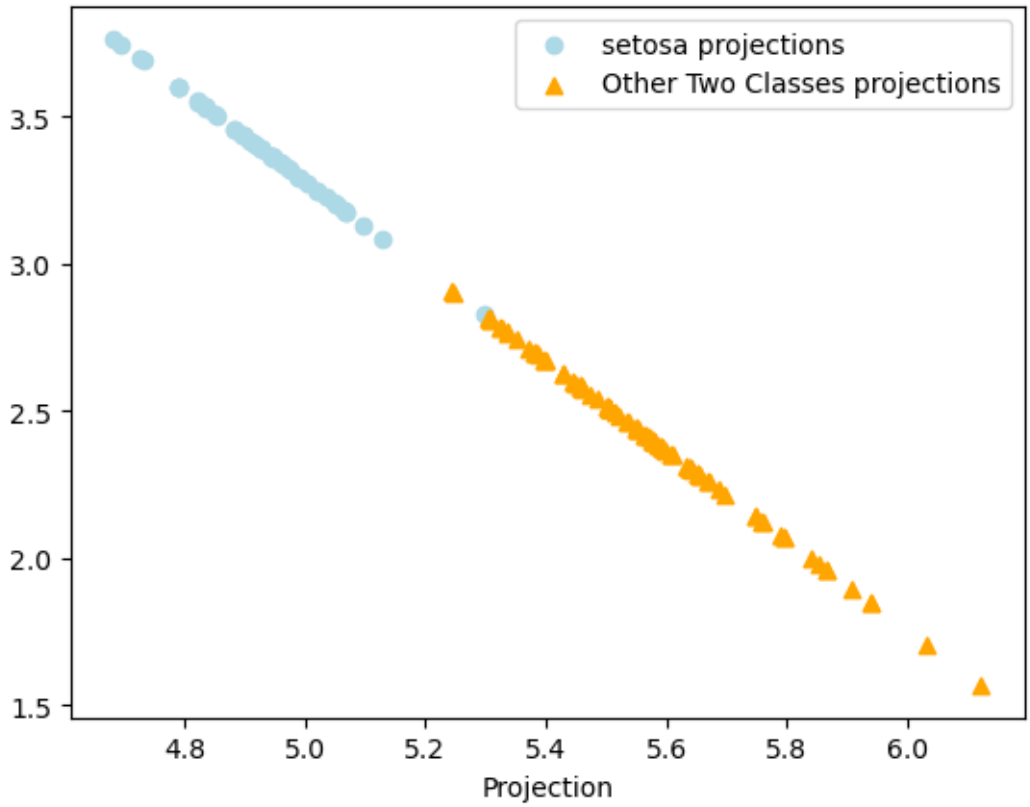
lda

September 24, 2023

[1]:
```
#best W projections
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
sepal_length = iris.data[:, 0]  # Sepal length values
sepal_width = iris.data[:, 1]   # Sepal width values
class_labels = iris.target       # Class labels (0: setosa, 1: versicolor, 2:␣
 ↪virginica)

# Separate the data into two classes
c1_data = np.array([[sl, sw] for sl, sw, cls in zip(sepal_length, sepal_width,␣
 ↪class_labels) if cls == 0])
c2_data = np.array([[sl, sw] for sl, sw, cls in zip(sepal_length, sepal_width,␣
 ↪class_labels) if cls != 0])

# Calculate the means of each class
mean_c1 = np.mean(c1_data, axis=0)
mean_c2 = np.mean(c2_data, axis=0)

# Calculate the between-class scatter matrix
sb = np.outer(mean_c1 - mean_c2, mean_c1 - mean_c2)

# Calculate the within-class scatter matrix
sw = ((c1_data-mean_c1).T)@(c1_data-mean_c1) + ((c2_data-mean_c2).
