Exercise 1

a) We can reformulate the problem with the constraint as follows:

$$\max_w \, w^\top S_W w$$

$$\text{s. t. } w^\top S_B w = 1$$

b) The Lagrangrian is given by

$$L(w, \lambda) = w^\top S_W w - \lambda(w^\top S_B w - 1)$$

Setting the gradient to zero gives us this necessary equation

$$\partial_w L(w, \lambda) = 2S_W w - 2\lambda S_B w = 0$$

This implies

$$S_W w = \lambda S_B w$$

So a solution of the problem in a) also solves this generalized eigenvalue problem.

c) Rewriting the between-class scatter matrix in the generalized eigenvalue problem yields

$$S_W w = \lambda(m_2 - m_1)(m_2 - m_1)^\top w$$

Now notice that $a := (m_2 - m_1)^\top w \in \mathbb{R}$ is just some scalar. So multiplying both sides of the equation with the inverse of S_W (which exists because we assumed S_W to be positive definite) gives us

$$w = a\lambda S_W^{-1}(m_2 - m_1)$$

Thus we get

$$w^\star \propto S_W^{-1}(m_2 - m_1)$$

2 a

$$p(error|x) = \min\{p(\omega_1|x), p(\omega_2|x)\}$$

We assume, $p(\omega_1|x) > p(\omega_2|x)$

$$p(error|x) = p(\omega_2|x) = \sqrt{p(\omega_2|x)p(\omega_2|x)} \leq \sqrt{p(\omega_1|x)p(\omega_2|x)}$$

if $p(\omega_1|x) < p(\omega_2|x)$

$$p(error|x) = p(\omega_1|x) = \sqrt{p(\omega_1|x)p(\omega_1|x)} \leq \sqrt{p(\omega_1|x)p(\omega_2|x)}$$

So combine the 2 inequalities above we conclude, $p(error|x) \leq \sqrt{p(\omega_1|x)p(\omega_2|x)}$

2 b

$$p(error) = \int_x p(error|x)p(x)dx$$

$$p(error) \leq \int_x \sqrt{p(\omega_1|x)p(\omega_2|x)}p(x)dx$$

$$p(error) \leq \int_x \sqrt{\frac{p(x|\omega_1)p(\omega_1)}{p(x)}\frac{p(x|\omega_2)p(\omega_2)}{p(x)}}p(x)dx$$

$$p(error) \leq \sqrt{p(\omega_1)p(\omega_2)}\int_x \sqrt{p(x|\omega_1)p(x|\omega_2)}dx$$

$$p(error) \leq \sqrt{p(\omega_1)p(\omega_2)}\int_x \sqrt{\frac{1}{(2\pi)^{0.5d}}\frac{1}{|\Sigma|^{0.5}}\exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}}$$
$$\sqrt{\frac{1}{(2\pi)^{0.5d}}\frac{1}{|\Sigma|^{0.5}}\exp\{-\frac{1}{2}(x+\mu)^T\Sigma^{-1}(x+\mu)\}}dx$$

$$p(error) \leq \sqrt{p(\omega_1)p(\omega_2)}\int_x \frac{1}{(2\pi)^{0.5d}}\frac{1}{|\Sigma|^{0.5}}\sqrt{\exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)-\frac{1}{2}(x+\mu)^T\Sigma^{-1}(x+\mu)\}}dx$$

Inner exponent it gave $-x^T\Sigma^{-1}x - \mu^T\Sigma^{-1}\mu$, so we have,

$$p(error) \leq \sqrt{p(\omega_1)p(\omega_2)}\exp\{-\frac{1}{2}\mu^T\Sigma^{-1}\mu\}\underbrace{\int_x \frac{1}{(2\pi)^{0.5d}}\frac{1}{|\Sigma|^{0.5}}\exp\{-\frac{1}{2}x^T\Sigma^{-1}x\}dx}_{\int_x N(0,\Sigma)dx=1}$$

So we could conclude, the Bayes error rate can then be upper-bounded by:

$$p(error) \leq \sqrt{p(\omega_1)p(\omega_2)}\exp(-\frac{1}{2}\mu^T\Sigma^{-1}\mu)$$

Exercise 3

a) We can maximize the ratio $\frac{w^\top S_B w}{w^\top S_W w}$ using the result from Exercise 1c):

$$w^\star = S_W^{-1}(m_1 - m_2)$$

The within-class scatter matrix S_W is the sum of the covariance matrices Σ_1, Σ_2. So we get the Fisher discriminant

$$w^\star = (\Sigma_1 + \Sigma_2)^{-1}(m_1 - m_2) = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}^{-1} \begin{pmatrix} -2 \\ -2 \end{pmatrix} = \begin{pmatrix} 1/4 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} -2 \\ -2 \end{pmatrix} = \begin{pmatrix} -1/2 \\ -1 \end{pmatrix}$$

b) Now to minimize the ratio $\frac{w^\top S_B w}{w^\top S_W w}$ notice that S_W is positive-definite since all eigenvalues are positive. Further observe the between-class scatter matrix

$$S_B = (m_2 - m_1)(m_2 - m_1)^\top = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \begin{pmatrix} 2 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix}$$

Further notice for all 2-dimensional vectors $w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$

$$w^\top S_B w = 4w_1^2 + 8w_1 w_2 + 4w_2^2 = 4(w_1 + w_2)^2 \geq 0$$

and $= 0$ if and only if $w_1 = -w_2$. So in total we obtain

$$\frac{w^\top S_B w}{w^\top S_W w} \geq 0$$

and $= 0$ if and only if $w_1 = -w_2$. So the ratio is minimized for example for the projection $w^\star = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

sheet04

November 29, 2020

1 Fisher Linear Discriminant

In this exercise, we apply Fisher Linear Discriminant as described in Chapter 3.8.2 of Duda et al. on the UCI Abalone dataset. A description of the dataset is given at the page https://archive.ics.uci.edu/ml/datasets/Abalone. The following two methods are provided for your convenience:

- **`utils.Abalone.__init__(self)`** reads the Abalone data and instantiates two data matrices corresponding to: *infant (I)*, *non-infant (N)*.
- **`utils.Abalone.plot(self,w)`** produces a histogram of the data when projected onto a vector `w`, and where each class is shown in a different color.

Sample code that makes use of these two methods is given below. It loads the data, looks at the shape of instantiated matrices, and plots the projection on the first dimension of the data representing the length of the abalone.

[2]:

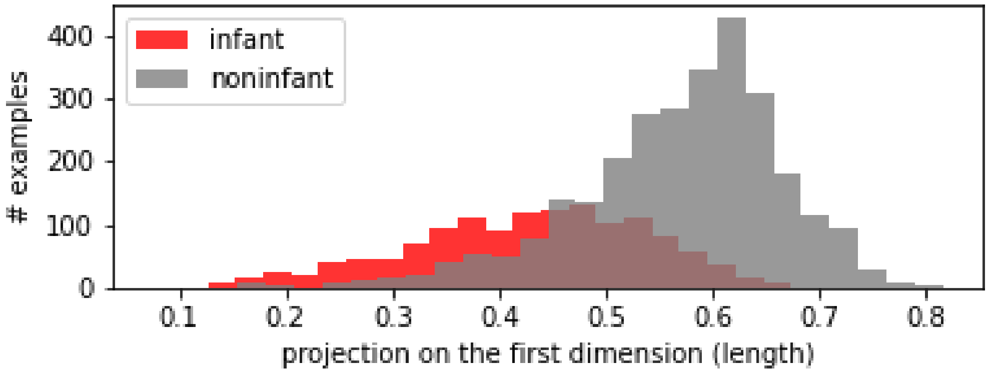
```
%matplotlib inline
import utils,numpy

# Load the data
abalone = utils.Abalone()

# Print dataset size for each class
print(abalone.I.shape, abalone.N.shape)

# Project data on the first dimension
w1 = numpy.array([1,0,0,0,0,0,0])
abalone.plot(w1,'projection on the first dimension (length)')
```

```
(1342, 7) (2835, 7)
```



1.1 Implementation (10 + 5 + 5 = 20 P)

- **Create a function `w = fisher(X1,X2)` that takes as input the data for two classes and returns the Fisher linear discriminant.**
- **Create a function `objective(X1,X2,w)` that evaluates the objective defined in Equation 96 of Duda et al. for an arbitrary projection vector `w`.**
- **Create a function `z = phi(X)` that returns a quadratic expansion for each data point `x` in the dataset. Such expansion consists of the vector `x` itself, to which we concatenate the vector of all pairwise products between elements of `x`.** In other words, letting $x = (x_1, \ldots, x_d)$ denote the d-dimensional data point, the quadratic expansion for this data point is a $d \cdot (d+3)/2$ dimensional vector given by $\phi(x) = (x_i)_{1 \le i \le d} \cup (x_i x_j)_{1 \le i \le j \le d}$. For example, the quadratic expansion for $d = 2$ is $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.

[3]:

```python
def fisher(X1,X2):
    ##### Replace by your code
    m_1 = X1.mean(axis=0)
    m_2 = X2.mean(axis=0)

    Sw = (X1 - m_1).T@(X1 - m_1) + (X2 - m_2).T@(X2 - m_2)

    w = numpy.linalg.inv(Sw)@(m_1 - m_2)
    return w/numpy.linalg.norm(w)
    #####

def objective(X1,X2,w):
    ##### Replace by your code

    # Compute the means mu
    m_1 = X1.mean(axis=0)
    m_2 = X2.mean(axis=0)
    # Scatter matrices
    Sb = numpy.outer((m_1 - m_2), (m_1 - m_2))
    Sw = (X1 - m_1).T@(X1 - m_1) + (X2 - m_2).T@(X2 - m_2)

    return (w.T@Sb@w) / (w.T@Sw@w)
    #####

def expand(X):
    ##### Replace by your code
    X_c = X.copy()
    for i in range(X.shape[1]):
        for j in range(i+1):
            A = X[:, i] * X[:, j]
            X_c = numpy.append(X_c, A[:, numpy.newaxis], axis=1)
    return X_c
    #####
```

1.2 Analysis (5 + 5 = 10 P)

- **Print value of the objective function and the histogram for several values of `w`:**
- `w` is a canonical coordinate vector for the first feature (length).
- `w` is the difference between the mean vectors of the two classes.
- `w` is the Fisher linear discriminant.
- `w` is the Fisher linear discriminant (after quadratic expansion of the data).

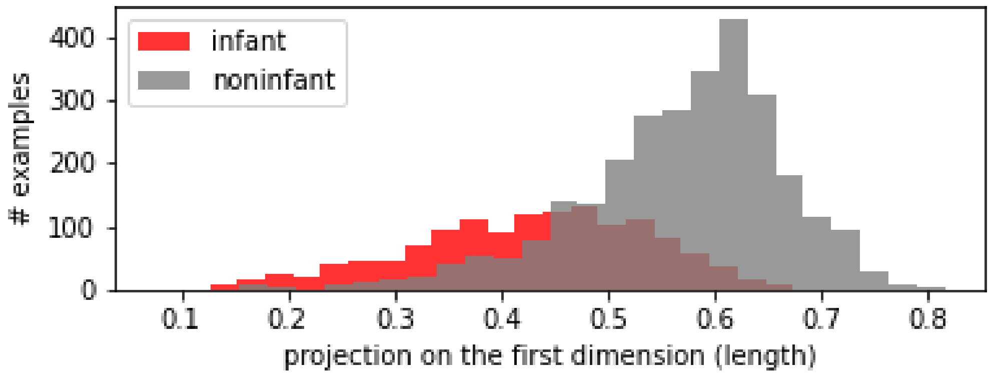
[4]:

```
##### REPLACE BY YOUR CODE
%matplotlib inline

X1 = abalone.I
X2 = abalone.N
w1 = numpy.array([1, 0, 0, 0, 0, 0, 0])

print(f"First dimension (length): {objective(X1, X2, w1):.5f}")
abalone.plot(w1, 'projection on the first dimension (length)')
#####
```

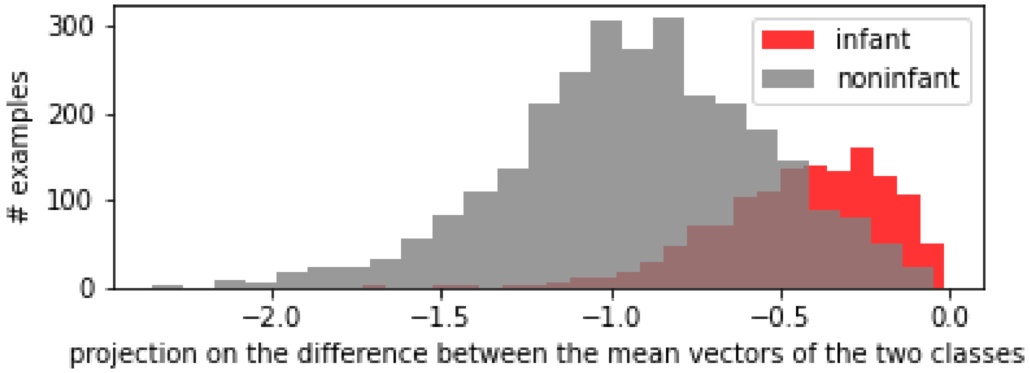
```
First dimension (length): 0.00048
```



```
[5]: w2 = X1.mean(axis=0) - X2.mean(axis=0)

     print(f"Means Linear: {objective(X1, X2, w2):.5f}")
     abalone.plot(w2, 'projection on the difference between the mean vectors of the␣
      ↪two classes')
```

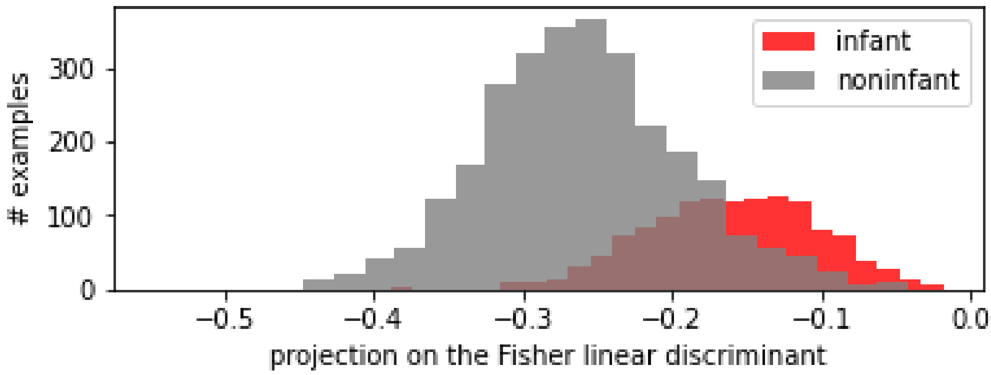
```
Means Linear: 0.00050
```



```
[6]: w3 = fisher(X1, X2)

     print(f"Fisher Linear: {objective(X1, X2, w3):.5f}")
     abalone.plot(w3, 'projection on the Fisher linear discriminant')
```

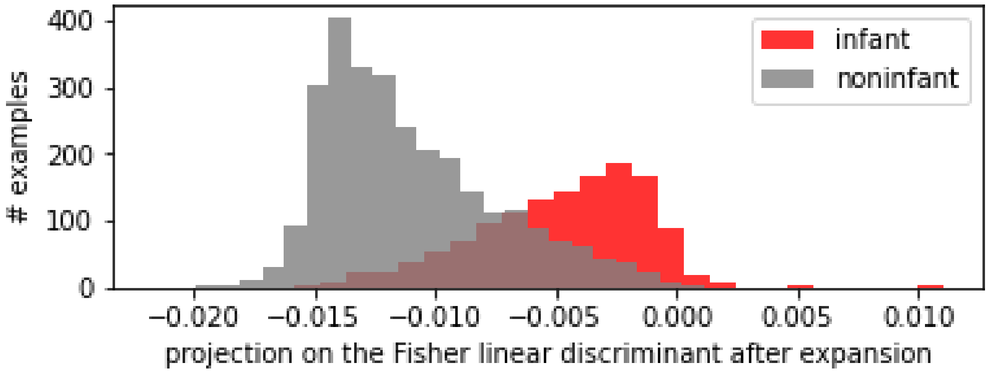
```
Fisher Linear: 0.00057
```



```
[7]: w4 = fisher(expand(X1), expand(X2))
     print(f"Fisher after expansion: {objective(expand(X1), expand(X2), w4):.5f}")
     name = 'projection on the Fisher linear discriminant after expansion'

     def plot(exb1,exb2,w,name):
         import matplotlib.pyplot as plt
         plt.figure(figsize=(6,2))
         plt.xlabel(name)
         plt.ylabel('# examples')
         plt.hist(numpy.dot(exb1,w),bins=25, alpha=0.8,color='red',label='infant')
         plt.hist(numpy.dot(exb2,w),bins=25, alpha=0.
      ↪8,color='gray',label='noninfant')
         plt.legend()
         plt.show()
     plot(expand(X1), expand(X2), w4, name)
```

```
Fisher after expansion: 0.00077
```



[ ]:

```
```