Math 156 Project 2

Aatmun Baxi

Problem 1

Part 1

Let $k_a(\mathbf{x}_a, \mathbf{x}'_a) = \Phi_a(\mathbf{x}_a)^T \Phi_a(\mathbf{x}'_a)$, $k_b(\mathbf{x}_b, \mathbf{x}'_b) = \Phi_b(\mathbf{x}_b)^T \Phi_b(\mathbf{x}'_b)$ so that

$$k(\mathbf{x}, \mathbf{x}') = \Phi_a(\mathbf{x}_a)^T \Phi_a(\mathbf{x}'_a) + \Phi_b(\mathbf{x}_b)^T \Phi_b(\mathbf{x}'_b).$$

We claim that k is defined by a nonlinear feature space mapping

$$\psi(\mathbf{x}) = (\Phi_a(\mathbf{x}_a), \Phi_b(\mathbf{x}_b)).$$

where the subscripts a and b define the disjoint subsets of $\mathbf{x}$ as in the problem statement. This feature space mapping is valid as all components of $\mathbf{x}$ are present in the union of the sets $\mathbf{x}_a$, $\mathbf{x}_b$.

Evaluating an inner product as block matrices, we get that

$$\psi(\mathbf{x})^T \psi(\mathbf{x}') = (\Phi_a(\mathbf{x}_a), \Phi_b(\mathbf{x}_b))^T (\Phi_a(\mathbf{x}'_a), \Phi_b(\mathbf{x}'_b)).$$

Note here that inner product is valid since we can transpose the block vector components individually to get valid inner products in the expansion of the above expression. i.e.

$$\psi(\mathbf{x})^T \psi(\mathbf{x}') = (\Phi_a(\mathbf{x}_a), \Phi_b(\mathbf{x}_b))^T (\Phi_a(\mathbf{x}'_a), \Phi_b(\mathbf{x}'_b)) = \Phi_a(\mathbf{x}_a)^T \Phi_a(\mathbf{x}'_a) + \Phi_b(\mathbf{x}_b)^T \Phi_b(\mathbf{x}'_b),$$

which is exactly $k(\mathbf{x}, \mathbf{x}')$.

Part 2

Let $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$, so

$$k(\mathbf{x}, \mathbf{y}) = (x_1 y_1 + x_2 y_2)^2 = x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2.$$

From here it is clear that the mapping

$$\Phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)$$

produces the expression for k when taking $\Phi(\mathbf{x})^T \Phi(\mathbf{y})$.

Problem 2

The log likelihood of the model is

$$F(\mathbf{w}) = \frac{-\beta}{2} \sum_{n=1}^{N} [y(x_n, \mathbf{w}) - t_n]^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi.$$

To maximize we differentiate F with respect to each component of $\mathbf{w}$ $w_1, \ldots, w_M$. To compute this we compute the general partial derivatives of y with respect to each of the components $w_1, \ldots, w_M$.

$$\frac{\partial y}{\partial w_i} = x^i \qquad i = 0, \ldots, M.$$

Therefore

$$\frac{\partial F}{\partial w_i} = -\beta \sum_{n=1}^{N} \left[y\left(x_n, \mathbf{w}\right) - t_n \right] x_n^i \qquad i = 1, \ldots, M.$$

Setting each of these partials equal to 0 we get

$$\begin{aligned}
0 &= -\beta \sum_{n=1}^{N} \left[y\left(x_n, \mathbf{w}\right) - t_n \right] x_n^i \qquad i = 1, \ldots, M \\
\sum_{n=1}^{N} t_n x_n^i &= \sum_{n=1}^{N} x_n^i y\left(x_n, \mathbf{w}\right) \qquad i = 1, \ldots, M \\
\sum_{n=1}^{N} t_n x_n^i &= \sum_{n=1}^{N} x_n^i \sum_{j=0}^{M} w_j x_n^j \qquad i = 1, \ldots, M \\
\sum_{n=1}^{N} t_n x_n^i &= \sum_{n=1}^{N} \sum_{j=0}^{M} w_j x_n^{i+j} \qquad i = 1, \ldots, M \\
\sum_{n=1}^{N} t_n x_n^i &= \sum_{j=0}^{M} \sum_{i=1}^{N} w_j x_n^{i+j} \qquad i = 1, \ldots, M \\
\iff T_i &= \sum_{j=0}^{M} A_{ij} w_j
\end{aligned}$$

Problem 3

It's enough to show that $A\left(A^T A\right)^{-1} A^T$ fixes vectors in $\mathrm{Im}\left(A\right)$ and vectors in $\mathrm{Im}\left(A\right)^{\perp}$ are mapped to 0, as this uniquely characterizes the orthogonal projection of a vector onto the column space of A.

Suppose $v \in \mathrm{Im}\left(A\right)$. Then there is w such that $v = Aw$. So

$$A\left(A^T A\right)^{-1} A^T A w = A \mathbf{1}_n w =: v.$$

On the other hand if $v \in \mathrm{Im}\left(A\right)^{\perp}$, then by the rank nullity theorem, $v \in \ker\left(A^T\right)$, so

$$A\left(A^T A\right)^{-1} A^T v = \mathbf{0}.$$

Problem 4

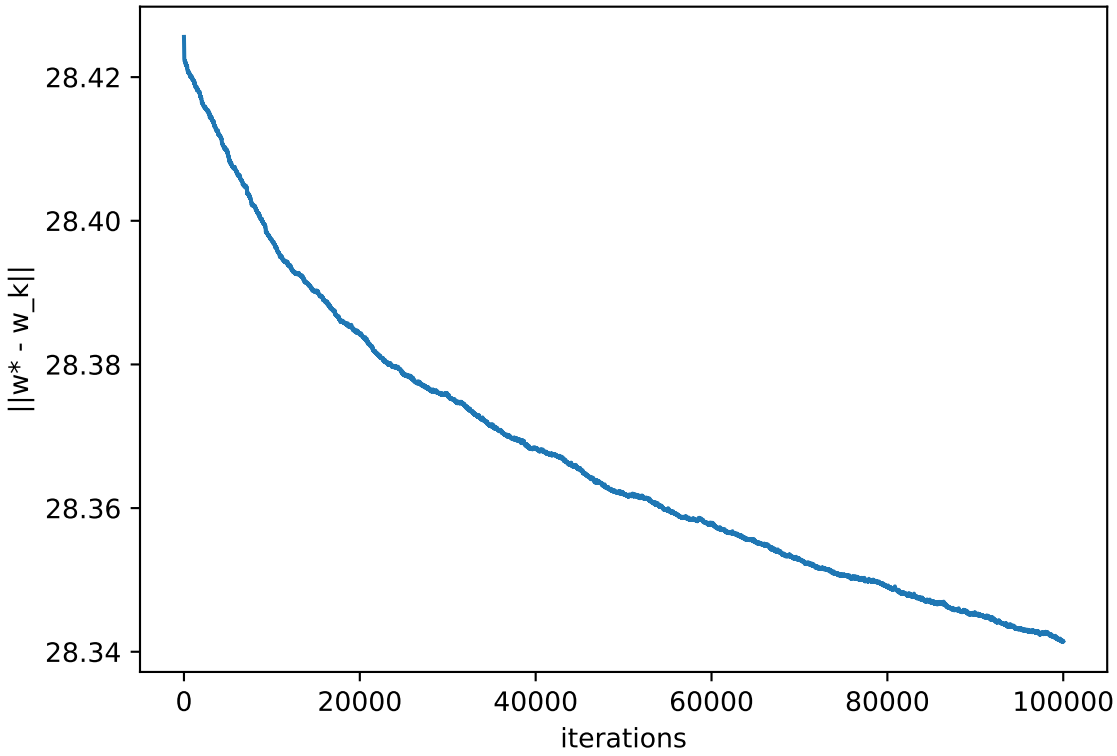
We differentiate in $\mathbf{w}$ and set equal to $\mathbf{0}$, solving for $\mathbf{w}$:

$$
\begin{aligned}
E_D(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}: \\
E_D'(\mathbf{w}) &= \left(\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}\right)\left(-2\left(t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n)\right) \\
\mathbf{0} &= \left(\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}\right)\left(-2\left(t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n)\right) \\
\mathbf{0} &= \left(\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}\right)\left(-2t_n + 2\mathbf{w}^T\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n) \\
\mathbf{0} &= \left(\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}\right)\phi(\mathbf{x}_n) \\
\mathbf{0}^T &= \left(\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}\right)\phi(\mathbf{x}_n)^T \\
\mathbf{0}^T &= \sum_{n=1}^{N} r_n t_n \phi^T(\mathbf{x}_n) - \sum_{n=1}^{N} r_n \mathbf{w}^T\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T \\
\sum_{n=1}^{N} r_n t_n \phi(\mathbf{x}_n)^T &= \sum_{n=1}^{N} r_n \mathbf{w}^T\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T \\
\sum_{n=1}^{N} r_n t_n \phi(\mathbf{x}_n) &= \sum_{n=1}^{N} r_n \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T\mathbf{w} \\
\mathbf{w} &= \left(\sum_{n=1}^{N} r_n \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T\right)^{-1}\left(\sum_{n=1}^{N} r_n t_n \phi(\mathbf{x}_n)\right)
\end{aligned}
$$

Problem 5

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from random import randint

df = pd.read_csv("winequality-red.csv",sep=";",header=0)

#######################################################
##           Closed-form solution                    ##
#######################################################

# Place data in design matrix
X =  np.array( df.iloc[:, :11] )

# Complete linear basis function model
X = np.insert(X , 0, 1,axis=1)
# Target vector
t = np.array( df['quality'])

# Compute Moore-Penrose pseudoinverse of design matrix
```

```
pinv = np.linalg.pinv( X )

# Compute w_star according to closed form solution
w_star = np.matmul( pinv , t)

# Compute average error of closed form
normed = np.subtract( np.matmul( X , w_star ) , t )
error = (np.linalg.norm(normed)**2 ) / 1599



#####################################################
##            Algorithmic Implementation         ##
#####################################################

# Number of algorithm iterations to run
iters = 100_000

w_k = np.zeros(12,float)
errors = np.zeros(iters,float)

# Initial error is just norm of optimal
errors[0] = np.linalg.norm(w_star )

# Iterate algorithm
for i in range(1,iters):

    # Sample data randomly
    n = randint(0,len(X)-1)

    # Compute coefficients of x_n in equation (1)
    eta_term = 1 / ( ( np.linalg.norm( X[n] ))**2 )
    num_coeff = t[n] - np.dot( w_k , X[n])
    coeff = num_coeff * eta_term

    # Update w_k
    w_k = np.add(w_k , (X[n] * coeff) )

    # Load error term
    errors[i] = np.linalg.norm( np.subtract( w_star , w_k))

# Compute average error for algorithm run
normed_alg = np.subtract( np.matmul( X , w_k ) , t )
error_alg = ( np.linalg.norm(normed_alg)**2 ) / (len(X))

# Plot distance over iterates
itvl = np.linspace(0,iters,num=len(errors))
plt.plot(itvl,errors)
plt.xlabel('iterations')
plt.ylabel('||w* - w_k||')

plt.show(block=False)
```




```
print("First 5 entries of w_star")
```

```
## First 5 entries of w_star
```

```
for i in range(0,5):
    print (w_star[i],",")
```

```
## 21.965208449452994 ,
## 0.0249905526716716 ,
## -1.0835902586934316 ,
## -0.18256394841070794 ,
## 0.01633126976547765 ,
```

```
print("Error from closed form:",error )
```

```
## Error from closed form: 0.4167671672214077
```

```
print("Error of algorithm:",error_alg)
```

```
## Error of algorithm: 1.451077265737554
```

Thoughts on Optimal error

Given the quality ratings of the wines are between 0 and 9, an error of < 0.5 seems perfectly reasonable for our value of N, which leads me to believe that this is a fairly good model

The iterative implementation showed wild fluctuations in the possible error of the model after 100,000 iterations, on the other hand. So it is clear that the closed form solution is superior given the other inputs in the algorithm like step size.

Final part

Using equation (1) we have

$$\left(w_n^{(k+1)}\right)^T \mathbf{x}_n = \left(\left(\mathbf{w}^{(k)}\right)^T + \frac{\left(t_n - \left(\mathbf{w}^{(k)}\right)^T\right)\mathbf{x}_n}{\|\mathbf{x}_n\|^2}\mathbf{x}_n^T\right)\mathbf{x}_n$$

$$\left(w_n^{(k+1)}\right)^T \mathbf{x}_n = \left(\mathbf{w}^{(k)}\right)^T \mathbf{x}_n + \frac{\left(t_n - \left(\mathbf{w}^{(k)}\right)^T \mathbf{x}_n\right)}{\|\mathbf{x}_n\|^2}\mathbf{x}_n^T\mathbf{x}_n$$

$$\left(w_n^{(k+1)}\right)^T \mathbf{x}_n = \left(\mathbf{w}^{(k)}\right)^T \mathbf{x}_n + \left(t_n - \left(\mathbf{w}^{(k)}\right)^T \mathbf{x}_n\right)$$

$$\left(w_n^{(k+1)}\right)^T \mathbf{x}_n = t_n.$$