 ↪T)@(c2_data-mean_c2)

# Calculate the eigenvectors and eigenvalues of the generalized eigenvalue␣
 ↪problem
eigenvalues, eigenvectors = np.linalg.eig(np.linalg.inv(sw) @ sb)

# Sort the eigenvectors based on the eigenvalues
sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvectors = eigenvectors[:, sorted_indices]
```

```
# Select the best eigenvector as the projection vector
w = eigenvectors[:, 0]

# Project all the points onto the projection vector
c1_projections = c1_data @ w
c2_projections = c2_data @ w

c1_distances = np.abs(c1_data @ w - mean_c1 @ w)
c2_distances = np.abs(c2_data @ w - mean_c2 @ w)

c1projectionXaxis = c1_projections * w[0] + 5
c1projectionYaxis = c1_projections * w[1] + 3.28
c2projectionXaxis = c2_projections * w[0] + 5
c2projectionYaxis = c2_projections * w[1] + 3.28


# Plot the projections
plt.scatter(c1projectionXaxis, c1projectionYaxis, marker='o',␣
 ↪color='lightblue', label='setosa projections')
plt.scatter(c2projectionXaxis, c2projectionYaxis, marker='^', color='orange',␣
 ↪label='Other Two Classes projections')

plt.xlabel('Projection')
plt.legend()
plt.show()

# Print the projected means
mean_c1_projection = np.mean(c1_projections)
mean_c2_projection = np.mean(c2_projections)
print("Projected Mean of setosa (c1):", mean_c1_projection)
print("Projected Mean of versicolor/virginica (c2):", mean_c2_projection)
print("W: ", w)

print("\n a_i class 1:", c1_distances)
print("\n\n a_i class 1:", c2_distances)
```



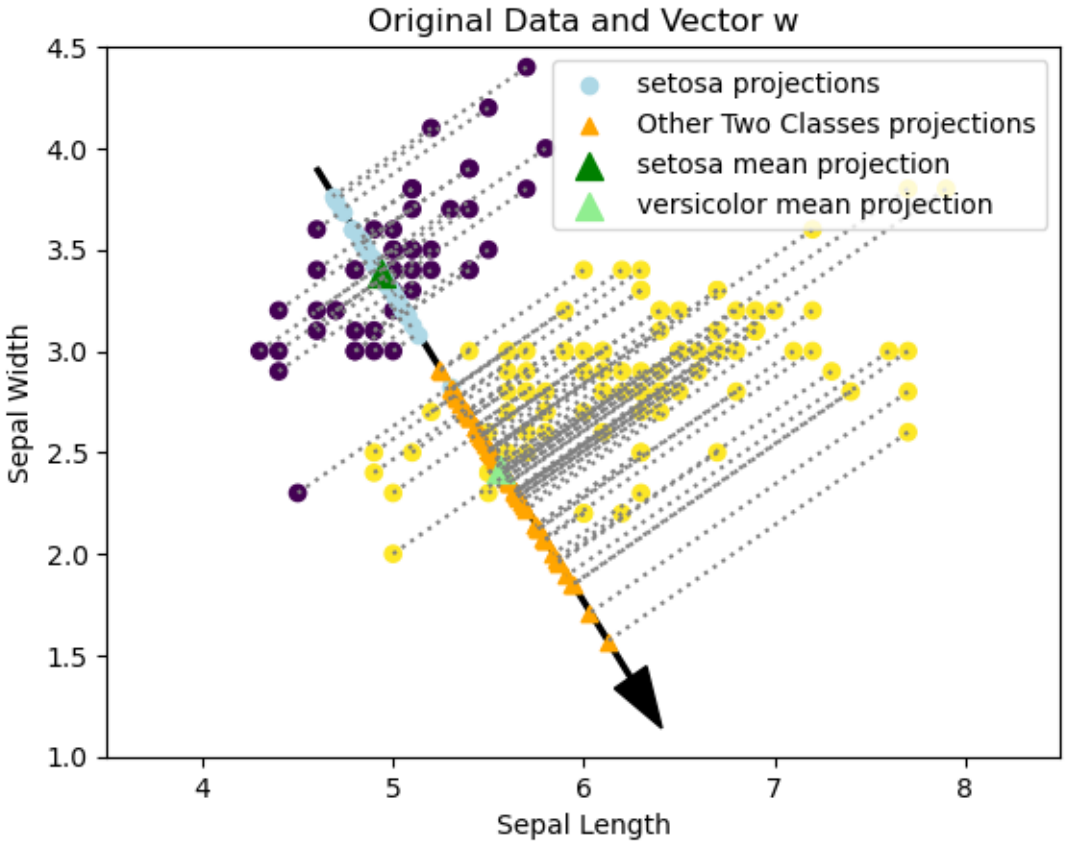

```
Projected Mean of setosa (c1): -0.12192442828309763
Projected Mean of versicolor/virginica (c2): 1.0317203184939974
W:  [ 0.5483082  -0.83627634]

 a_i class 1: [0.00867093 0.2998056  0.0228887  0.05168551 0.14712938 0.178689
 0.19919739 0.02012589 0.10927914 0.21617797 0.01143373 0.08953575
 0.24497478 0.02917932 0.04299335 0.43233471 0.178689   0.00867093
 0.06943109 0.25955383 0.23944917 0.17592619 0.36645266 0.15858434
 0.08953575 0.35463642 0.02012589 0.04615989 0.12978753 0.0228887
 0.16134715 0.23944917 0.45560591 0.37474108 0.21617797 0.18738116
 0.21065236 0.2019602  0.0256515  0.07495671 0.06350175 0.66587576
 0.14160377 0.06350175 0.25955383 0.24497478 0.25955383 0.03194212
 0.06626455 0.10375352]


 a_i class 1: [1.30352814e-01 1.98632107e-01 1.59149628e-01 6.05392156e-02
 1.90709249e-01 2.47937313e-01 3.37090561e-01 3.52073340e-01
 1.61912435e-01 4.38463781e-01 3.72680160e-02 3.05530941e-01
 4.18320951e-01 1.12241666e-01 3.86395768e-01 4.94879874e-02
 4.70023401e-01 1.09478859e-01 5.27982591e-01 5.18852320e-02
 4.72786208e-01 2.86140324e-02 3.31930510e-01 2.86140324e-02
```

```
5.22507945e-02 7.82848011e-02 3.55201709e-01 1.33115621e-01
1.67072487e-01 8.06820456e-02 2.30884183e-02 2.30884183e-02
1.09478859e-01 1.82781248e-04 5.79685042e-01 5.85210656e-01
4.94879874e-02 4.99185778e-01 4.70023401e-01 1.06716052e-01
1.90343686e-01 1.95869300e-01 2.58512254e-02 2.13614886e-01
2.19140500e-01 4.15192581e-01 3.31564947e-01 5.74108460e-02
3.26039333e-01 2.47937313e-01 3.37090561e-01 1.09478859e-01
3.52438902e-01 2.58002580e-03 2.34539808e-02 6.26593004e-01
4.35700974e-01 5.45728177e-01 5.51253791e-01 9.44960808e-02
1.43801287e-01 2.19506062e-01 1.87946442e-01 2.94558829e-03
1.93106493e-01 1.98632107e-01 2.34539808e-02 1.24027526e-02
1.01593436e+00 4.18320951e-01 7.55219940e-02 3.02768134e-01
8.48679092e-01 1.64675242e-01 1.17767280e-01 2.40014455e-01
2.62167879e-02 1.95869300e-01 1.35878428e-01 4.07269723e-01
6.84186631e-01 1.22064393e-01 1.35878428e-01 8.10476081e-02
1.38641235e-01 6.81423824e-01 4.20718195e-01 1.15004473e-01
2.50700120e-01 1.59149628e-01 4.94879874e-02 1.59149628e-01
1.09478859e-01 2.06911738e-02 1.17767280e-01 1.33115621e-01
3.31930510e-01 2.34539808e-02 4.75549016e-01 3.05530941e-01]
```

[2]:
```
#projections on the best W
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
sepal_length = iris.data[:, 0]  # Sepal length values
sepal_width = iris.data[:, 1]   # Sepal width values
class_labels = iris.target       # Class labels (0: setosa, 1: versicolor, 2:␣
 ↪virginica)

# Separate the data into two classes
c1_data = np.array([[sl, sw] for sl, sw, cls in zip(sepal_length, sepal_width,␣
 ↪class_labels) if cls == 0])
c2_data = np.array([[sl, sw] for sl, sw, cls in zip(sepal_length, sepal_width,␣
 ↪class_labels) if cls != 0])

# Calculate the means of each class
mean_c1 = np.mean(c1_data, axis=0)
mean_c2 = np.mean(c2_data, axis=0)

# Calculate the between-class scatter matrix
sb = np.outer(mean_c1 - mean_c2, mean_c1 - mean_c2)

# Calculate the within-class scatter matrix
```

```
sw = ((c1_data-mean_c1).T)@(c1_data-mean_c1) + ((c2_data-mean_c2).
 ↪T)@(c2_data-mean_c2)

# Calculate the eigenvectors and eigenvalues of the generalized eigenvalue␣
 ↪problem
eigenvalues, eigenvectors = np.linalg.eig(np.linalg.inv(sw) @ sb)

# Sort the eigenvectors based on the eigenvalues
sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvectors = eigenvectors[:, sorted_indices]

# Select the best eigenvector as the projection vector
w = eigenvectors[:, 0]


# Project all the points onto the projection vector
c1_projections = c1_data @ w
c2_projections = c2_data @ w

# make other two classes than Setosa as one single class
class_labels[class_labels == 2] = 1


# Scale the vector w
scaling_factor = 3.0  # Adjust the scaling factor as desired
scaled_w = w * scaling_factor


# Plot the original data points and the vector w
plt.scatter(sepal_length, sepal_width, c=class_labels, cmap='viridis')
plt.arrow(4.6, 3.9, scaled_w[0], scaled_w[1], color='black', width=0.02,␣
 ↪head_width=0.2)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Original Data and Vector w')

c1projectionXaxis = c1_projections * w[0] + 5
c1projectionYaxis = c1_projections * w[1] + 3.28
c2projectionXaxis = c2_projections * w[0] + 5
c2projectionYaxis = c2_projections * w[1] + 3.28

# Plot the projections of the points on vector w

plt.scatter(c1projectionXaxis , c1projectionYaxis, marker='o',␣
 ↪color='lightblue', label='setosa projections')
plt.scatter(c2projectionXaxis,c2projectionYaxis , marker='^', color='orange',␣
 ↪label='Other Two Classes projections')
```

```
# Plot the means of the projections
plt.scatter(np.mean(c1projectionXaxis), np.mean(c1projectionYaxis), marker='^',␣
 ↪color='green', label='setosa mean projection', s=100)
plt.scatter(np.mean(c2projectionXaxis), np.mean(c2projectionYaxis), marker='^',␣
 ↪color='lightgreen', label='versicolor mean projection', s=100)

# Plot the distances as dotted lines
for i in range(len(c1_data)):
    plt.plot([c1_data[i, 0], c1projectionXaxis[i]], [c1_data[i, 1],␣
 ↪c1projectionYaxis[i]], color='grey', linestyle='dotted')
for i in range(len(c2_data)):
    plt.plot([c2_data[i, 0], c2projectionXaxis[i]], [c2_data[i, 1],␣
 ↪c2projectionYaxis[i]], color='grey', linestyle='dotted')

plt.xlim(3.5,8.5)
plt.ylim(1,4.5)

plt.legend()
plt.show()
```



[ ]: