give large and small errors respectively. Plot both boundaries on the dataset. You will only use the 2nd and 3rd iris classes in this problem. (5 marks)



c) Give a mathematical derivation of the gradient of the objective function above with respect to the neural network weights. You will have to use chain rule and the derivative of the sigmoid function as discussed in class. Use w_0 to represent the bias term. You should show and explain each step. (10 marks)

c) Objective function:

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}\left[y^i \log(\hat{y}^i) + (1-y^i)\log(1-\hat{y}^i)\right] + \frac{\lambda}{2}\sum_{j=1}^{D} w_j^2$$

N is # of data points & D is # of features

y^i is the label for ith data point & $\hat{y}^i$ is the predicted probability for it.

w is vector weights & λ is the reg. parameter.

Predicted prob: $\hat{y}^i = \sigma(w^T x^i)$

* chain rule, derive gradient

$$\frac{\partial J(w)}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}(y^i - \hat{y}^i)\, x_j^i + \lambda w_j$$

→ gradient derived of objective function w/ respect to weights

Derivative of sigmoid function:

$$\sigma(z)(1-\sigma(z)) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

Substitute $\hat{y}^i = \sigma(w^T x^i)$

$$\frac{\partial J(w)}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}\left(y^i - \sigma(w^T x^i)\right) x_j^i + \lambda w_j$$

d) Show how the gradient can be written in both scalar and vector form. (5 marks)

d) scalar form:

$$\frac{\partial J}{\partial w_i} = (y - \hat{y}) * f'(z) * x_i$$

y is true output

$\hat{y}$ is predicted output

$f'(z)$ is derivative of activation function

x_i is input to ith neuron

Vector form

$$\nabla J(w) = (y - \hat{y}) \cdot f'(z) * X \quad (= \underbrace{\sigma}_{\text{error term}} \cdot X)$$

$\nabla J(w)$ is gradient vector

y & $\hat{y}$ are true & predicted outputs

$f'(z)$ is derivative of activation functions

X is input vector

e) Write code that computes the summed gradient for an ensemble of patterns. Illustrate the gradient by showing (i.e. plotting) how the decision boundary changes for a small step. (10 marks)