ML1_assignment6_program

November 26, 2018

1 Fisher Linear Discriminant

In this exercise, you will 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 three data matrices of size $(1528, 7)$, $(1307, 7)$, and $(1342, 7)$ corresponding to the three classes in the dataset: *male (M)*, *female (F)*, and *infant (I)*.

- `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 various projections of the data: (1) projection on the first dimension of the data, and (2) projection on a random direction.

```
In [1]: %matplotlib inline
        import utils,numpy

        # Load the data
        abalone = utils.Abalone()

        # Print dataset size for each class
        print(abalone.M.shape,abalone.F.shape, abalone.I.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')

        # Project data on a random direction
        w2 = numpy.random.normal(0,1,[7])
        w2 /= (w2**2).sum()**.5
        abalone.plot(w2,'projection on a random direction')

(1528, 7) (1307, 7) (1342, 7)
```
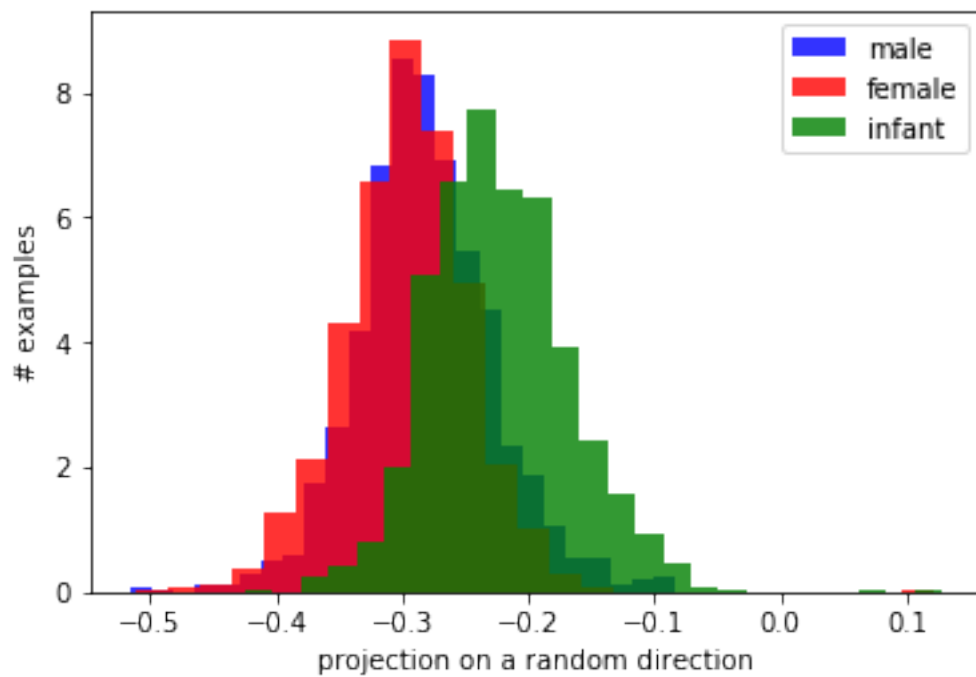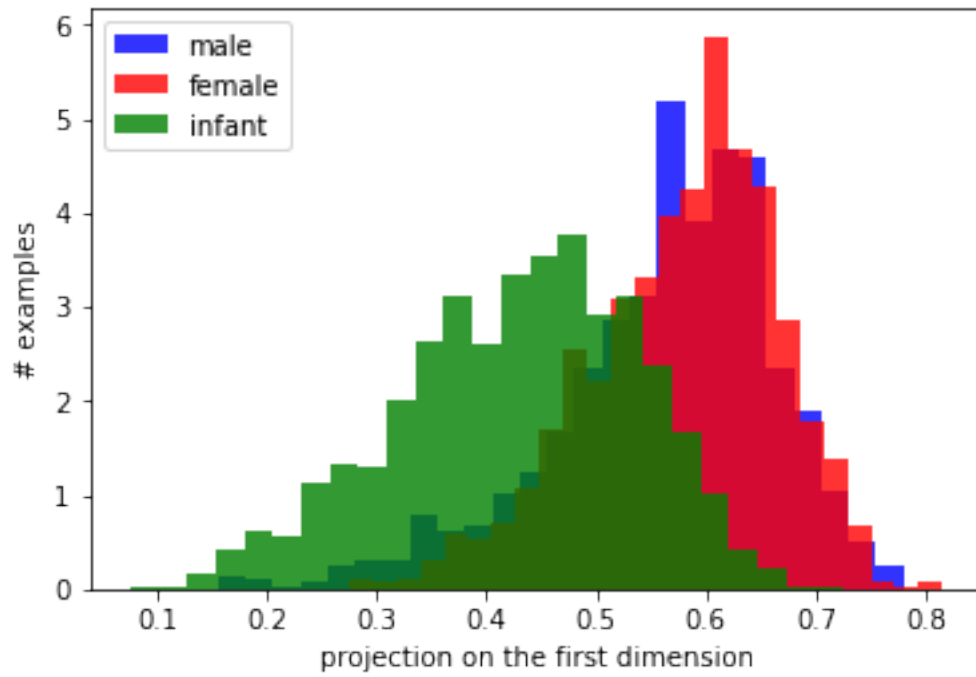
C:\Users\Nika\Anaconda3\lib\site-packages\matplotlib\axes_axes.py:6571: UserWarning: The 'norme
warnings.warn("The 'normed' kwarg is deprecated, and has been "

1.1 Implementation (30 P)

- **Create a method `w = fisher(X1,X2)` that takes as input the data for two classes and returns the Fisher linear discriminant.**

- **Create a method `J(X1,X2,w)` that evaluates the objective defined in Equation 96 of Duda et al. for an arbitrary projection vector `w`.**

- **Create a method `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 $\phi(x) = (x_i)_{1 \leq i \leq d} \cup (x_i x_j)_{1 \leq i \leq j \leq d}$. For example, the quadratic expansion for $d = 2$ is $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.

```
In [2]: def calcMean(X):
            return numpy.mean(X, axis = 0)

        def SB(X1, X2):
            mu1 = calcMean(X1)
            mu2 = calcMean(X2)

            mu = numpy.zeros((1, len(mu1)))
            mu[0] = (mu1-mu2)

            return numpy.dot(mu.T, mu)

        def SW(X1, X2):
            mu1 = calcMean(X1)
            mu2 = calcMean(X2)

            w1 = numpy.dot((X1 - mu1).T,(X1 - mu1))
            w2 = numpy.dot((X2 - mu2).T,(X2 - mu2))

            return w1 + w2

        def fisher(X1, X2):
            mu1 = calcMean(X1)
            mu2 = calcMean(X2)
            Sw = SW(X1,X2)

            return numpy.dot(numpy.linalg.inv(Sw), (mu2 - mu1))

        def JW(X1, X2, w):
            Sb = SB(X1, X2)
            Sw = SW(X1, X2)
```

```
        W = numpy.zeros((1, len(w)))
        W[0] = (w)

        Numerator = numpy.dot(W, numpy.dot(Sb, W.T))
        denominator = numpy.dot(W, numpy.dot(Sw, W.T))

        return Numerator/denominator

    def phi(X):
        N = numpy.zeros((int(X.shape[1]*(X.shape[1]+3)/2),7))
        for i in range(0,X.shape[1]):
            N[i] = X[i]
        index = X.shape[1]
        for i in range(0,X.shape[1]):
            for j in range (i, X.shape[1]):
                N[index] = (X[i]*X[j])
                index += 1
        return N
```

1.2 Analysis (20 P)

- **Print the value of `J(w)` for each discriminated pair of classes (M/F, M/I, F/I), and for several values of `w`:**

- `w` is a vector that projects the data on the each dimension of the data.

- `w` is the difference between the mean vectors of the two classes.

- `w` is the difference between the mean vectors of the two classes (after quadratic expansion of the data).

- `w` is the Fisher linear discriminant.

- `w` is the Fisher linear discriminant (after quadratic expansion of the data).

- **For the simple Fisher linear discriminant, plot a histogram of the projected data for each discriminated pair of classes using the function `utils.Abalone.plot()`.**

In [3]: %matplotlib inline

```
        eye = numpy.eye(7, dtype = float)
        Mdata = phi(abalone.M)
        Fdata = phi(abalone.F)
        Idata = phi(abalone.I)

        print("    w                 M/F      M/I      F/I")
        print("-------------------------------------------")
        iterater = 0
        #w is a vector that projects the data on the each dimension of the data.
        for w in eye:
```

4

```python
        JW_MF = JW(abalone.M,abalone.F,w)
        JW_MI = JW(abalone.M,abalone.I,w)
        JW_FI = JW(abalone.F,abalone.I,w)
        print("Dimension %d \t %.5f %.5f %.5f" % (iterater,JW_MF,JW_MI,JW_FI))
        iterater +=1
print("-----------------------------------------")

#w is the difference between the mean vectors of the two classes.
wMF = numpy.mean(abalone.F,axis=0) - numpy.mean(abalone.M,axis=0)
wMI = numpy.mean(abalone.I,axis=0) - numpy.mean(abalone.M,axis=0)
wFI = numpy.mean(abalone.I,axis=0) - numpy.mean(abalone.F,axis=0)
JW_MF = JW(abalone.M,abalone.F,wMF)
JW_MI = JW(abalone.M,abalone.I,wMI)
JW_FI = JW(abalone.F,abalone.I,wFI)
print("Means Linear\t %.5f %.5f %.5f" % (JW_MF,JW_MI,JW_FI))

#w is the difference between the mean vectors of the two classes (after quadratic expans
wMF = numpy.mean(Mdata,axis=0) - numpy.mean(Fdata,axis=0)
wMI = numpy.mean(Mdata,axis=0) - numpy.mean(Idata,axis=0)
wFI = numpy.mean(Fdata,axis=0) - numpy.mean(Idata,axis=0)
JW_MF = JW(abalone.M,abalone.F,wMF)
JW_MI = JW(abalone.M,abalone.I,wMI)
JW_FI = JW(abalone.F,abalone.I,wFI)
print("Means Quadric \t %.5f %.5f %.5f" % (JW_MF,JW_MI,JW_FI))
print("-----------------------------------------")

#w is the Fisher linear discriminant.
wMF = fisher(abalone.M,abalone.F)
wMI = fisher(abalone.M,abalone.I)
wFI = fisher(abalone.F,abalone.I)
JW_MF = JW(abalone.M,abalone.F,wMF)
JW_MI = JW(abalone.M,abalone.I,wMI)
JW_FI = JW(abalone.F,abalone.I,wFI)
print("Fisher Linear\t %.5f %.5f %.5f" % (JW_MF,JW_MI,JW_FI))

#w is the Fisher linear discriminant (after quadratic expansion of the data).
wMF = fisher(Mdata,Fdata)
wMI = fisher(Mdata,Idata)
wFI = fisher(Fdata,Idata)
JW_MF = JW(abalone.M,abalone.F,wMF)
JW_MI = JW(abalone.M,abalone.I,wMI)
JW_FI = JW(abalone.F,abalone.I,wFI)
print("Fisher Quadritc  %.5f %.5f %.5f" % (JW_MF,JW_MI,JW_FI))


#plot
wMF = fisher(abalone.M,abalone.F)
wMI = fisher(abalone.M,abalone.I)
```
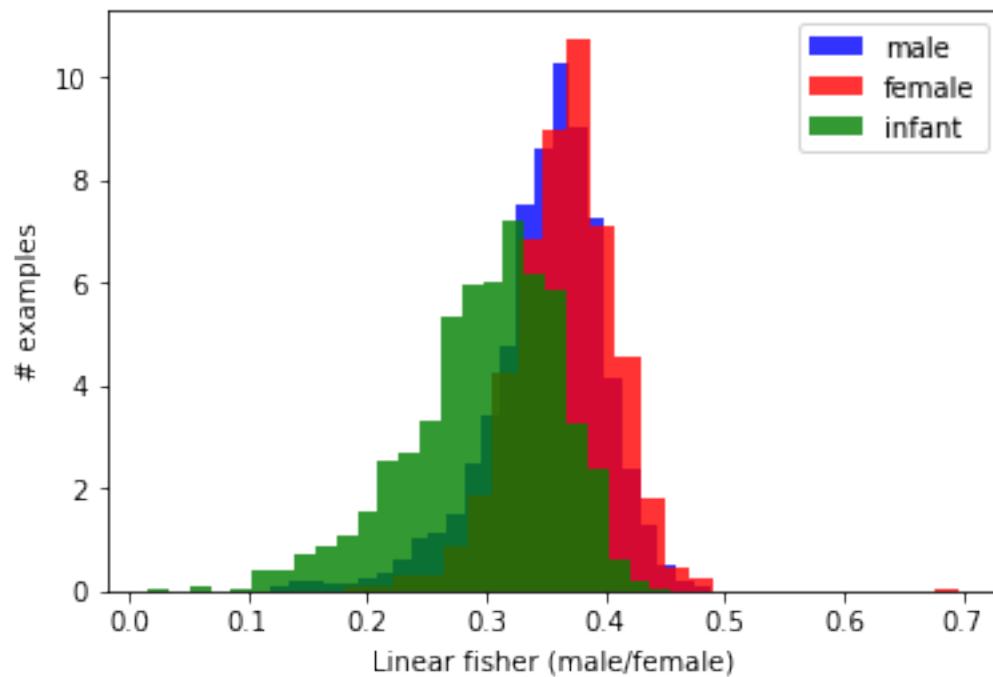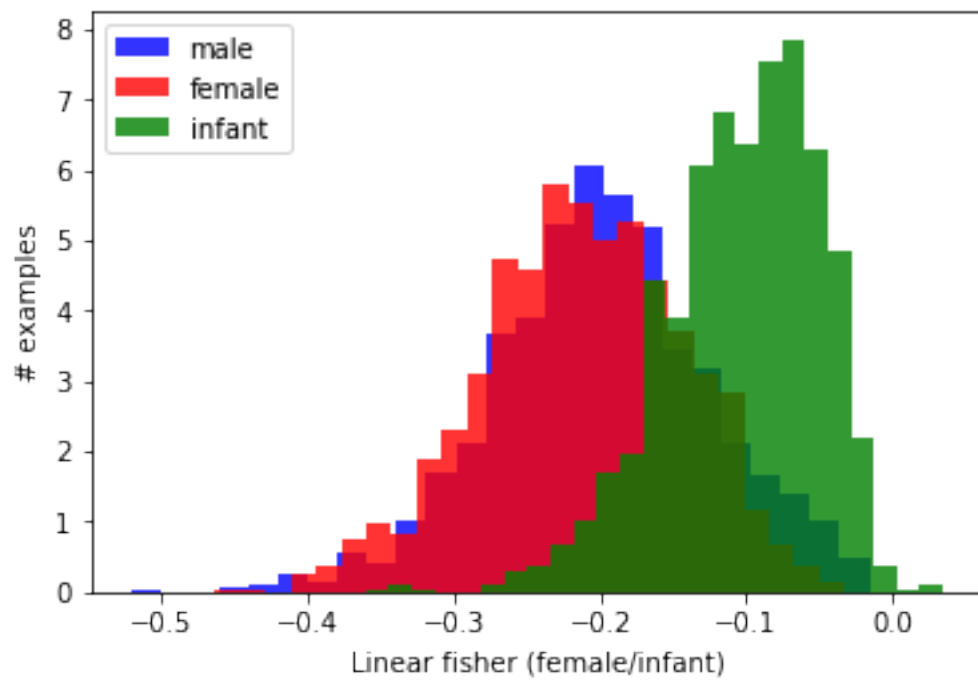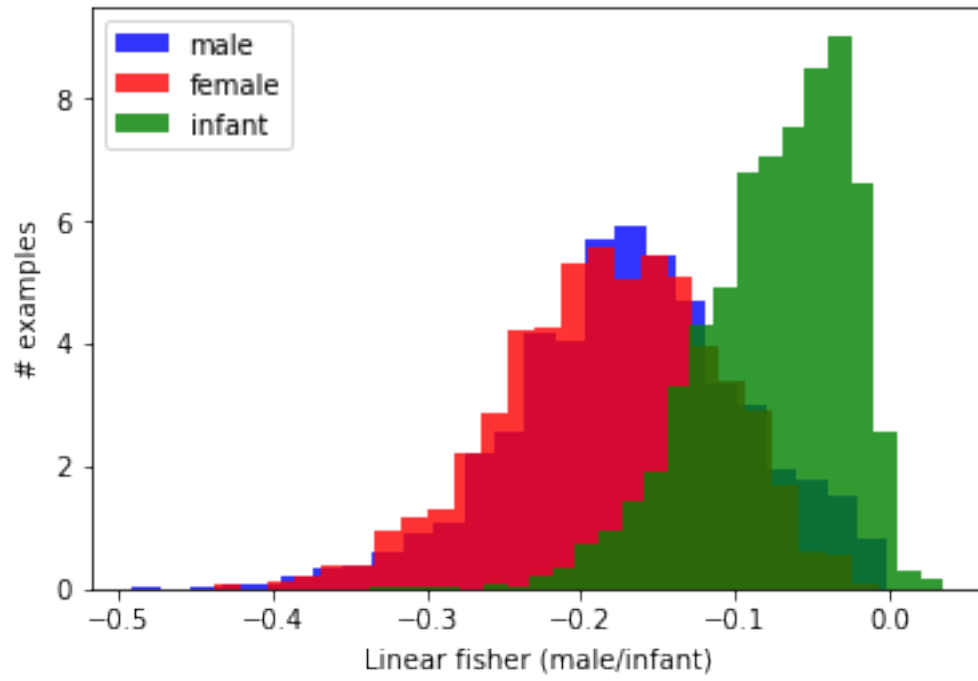
```
wFI = fisher(abalone.F,abalone.I)
abalone.plot(wMF/numpy.linalg.norm(wMF),'Linear fisher (male/female)')
abalone.plot(wMI/numpy.linalg.norm(wMI),'Linear fisher (male/infant)')
abalone.plot(wFI/numpy.linalg.norm(wFI),'Linear fisher (female/infant)')
```

```
  w              M/F     M/I     F/I
-------------------------------------------
Dimension 0          0.00001 0.00056 0.00090
Dimension 1          0.00001 0.00060 0.00097
Dimension 2          0.00001 0.00058 0.00072
Dimension 3          0.00001 0.00070 0.00108
Dimension 4          0.00000 0.00060 0.00088
Dimension 5          0.00001 0.00069 0.00109
Dimension 6          0.00001 0.00066 0.00100
-------------------------------------------
Means Linear         0.00001 0.00070 0.00108
Means Quadric         0.00001 0.00070 0.00108
-------------------------------------------
Fisher Linear         0.00004 0.00075 0.00120
Fisher Quadritc  0.00001 0.00023 0.00041
```

C:\Users\Nika\Anaconda3\lib\site-packages\matplotlib\axes_axes.py:6571: UserWarning: The 'norme
  warnings.warn("The 'normed' kwarg is deprecated, and has been "


```

In [ ]: