Machine Learning 1 WS18/19

Submission for Exercise Sheet 6

Hendrik Makait 384968
Michael Hoppe 362514
Wai Tang Victor Chan 406094
Rudi Poepsel Lemaitre 373017
Jonas Piotrowski 399334
Aki Saksala 399293

Exercise Sheet 6

1a) Consider $\frac{d}{dx}\left(\frac{w^T S_B w}{w^T S_w w}\right) = 0$

$$\frac{(w^T S_w w)(2S_B w) - (w^T S_B w)(2S_w w)}{(w^T S_w w)^2} = 0$$

$$(w^T S_w w)(S_B w) = (w^T S_B w)(S_w w)$$

$$S_B w = \underbrace{\left(\frac{w^T S_B w}{w^T S_w w}\right)}_{\text{Scalar}}(S_w w)$$

Hence $S_B w$ is a multiple of $S_w w$
or $S_B w = \lambda S_w w$ for some $\lambda \in \mathbb{R}$

(b) If S_w is invertible,

We can write $S_w^{-1} S_B w = \lambda w$

$$S_w^{-1}\left((m_1 - m_2)(m_1 - m_2)^T w\right) = \lambda w$$

$$w = \underbrace{\left(\frac{(m_1 - m_2)^T w}{\lambda}\right)}_{\text{Scalar}} S^{-1}(m_1 - m_2)$$

$\Rightarrow$ w is a multiple of $S^{-1}(m_1 - m_2)$.

Since w is an eigenvector of $S_w^{-1} S_B$ anyway, the scalar does not matter.

We can take $w = S^{-1}(m_1 - m_2)$

2a) $\mu_1 = \begin{pmatrix} \frac{4}{2} \\ \frac{3-1}{2} \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ $\mu_2 = \begin{pmatrix} \frac{-4}{2} \\ \frac{-3+1}{2} \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$

$\sigma^2 = \frac{1}{12} 4^2 = \frac{4}{3}$

$\Sigma_1 = \Sigma_2 = \frac{4}{3} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

b) $w = \Sigma^{-1} (\mu_1 - \mu_2)$

$= \frac{3}{4} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix}$

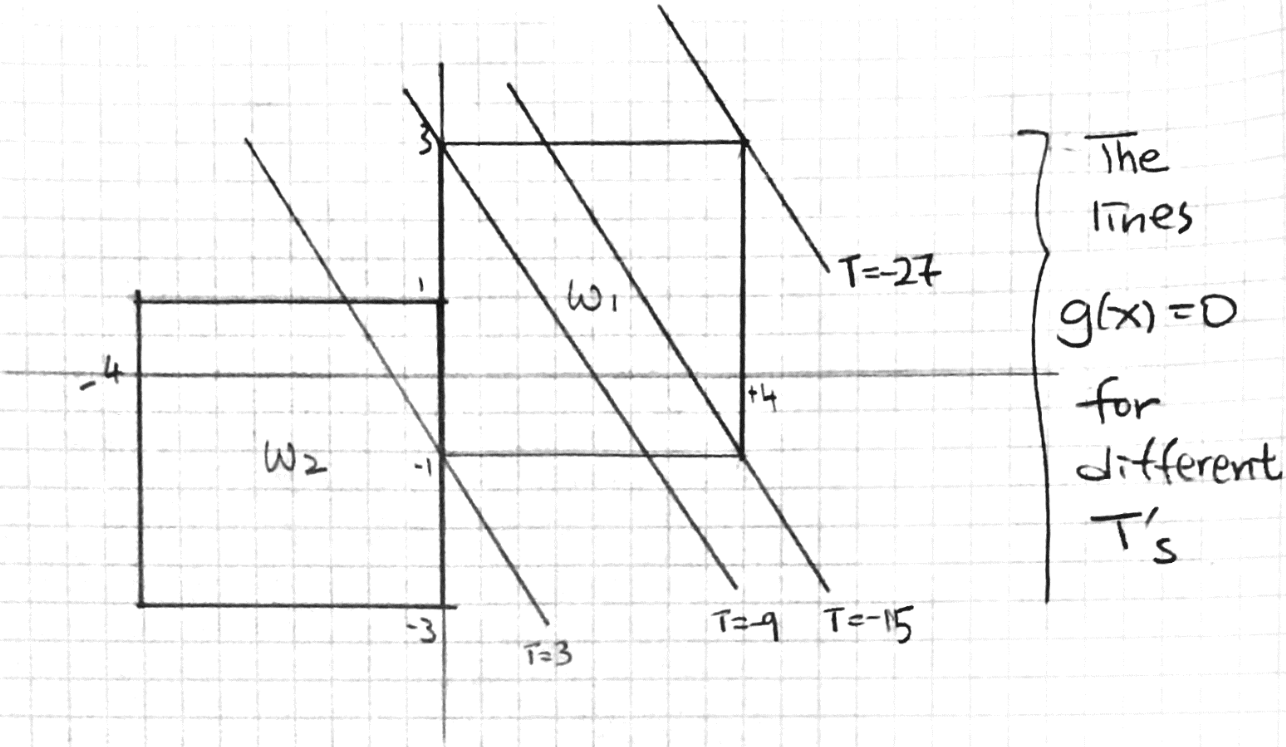
$= \begin{pmatrix} \frac{9}{4} \\ \frac{3}{2} \end{pmatrix}$

$b = \frac{1}{2} \left(T + \mu_2^T \Sigma^{-1} \mu_2 - \mu_1^T \Sigma^{-1} \mu_1 \right)$

$= \frac{1}{2} \left(T + \frac{3}{4}(4+1) - \frac{3}{4}(4+1) \right)$

$= \frac{1}{2} T$

2c)



Let $h(T) = P(g(x)<0 \mid w_1)$

When $T \geq 3$, $h(T) = 0$

When $-9 \leq T \leq 3$, $h(T) = \left(\frac{6-2T}{9}\right)\left(-\frac{T}{3}+1\right)\left(\frac{1}{2}\right)\left(\frac{1}{16}\right)$

$$= \frac{1}{432}T^2 - \frac{1}{72}T + \frac{1}{48}$$

When $-15 \leq T \leq -9$, $h(T) = \left(\frac{-18-2T}{9} + \frac{6-2T}{9}\right)\left(\frac{1}{2}\right)(4)\left(\frac{1}{16}\right)$

$$= -\frac{1}{18}T - \frac{1}{6}$$

When $-27 \leq T \leq -15$, $h(T) = 1 - \left(4 - \frac{-18-2T}{9}\right)\left(3 - \frac{-36-2T}{6}\right)\left(\frac{1}{2}\right)\left(\frac{1}{16}\right)$

$$= -\frac{1}{432}T^2 - \frac{1}{8}T - \frac{11}{16}$$

When $T \leq -27$, $h(T) = 1$

2c)

Similarly

$$P(g(x) > 0 | \omega_2)$$

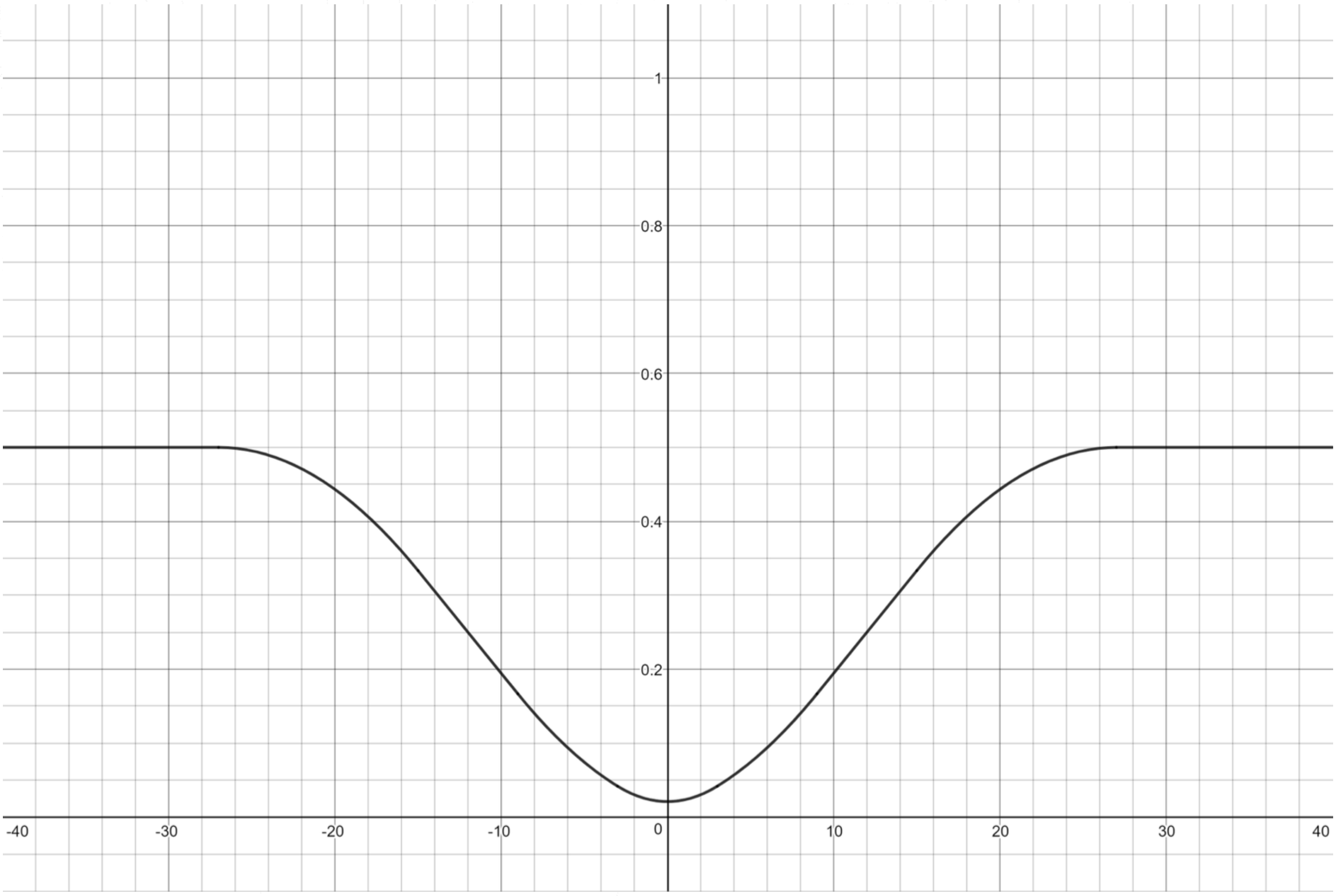
$$= \begin{cases} 0 & , T \geq 27 \\ -\frac{1}{432}T^2 + \frac{1}{8}T - \frac{11}{16} & , 15 \leq T \leq 27 \\ \frac{1}{18}T - \frac{1}{6} & , 9 \leq T \leq 15 \\ \frac{1}{432}T^2 + \frac{1}{72}T + \frac{1}{48} & , -3 \leq T \leq 9 \\ 0 & , T \leq -3 \end{cases}$$

Hence,

$$P(\text{error}) = \frac{1}{2} P(g(x) < 0 | \omega_1) + \frac{1}{2} P(g(x) > 0 | \omega_2)$$

$$= \begin{cases} \frac{1}{2} & T \leq -27 \\ -\frac{1}{864}T^2 - \frac{1}{16}T - \frac{11}{32} & -27 \leq T \leq -15 \\ -\frac{1}{36}T - \frac{1}{12} & -15 \leq T \leq -9 \\ \frac{1}{864}T^2 - \frac{1}{144}T + \frac{1}{96} & -9 \leq T \leq -3 \\ \frac{1}{432}T^2 + \frac{1}{48} & -3 \leq T \leq 3 \\ \frac{1}{864}T^2 + \frac{1}{144}T + \frac{1}{96} & 3 \leq T \leq 9 \\ \frac{1}{36}T - \frac{1}{12} & 9 \leq T \leq 15 \\ -\frac{1}{864}T^2 + \frac{1}{16}T - \frac{11}{32} & 15 \leq T \leq 27 \\ \frac{1}{2} & T \geq 27 \end{cases}$$

Graph:



2d) The Bayes Error Rate = 0,
since the two distributions are
completely disjoint.

2e) No

In (c), the lowest error (corr. to $T=0$) $= \frac{1}{48}$

2f) We can take $w = (1,0)^T$, $b = 0$

Then we see that it is a perfect classification

Programming Part

November 26, 2018

1 Fisher Linear Discriminant

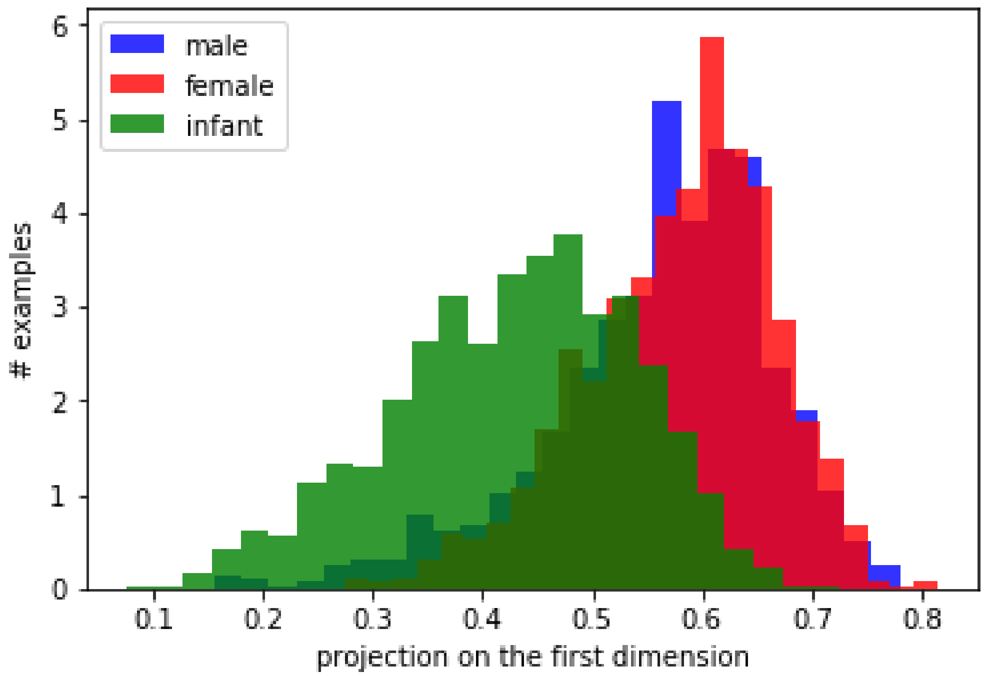
In this exercise, you will apply Fisher Linear Discriminant as described in Chapter 3.8.2 of Duda et al. on the UCI Abalone dataset. A description of the dataset is given at the page https://archive.ics.uci.edu/ml/datasets/Abalone. The following two methods are provided for your convenience:

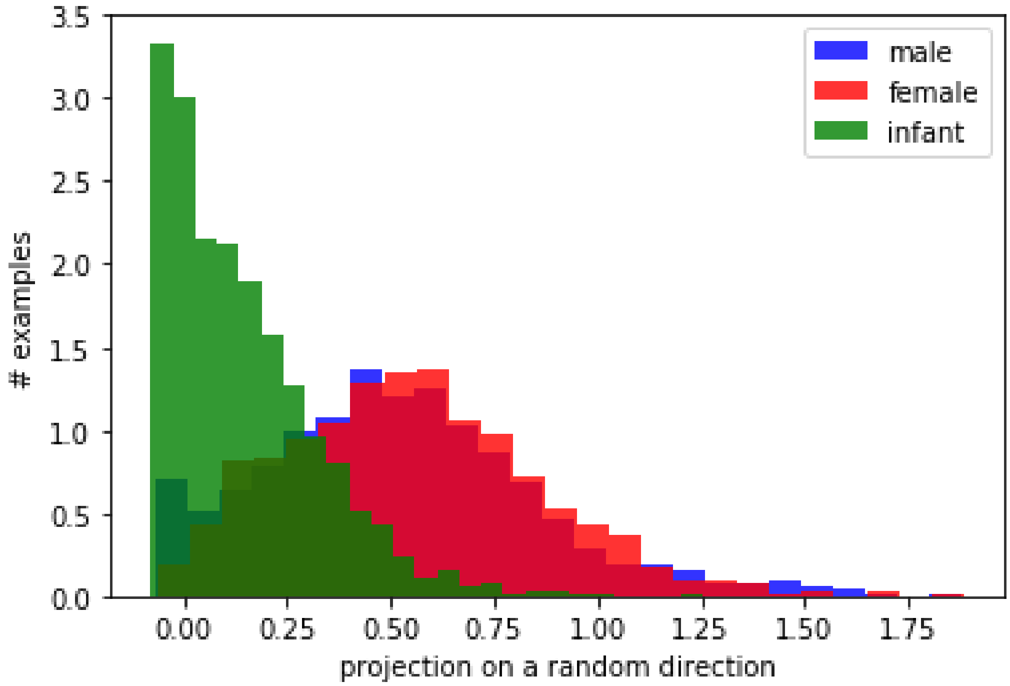
- **`utils.Abalone.__init__(self)`** reads the Abalone data and instantiates three data matrices of size (1528, 7), (1307, 7), and (1342, 7) corresponding to the three classes in the dataset: *male (M)*, *female (F)*, and *infant (I)*.
- **`utils.Abalone.plot(self,w)`** produces a histogram of the data 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 [53]: %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)
```


male
female
infant
examples
projection on the first dimension


male
female
infant
examples
projection on a random direction

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 [54]: import itertools

         def mean_vector(X):
             return numpy.sum(X, axis=0) / X.shape[0]

         def S_b(m1, m2):
             return numpy.outer((m1 - m2), (m1 - m2)) #https://docs.scipy.org/doc/numpy-1.15.1/

         def S_w(X1, X2, m1, m2):
             Sw1 = numpy.sum([numpy.outer((xi - m1), (xi - m1)) for xi in X1], axis=0)
             Sw2 = numpy.sum([numpy.outer((xi - m2), (xi - m2)) for xi in X2], axis=0)
             return Sw1 + Sw2

         def fisher(X1, X2):
             # mean vectors
             m1 = mean_vector(X1)
             m2 = mean_vector(X2)
             # between-class covariance matrix
             Sb = S_b(m1, m2)

             # within-class covariance matrix
             Sw = S_w(X1, X2, m1, m2)

             # compute the eigenvalue decomposition (Sb * w = Sw * w * lambda) -> (Sw^-1 * Sb

             Sw_inv = numpy.linalg.inv(Sw)

             eig_vals, eig_vecs = numpy.linalg.eig(Sw_inv.dot(Sb))

             # w is the largest eigenvector
             w = eig_vecs[numpy.argmax(eig_vals)]

             return w

         def J(X1, X2, w):
```

```
    m1 = mean_vector(X1)
    m2 = mean_vector(X2)
    Sb = S_b(m1, m2)
    Sw = S_w(X1, X2, m1, m2)
    # (wT * Sb * w) / (wT * Sw * w)
    t1 = numpy.matmul(numpy.matmul(w, Sb), w[:, numpy.newaxis])
    t2 = numpy.matmul(numpy.matmul(w, Sw), w[:, numpy.newaxis])
    return t1[0] / t2[0]

def phi(X):
    z = list()
    for x in X:
        a = list(x)
        a.extend([xi * xi for xi in x])
        a.extend([c[0] * c[1] for c in itertools.combinations(x, 2)])
        z.append(a)
    return numpy.asarray(z)
```

1.2 Analysis (20 P)

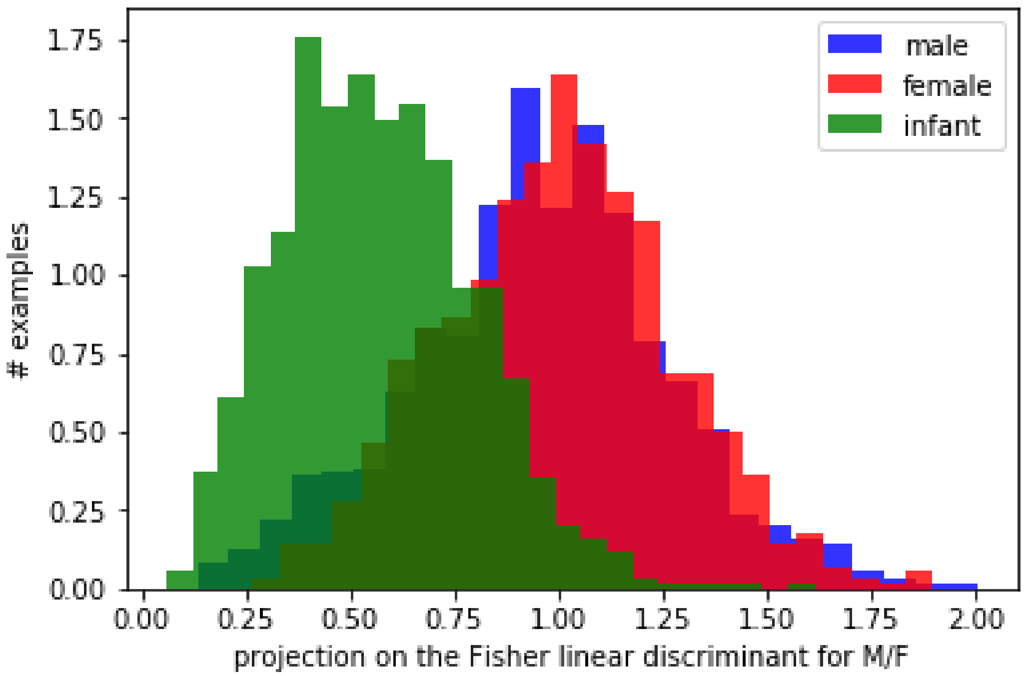
- **Print the value of `J(w)` for each discriminated pair of classes (M/F, M/I, F/I), and for several values of `w`:**
  - `w` is a vector that projects the data on the each dimension of the data.
  - `w` is the difference between the mean vectors of the two classes.
  - `w` is the difference between 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()`.**

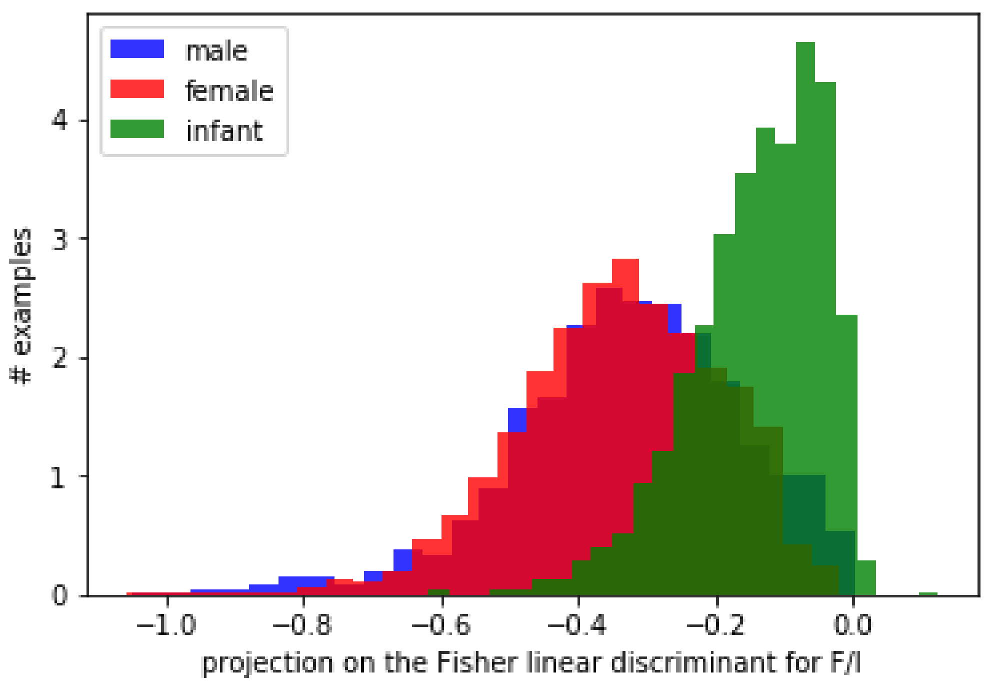
In [55]:
```
# matplot libb inlein blabla
m = abalone.M
f = abalone.F
i = abalone.I

m_m = mean_vector(m)
m_f = mean_vector(f)
m_i = mean_vector(i)

m_q = phi(m)
f_q = phi(f)
i_q = phi(i)

def output(mf, mi, fi):
    return "M/F: {}\n\nM/I: {}\n\nF/I: {}\n\n================================\n".forma
```

```python
# w is a vector that projects the data on the each dimension of the data
w = numpy.array([1, 1, 1, 1, 1, 1, 1])
print(output(J(m, f, w), J(m, i, w), J(f, i, w)))

# w is the difference between the mean vectors of the two classes
w1 = m_m - m_f
w2 = m_m - m_i
w3 = m_f - m_i
print(output(J(m, f, w1), J(m, i, w2), J(f, i, w3)))

# w is the difference between the mean vectors of the two classes (after quadratic ex
m_m_q = mean_vector(m_q)
m_f_q = mean_vector(f_q)
m_i_q = mean_vector(i_q)

w1 = m_m_q - m_f_q
w2 = m_m_q - m_i_q
w3 = m_f_q - m_i_q

print(output(J(m_q, f_q, w1), J(m_q, i_q, w2), J(f_q, i_q, w3)))

# w is the Fisher linear discriminant
w1 = fisher(m, f)
w2 = fisher(m, i)
w3 = fisher(f, i)

print(output(J(m, f, w1), J(m, i, w2), J(f, i, w3)))

# plot

abalone.plot(w1, "projection on the Fisher linear discriminant for M/F")
# for some reason, this one produces an IndexError
#abalone.plot(w2, "projection on the Fisher linear discriminant for M/I")
abalone.plot(w3, "projection on the Fisher linear discriminant for F/I")

print("\n================================\n")

# w is the Fisher linear discriminant (after quadratic expansion of the data)
w1 = fisher(m_q, f_q)
w2 = fisher(m_q, i_q)
w3 = fisher(f_q, i_q)

print(output(J(m_q, f_q, w1), J(m_q, i_q, w2), J(f_q, i_q, w3)))
```

```
M/F: 6.422166732712685e-06

M/I: 0.000698302846221254
```

```
F/I: 0.0010890568974697811

================================

M/F: 5.889571649375587e-06

M/I: 0.0006969087306038125

F/I: 0.0010756345754978094

================================

M/F: 3.2151101929503037e-06

M/I: 0.0005428791118560174

F/I: 0.0008329990403598673

================================

M/F: (8.284345708633506e-06+5.012239372286862e-08j)

M/I: (0.0007288455082176281-5.1627348479417515e-05j)

F/I: (0.0009917427151645668-1.582629150092196e-05j)

================================
```


male
female
infant
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
examples
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
projection on the Fisher linear discriminant for M/F


male
female
infant
4
3
2
1
0
examples
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
projection on the Fisher linear discriminant for F/I

================================

```
M/F: (4.4380242429041596e-06+1.7804104470271078e-06j)

M/I: (0.0004576776980198864+7.764105106391163e-06j)

F/I: (0.0007579063330080531-6.262831189491271e-05j)

================================
```

In [ ]: