Introduction to Machine Learning
Homework 5 Solutions: Gradient Calculations and Nonlinear Optimization

Prof. Sundeep Rangan

1. (a) First let

$$\mathbf{A} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \cdots & \cdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix},$$

so that if $\mathbf{z} = \mathbf{A}\mathbf{w}$ then,

$$z_i = w_0 + \sum_{j=1}^{d} w_j x_{ij}.$$

Then, if we let

$$g(\mathbf{z}) = \sum_{i=1}^{n} g_i(z_i), \quad g_i(z_i) = \left[y_i - \frac{1}{z_i}\right]^2,$$

we have $J(\mathbf{w}) = g(\mathbf{A}\mathbf{w})$.

(b) The gradient of $g(\mathbf{z})$ is

$$\nabla_{\mathbf{z}} g(\mathbf{z}) = \left[g_1'(z_1), \cdots, g_n'(z_n)\right]^{\mathsf{T}}, \quad g_i'(z_i) = -\frac{1}{z_i^2}\left[y_i - -\frac{1}{z_i}\right].$$

From the forward-backward rule,

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{A}^{\mathsf{T}} \nabla_{\mathbf{z}} g(\mathbf{z}), \quad \mathbf{z} = \mathbf{A}\mathbf{w}.$$

(c) The gradient descent update is

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \nabla_{\mathbf{w}} f(\mathbf{w}^k).$$

(d) Assume we represent the data samples $\mathbf{x}_i$ and $\mathbf{y}_i$ in a python matrix `X` and `y`. Then, we can implement the function and gradient in python as:

```
# Compute function
n = X.shape[0]
A = np.column_stack((np.ones(n), X))
z = A.dot(w)
yerr = y - 1/z
```

```python
J = np.sum(yerr**2)

# Compute gradient
ggrad = -yerr/(z**2)
Jgrad = A.T.dot(ggrad)
```

2. (a) The gradient $\nabla J(\mathbf{w})$,

$$\nabla J(\mathbf{w}) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}\right]^{\mathsf{T}} = [b_1 w_1, b_2 w_2]^{\mathsf{T}}.$$

(b) Set $\nabla J(\mathbf{w}) = 0$, then we get $\mathbf{w}^* = 0$.

(c) We have

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \nabla J(\mathbf{w}^k) \Rightarrow w_i^{k+1} = w_i^k - \alpha b_i w_i^k = \rho_i w_i^k, \tag{1}$$

where $\rho_i = 1 - b_i \alpha$.

(d) In order that $\mathbf{w}^k \to \mathbf{w}^* = 0$, we need $|\rho_i| < 1$ for $i = 1, 2$. Since $b_i > 0$ and $\alpha > 0$, we need

$$|1 - b_i \alpha| < 1 \Rightarrow \alpha < \frac{2}{b_i},$$

for $i = 1, 2$.

(e) For $\alpha = 2/(b_1 + b_2)$, we have

$$\rho_1 = 1 - b_1 \alpha = \frac{b_2 - b_1}{b_1 + b_2}, \quad \rho_2 = 1 - b_2 \alpha = \frac{b_1 - b_2}{b_1 + b_2}.$$

Hence, if we set

$$C = \frac{b_2 - b_1}{b_1 + b_2},$$

we have $|\rho_i| = C$ for $i = 1, 2$. Also, since $\kappa = b_2/b_1$,

$$C = \frac{\kappa - 1}{\kappa + 1}.$$

Now, from (1), $w_i^k = \rho_i^k w_i^0$ so $|w_i^k| = C|w_i^0|$. Therefore,

$$\|\mathbf{w}^k\|^2 = |w_1^k|^2 + |w_2^k|^2 = C^{2k}\left[|w_1^0|^2 + |w_2^0|^2\right] = C^{2k}\|\mathbf{w}^0\|^2.$$

This shows $\|\mathbf{w}^k\| = C\|\mathbf{w}^0\|$.

3. (a) First observe that

$$z_i = \mathbf{x}_i^{\mathsf{T}} \mathbf{P} \mathbf{x}_i = \sum_{j,k} x_{ij} x_{ik} P_{jk} \Rightarrow \frac{\partial z_i}{\partial P_{jk}} = x_{ij} x_{ik}.$$

So $\nabla_{\mathbf{P}} z_i$ is the matrix with elements $x_{ij} x_{ik}$. Therefore,

$$\nabla_{\mathbf{P}} z_i = [x_{ij} x_{ik}]_{jk} = \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}}.$$

(b) By the chain rule,

$$\frac{\partial J}{\partial P_{jk}} = \sum_{i=1}^{n} \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial P_{jk}} = \sum_{i=1}^{n}\left[\frac{1}{y_i} - \frac{1}{z_i}\right]\frac{\partial z_i}{\partial P_{jk}}.$$

Therefore,

$$\nabla_{\mathbf{P}} J = \sum_{i=1}^{n}\left[\frac{1}{y_i} - \frac{1}{z_i}\right]\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}.$$

(c) We can first write this with a for loop as follows:

```
# Compute z
n = X.shape[0]
z = np.zeros(n)
for i in range(n):
    z[i] = X[i,:].dot(P.dot(X[i,:])

# Compute J
J = np.sum(z/y - np.log(z))
g = 1/y - 1/z

# Compute gradient
Jgrad = np.zeros((n,n))
for i in range(n):
    xi = X[i,:]
    Jgrad += g[i]*xi[:,None]*xi[None,:]
```

Note the use of `xi[:,None]*xi[None,:]` to compute $\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}$.

(d) To remove the for-loops, we can use the following code:

```
# Compute z
XP = X.dot(P)
z = np.sum(XP*X, axis=1)

# Compute J
J = np.sum(z/y - np.log(z))
g = 1/y - 1/z

# Compute gradient
GX = g[:,None]*X
Jgrad = X.T.dot(GX)
```

To understand this code, first observe that the i-th row of the matrix `XP` is simply $\mathbf{x}_i^{\mathsf{T}}\mathbf{P}$. Thus, element (i,j) of `XP*X` is

$$(\mathbf{x}_i^{\mathsf{T}}\mathbf{P})_j x_{ij}.$$

When we sum this over `axis=1`, we obtain the sum,

$$\sum_{j=1}^{d}(\mathbf{x}_i^{\mathsf{T}}\mathbf{P})_j x_{ij} = \mathbf{x}_i^{\mathsf{T}}\mathbf{P}\mathbf{x}_i = z_i.$$

Hence, we obtain `z = np.sum(XP*X, axis=1)`. For the gradient, note that element (i,j) of `GX` is $g_i x_{ij}$, where

$$g_i = \frac{1}{y_i} - \frac{1}{z_i}.$$

Therefore, the (k,j) element of `Jgrad` is

$$\sum_{i=1}^{n} x_{ik} g_i x_{ij} = \sum_{i=1}^{n} g_i \left[\mathbf{x}_i \mathbf{x}_i^{\mathsf{T}}\right]_{kj},$$

and hence `Jgrad` is the matrix,

$$\sum_{i=1}^{n} g_i \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} = \nabla_{\mathbf{P}} J(\mathbf{P}).$$

4. (a) Since $\widehat{\mathbf{w}}_2 = \arg\min_{\mathbf{w}_2} J(\mathbf{w}_1, \mathbf{w}_2)$, we have

$$J_1(\mathbf{w}_1) = J(\mathbf{w}_1, \widehat{\mathbf{w}}_2).$$

To take the derivative with respect to $\mathbf{w}_1$ we must remember that $\widehat{\mathbf{w}}_2$ is a function of $\mathbf{w}_1$. Therefore,

$$\begin{aligned} \frac{\partial J_1}{\partial w_{1j}} &= \frac{\partial J(\mathbf{w}_1, \widehat{\mathbf{w}}_2)}{\partial w_{1j}} \\ &= \left.\frac{\partial J(\mathbf{w}_1, \mathbf{w}_2)}{\partial w_{1j}}\right|_{\mathbf{w}_2 = \widehat{\mathbf{w}}_2} + \sum_k \left.\frac{\partial J(\mathbf{w}_1, \mathbf{w}_2)}{\partial w_{2k}}\right|_{\mathbf{w}_2 = \widehat{\mathbf{w}}_2} \frac{\partial w_{2k}}{\partial w_{1j}} \end{aligned} \qquad (2)$$

Now, since $\widehat{\mathbf{w}}_2$ minimizes $J(\mathbf{w}_1, \mathbf{w}_2)$ over $\mathbf{w}_2$, we must have

$$\nabla_{\mathbf{w}_2} J(\mathbf{w}_1, \mathbf{w}_2)|_{\mathbf{w}_2 = \widehat{\mathbf{w}}_2} = 0 \Rightarrow \left.\frac{\partial J(\mathbf{w}_1, \mathbf{w}_2)}{\partial w_{2k}}\right|_{\mathbf{w}_2 = \widehat{\mathbf{w}}_2} = 0,$$

for all k. Therefore, from (2),

$$\frac{\partial J_1}{\partial w_{1j}} = \frac{\partial J(\mathbf{w}_1, \widehat{\mathbf{w}}_2)}{\partial w_{1j}} \Rightarrow \nabla_{\mathbf{w}_1} J_1(\mathbf{w}_1) = \nabla_{\mathbf{w}_1} J(\mathbf{w}_1, \mathbf{w}_2)|_{\mathbf{w}_2 = \widehat{\mathbf{w}}_2}.$$

(b) When $\mathbf{a}$ is fixed, the minimization over $\mathbf{b}$ is a linear least squares problem. To see this, let

$$\hat{y}_i = \sum_{j=1}^{d} b_j e^{-a_j x_i}, \qquad (3)$$

so we can write the loss function as

$$J(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

Thus, $J(\mathbf{a}, \mathbf{b})$ is exactly the RSS. Also, we have $\hat{\mathbf{y}} = \mathbf{A}\mathbf{b}$ where $\mathbf{A}$ is the matrix

$$\mathbf{A} = \begin{bmatrix} e^{-a_1 x_1} & \cdots & e^{-a_d x_1} \\ \vdots & \cdots & \vdots \\ e^{-a_1 x_n} & \cdots & e^{-a_d x_n} \end{bmatrix}.$$

It follows that the optimal $\mathbf{b}$ is given by the least-squares formula

$$\hat{\mathbf{b}} = \arg\min_{\mathbf{b}} J(\mathbf{a}, \mathbf{b}) = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{y}.$$

(c) The partial derivative

$$\begin{aligned} \frac{\partial J(\mathbf{a}, \mathbf{b})}{\partial a_j} &= \sum_{i=1}^{n} \frac{\partial (y_i - \hat{y}_i)^2}{\partial a_j} = -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial a_j} \\ &= 2 \sum_{i=1}^{n} (y_i - \hat{y}_i) b_j x_i e^{-a_j x_i}. \end{aligned} \tag{4}$$

The gradient is

$$\nabla_{\mathbf{a}} J(\mathbf{a}, \mathbf{b}) = \left[\frac{\partial J(\mathbf{a}, \mathbf{b})}{\partial a_1}, \cdots, \frac{\partial J(\mathbf{a}, \mathbf{b})}{\partial a_d} \right]^\top.$$