Problem 1: convex functions

Answer Q1A: Let us derive this from the probability of one data point, and assume IID data:

$$\begin{aligned} P(Y=y|X=\boldsymbol{x}) &= \sigma(\theta^T \boldsymbol{x})^y \cdot [1-\sigma(\theta^T \boldsymbol{x})]^{(1-y)} \\ \implies L(\theta|y) &= \prod_{i=1}^{n} P(Y=y_i|X=\theta_i) \\ &= \prod_{i=1}^{n} p(\boldsymbol{x}_i)^{y_i} \cdot \Big(1-p(\boldsymbol{x}_i)\Big)^{1-y_i} \end{aligned} \tag{1}$$

Now take logs and substitute $p(\boldsymbol{x})$ with the exponent form to give:

$$\begin{aligned} \log L(\theta|y) &= \sum_{i=1}^{n} y_i \left(\frac{1}{1+e^{\theta \boldsymbol{x}_i}}\right) + (1-y_i)\log\left(\frac{e^{-\theta \boldsymbol{x}_i}}{1+e^{-\theta \boldsymbol{x}_i}}\right) \\ \implies &= \sum_{i=1}^{n} y_i \left[\log\left(\frac{1}{1+e^{\theta \boldsymbol{x}_i}}\right) - \log\left(\frac{e^{-\theta \boldsymbol{x}_i}}{1+e^{-\theta \boldsymbol{x}_i}}\right)\right] + \log\left(\frac{e^{-\theta \boldsymbol{x}_i}}{1+e^{-\theta \boldsymbol{x}_i}}\right) \\ \implies &= \sum_{i=1}^{n} y_i \theta_i \boldsymbol{x}_i - \log(1+e^{\theta \boldsymbol{x}_i}) \\ \implies -\log L(\theta|y) &= \sum_{i=1}^{n} -y_i \theta^T \boldsymbol{x} + \log(1+e^{\theta \boldsymbol{x}_i}), \text{ as required.} \quad \square \end{aligned} \tag{2}$$

Answer Q1B: First, let:

$$p_i = p(x_i) = \sigma(\theta^T \boldsymbol{x}_i), \quad a_i = y_i \log(p_i) + (1-y_i)\log(1-p_i), \quad t_i = \theta^T \boldsymbol{x}_i$$

Now we have:

$$\begin{aligned} L(\theta) &= -\sum_{i=1}^{n} \Big(y_i \log p_i + (1-y_i)\log(1-p_i)\Big) \\ \implies \nabla_\theta a_i &= \frac{y_i}{p_i}\big(p_i(1-p_i)\big)\nabla_\theta t_i - \frac{(1-y_i)}{(1-p_i)}\big(p_i(1-p_i)\big)\nabla_\theta t_i \\ \implies \nabla_\theta a_i &= y_i(1-p_i)\boldsymbol{x}_i - (1-y_i)p_i \boldsymbol{x}_i = (y_i - p_i)\boldsymbol{x}_i. \end{aligned} \tag{3}$$

Finally we have:

$$\begin{aligned} \nabla_\theta \log L(\theta|y) &= \sum_{i=1}^{n} (p_i - y_i)\boldsymbol{x}_i \equiv \boldsymbol{X}^T(\boldsymbol{p}-\boldsymbol{y}) \\ \implies -\nabla_\theta \log L(\theta|y) &= -\boldsymbol{X}^T(\boldsymbol{y}-\sigma(\theta)), \text{ as required.} \quad \square \end{aligned} \tag{4}$$

Answer Q1C: From Equation 3 we derived the first gradient which can now be utilized to derive the Hessian of the negative log-likelihood. Thus we can now give the the elements of the Hessian matrix as follows for elements k, j (row index, column index):

$$\begin{aligned}\left[\nabla_\theta^2 a_i\right]_{kj} &= \frac{d}{d\theta_i}\Big[(p_i - y_i)x_i(j)\Big] \\ &= (p_i(1-p_i)) \cdot x_i(k) \cdot x_i(j) \\ &\equiv \left[\boldsymbol{x}_i \boldsymbol{d}_i \boldsymbol{x}_i^T\right]_{kj} \\ \implies \nabla_\theta^2 a_i &= \boldsymbol{x}_i \boldsymbol{d}_i \boldsymbol{x}_i^T\end{aligned} \tag{5}$$

Using the same notation for the diagonal matrix shown in the question this implies:

$$-\nabla_\theta^2 \log L(\theta|y) = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{d}_i \boldsymbol{x}_i^T = \boldsymbol{X}^T \boldsymbol{D}(\theta) \boldsymbol{X}, \text{ as required.} \quad \square \tag{6}$$

Answer Q1D: Using the Equation 6 and the positive definite of our Diagonal matrix $\boldsymbol{D}(\theta)$ we can simply derive the following:

$$\begin{aligned}-\nabla_\theta^2 \log L(\theta|y) &= \boldsymbol{X}^T \boldsymbol{D}(\theta) \boldsymbol{X} \\ \implies \boldsymbol{x}^T \boldsymbol{X}^T \boldsymbol{D}(\theta) \boldsymbol{X} \boldsymbol{x} &= \boldsymbol{y}^T \boldsymbol{D}(\theta) \boldsymbol{y} > 0.\end{aligned} \tag{7}$$

where $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{x}$, and thus if our design matrix $\boldsymbol{X}$ is full-rank then our Hessian is a positive definite matrix by definition.

Answer Q1E: We must show that Newton-Raphson for the two-class logistic regression is equivalent to a form of the IRLS (Iteratively reweighted least squares) algorithm. Newton-Raphson starts with an initial guess of the solution $\hat{\theta}_0$ and recursively updates and iterates until numerical convergence too the solution $\hat{\theta}$. We begin with the l-th iteration:

$$\theta^{(\ell+1)} = \theta^{(\ell)} - (\nabla_\theta^2 \log L(\theta|y)|_{\theta=\theta^{(\ell)}})^{-1} \nabla_\theta \log L(\theta|y)|_{\theta=\theta^{(\ell)}}. \tag{8}$$

Let us denote $\hat{y}_i \in \mathbb{R}^n$ the vector of conditional probabilities:

$$\hat{y}_{(l)} = \begin{bmatrix} \sigma\left(\theta_{(l)}^T \boldsymbol{x}_1\right) \\ \vdots \\ \sigma\left(\theta_{(l)}^T \boldsymbol{x}_n\right) \end{bmatrix}$$

Now let $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ denote the diagonal matrix;

$$\boldsymbol{W} := \begin{bmatrix} \sigma\big(\theta_{(l)}^T \boldsymbol{x}_1\big)\big(1-\sigma\big(\theta_{(l)}^T \boldsymbol{x}_1\big)\big) & 0 & \cdots & 0 \\ 0 & \sigma\big(\theta_{(l)}^T x_2\big)\big(1-\sigma\big(\theta_{(l)}^T x_2\big)\big) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma\big(\theta_{(l)}^T \boldsymbol{x}_n\big)\big(1-\sigma\big(\theta_{(l)}^T \boldsymbol{x}_n\big)\big) \end{bmatrix}$$

Our design matrix $\boldsymbol{X}$ is assumed to be a full-rank. Now, the gradient (or score function) and Hessian can be written as follows:

$$\begin{aligned} \nabla_\theta \log L(\theta|y) &= \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\theta), \\ \nabla_\theta^2 \log L(\theta|y) &= -\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}. \end{aligned} \tag{9}$$

Substitute this into Equation 8 to give the following:

$$\begin{aligned} \theta^{(\ell+1)} &= \theta^{(\ell)} - (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\theta) \\ &= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}(X\hat{\theta}_{(l-1)} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{X}\theta)) \\ &= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{z} \end{aligned} \tag{10}$$

Where the adjusted response is given by:

$$\boldsymbol{z} = \boldsymbol{X}\theta^{(\ell)} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{X}\theta).$$

We now can derive the final IRLS equation:

$$\begin{aligned} \theta^{(\ell+1)} &= \underset{\theta}{\operatorname{argmin}} \frac{1}{2}(\boldsymbol{z} - \boldsymbol{X}\theta^T)^T\boldsymbol{W}(\boldsymbol{z} - \boldsymbol{X}\theta) \\ &= \underset{\theta}{\operatorname{argmin}} \frac{1}{2}(\tilde{\boldsymbol{y}}(\theta^{(\ell)}) - \boldsymbol{X}\theta^T)^T\boldsymbol{W}(\tilde{\boldsymbol{y}}(\theta^{(\ell)}) - \boldsymbol{X}\theta^T), \text{ as required.} \quad \square \end{aligned} \tag{11}$$

Problem 2: convex functions

Definition 2.1 (Convex Function). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain is a convex set and for all $\boldsymbol{x}, \boldsymbol{y}$ in its domain and all $\lambda \in [0,1]$ we have:*

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$$

By definition 2.1 and assuming $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \operatorname{dom}(f)$, we have:

$$\begin{aligned} g(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) &= f(\boldsymbol{A}(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) + \boldsymbol{b}) \\ &= f(\lambda(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) + (1-\lambda)(\boldsymbol{A}\boldsymbol{y} + \boldsymbol{b})) \\ &\leq \lambda f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) + (1-\lambda) f(\boldsymbol{A}\boldsymbol{y} + \boldsymbol{b}) \\ &\equiv \lambda g(\boldsymbol{x}) + (1-\lambda) g(\boldsymbol{y}) \\ \implies g(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) &\leq \lambda g(\boldsymbol{x}) + (1-\lambda) g(\boldsymbol{y}) \text{ as required.} \quad \square \end{aligned} \tag{12}$$

Problem 3: convex functions

Answer Q3: We are assuming $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $\boldsymbol{x}_*$ is a local minimum of f. We must show that this local minimum is a global minimizer.

Let our convex domain be $\mathcal{D}$. Define $\boldsymbol{y}$ to be any arbitrary vector in our domain, which implies $\boldsymbol{y} - \boldsymbol{x}_*$ is well defined. Select an arbitrarily small scalar $\varepsilon > 0$ such that:

$$f(\boldsymbol{x}_*) \leq f(\boldsymbol{x}_* + \varepsilon(\boldsymbol{y} - \boldsymbol{x}_*)) \tag{13}$$

Given convexity of f we have:

$$f(\boldsymbol{x}_* + \varepsilon(\boldsymbol{y} - \boldsymbol{x}_*)) = f(\varepsilon \boldsymbol{y} + (1-\varepsilon)\boldsymbol{x}_*) \leq \varepsilon f(\boldsymbol{y}) + (1-\varepsilon) f(\boldsymbol{x}_*) \tag{14}$$

Now combing our equations we have the final solution:

$$f(\boldsymbol{x}_*) \leq \varepsilon f(\boldsymbol{y}) + (1-\varepsilon) f(\boldsymbol{x}_*) \tag{15}$$

which implies $f(\boldsymbol{x}_*) \leq f(\boldsymbol{y})$ for any arbitrary $\boldsymbol{y}$ or $\boldsymbol{x} \in \mathcal{D}$, thus proving $\boldsymbol{x}_*$ is a global minimum for our convex f as required. $\square$

Problem 4: majorize-minimization algorithms

Definition 4.1 (Majorizer [1]). *Suppose g, h are real valued functions on $\mathcal{X} \in \mathbb{R}^n$. For an optimization problem a surrogate function $h(\boldsymbol{\theta}|\overline{\boldsymbol{\theta}})$ is said to be a majorizer of objective function $g(\boldsymbol{\theta})$ provided*

$$h(\boldsymbol{\theta}|\overline{\boldsymbol{\theta}}) \geq g(\boldsymbol{\theta}), \text{ for all } \boldsymbol{\theta} \in \mathcal{X},$$
$$h(\overline{\boldsymbol{\theta}}|\overline{\boldsymbol{\theta}}) = g(\overline{\boldsymbol{\theta}}).$$

This is almost self-evident by definition and is utilized in cases where we optimize over univariate and separable functions in our Majorize-Minimization (MM) algorithms. We are told $g_i(\boldsymbol{x}|\overline{\boldsymbol{x}}) \succ f_i(\boldsymbol{x})$ where we use $\succ$ to denote majorization. Now consider we have

$$g(\boldsymbol{x}|\overline{\boldsymbol{x}}) = \sum_{i=1}^{n} g_i(\boldsymbol{x}|\overline{\boldsymbol{x}}), \quad f(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(\boldsymbol{x}).$$

For each separable function in $\{g_i\}_{i=1}^n, \{f_i\}_{i=1}^n$ we have:

$$g_i(\boldsymbol{x}|\overline{\boldsymbol{\theta}}) \geq f_i(\boldsymbol{x}), \text{ for all } \boldsymbol{x} \in \mathcal{R}^p \text{ and } g_i(\overline{\boldsymbol{x}}|\overline{\boldsymbol{x}}) = f_i(\overline{\boldsymbol{x}}).$$

By definition, the sum of these separable functions will also satisfy both constraints in Definition 4.1 implying

$$g(\boldsymbol{x}|\overline{\boldsymbol{x}}) = \sum_{i=1}^{n} g_i(\boldsymbol{x}|\overline{\boldsymbol{x}}) \succ f(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(\boldsymbol{x}), \text{ as required.} \quad \square$$

Problem 5: majorize-minimization algorithms

Answer Q5A: The Hessian that we derived above of the negative log-likelihood in Equation 6 is bounded above by $\frac{1}{4}$ which is similar to the quadratic bounding principle for MM algorithms applied in the paper Böhning, 1992 [2]. Now apply the Taylor's expansion of a 2nd order to define the quadratic function for our MM algorithm: $\ell_i(\theta) \simeq \ell_T(\theta)$ to give:

$$\ell_T(\theta) = \ell_i(\overline{\theta}) - (y_i - \sigma(\overline{\theta}^T \boldsymbol{x}_i))\boldsymbol{x}_i^T(\theta - \overline{\theta}) + \frac{1}{2}(\theta - \overline{\theta})^T \boldsymbol{x}_i \boldsymbol{x}_i^T (\theta - \overline{\theta})$$

which implies $g_i(\theta|\overline{\theta})$ is the majorizer of $\ell_i(\theta)$ at $\overline{\theta}$. The MM algorithm then maximizes this quadratic.

Answer Q5B: After using a Taylor series expansion of the second order consider taking the second derivatives. The truncate Taylor expansion of $g(\theta|\overline{\theta})$ gives

$$g(\theta|\overline{\theta}) \approx g(\theta_0) + \nabla g(\theta_0)(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^2 \nabla^2 g(\theta_0)$$

Setting the derivative to zero gives:

$$\frac{\partial g}{\partial \theta} = f(\theta) = \nabla g(\theta_0) + \nabla^2 g(\theta_0)(\theta - \theta_0) = 0$$

Which results in the following update rule:

$$\theta^{(\ell+1)} = \theta^{(\ell)} - \frac{f(\theta_n)}{\nabla f(\theta_n)} = \theta^{(\ell)} - \frac{\nabla g(\theta)}{\nabla^2 g(\theta_n)}. \tag{16}$$

If we pattern match this equation to our solution in Q1E we have the same update rule and thus our update rule for our separable MM algorithm reduces to the same form:

$$\theta^{(\ell+1)} = \underset{\theta}{\operatorname{argmin}} \frac{1}{2}(\tilde{\boldsymbol{y}}(\theta^{(\ell)}) - \boldsymbol{X}\theta^T)^T \boldsymbol{W}(\tilde{\boldsymbol{y}}(\theta^{(\ell)}) - \boldsymbol{X}\theta^T), \text{ as required.} \quad \square \tag{17}$$

Answer Q5C: For our general second order Taylor series of our log-likelihood at $\overline{\theta}$ we have:

$$\ell_T(\theta) = \ell_i(\overline{\theta}) + (\theta - \overline{\theta})\nabla \ell_i(\overline{\theta}) + \frac{1}{2}(\theta - \overline{\theta})^T \nabla^2 \ell_i (\theta - \overline{\theta}). \tag{18}$$

We are assuming our function $f : \mathbb{R}^n \to \mathbb{R}$ is upper-bounded Hessian meaning there exists $\boldsymbol{M} = \mu \boldsymbol{I}$, such that the following holds:

$$\nabla^2 f(\theta) \leq \mu \boldsymbol{I} \implies \nabla^2 f(\theta) \leq \boldsymbol{M}. \tag{19}$$

This implies that $\boldsymbol{M} - \nabla^2 f(\theta)$ is non-negative definite for all θ and $\boldsymbol{M}$ serves as our Hessian which is positive semi-definite. Now we can majorize our function using our quadratic upper bound:

$$\begin{aligned} f_T(\theta) &= f(\bar{\theta}) + (\theta - \bar{\theta})\nabla f(\bar{\theta}) + \frac{1}{2}(\theta - \bar{\theta})^T \nabla^2 f(\theta - \bar{\theta}) \\ &\leq f_i(\bar{\theta}) + (\theta - \bar{\theta})\nabla f(\bar{\theta}) + \frac{1}{2}(\theta - \bar{\theta})^T (\mu \boldsymbol{I})(\theta - \bar{\theta}) \end{aligned} \tag{20}$$

Erdogdu, Hosseinzadeh, and Zhang, [3] prove that for convex f which has an upper-bounded Hessian $\nabla^2 f \leq \mu \boldsymbol{I}$ we have the following identity:

$$\frac{1}{2}(\theta - \bar{\theta})^T (\mu \boldsymbol{I})(\theta - \bar{\theta}) \leq \frac{\mu}{2} \left\| (\theta - \bar{\theta}) \right\|_2^2 .$$

Substitute this into our inequality in Equation 20 to give

$$f_T(\theta) \leq f_i(\bar{\theta}) + (\theta - \bar{\theta})\nabla f(\bar{\theta}) + \frac{\mu}{2} \left\| (\theta - \bar{\theta}) \right\|_2^2 \equiv \ell(\theta). \tag{21}$$

Thus we have shown that $\ell(\theta)$ is the majorizer for $f(\theta)$ with equality holding when $\theta = \bar{\theta}$.

Problem 6: classification

Answer Q6A: Prove unconstrained logistic regression does not have a unique solution:

Answer Q6B: Constrained logistic regression with IRLS algorithm with $\beta^{(0)}$ from problem set 1. Python code below:

```
def logistic_IRLS(X, y, w=None, max_iter=25):
    """ Implement IRLS algorithm for logistic regression

    References:
        [1] Bishop, Pattern Recognition & Machine Learning (2006)

    :X: Design matrix
    :y: Boolean vector of responses
    :w: Initial weight coefficient vector
    :return: w, nll_sequence
    """
    # Set initial weight matrix
    if w is None:
        w = np.array([0]* X.shape[1], dtype='float64')
    y_bar = np.mean(y)
    w_init = math.log(y_bar/(1-y_bar))
    converged = False
    nll_sequence = np.zeros(max_iter)

    for i in range(max_iter):
        h = X.dot(w)
        p = 1/(1+np.exp(-h))
        p_adj = p
        p_adj[p_adj==1.0] = 0.99999999
        nll = -( 1 - y.dot(np.log(1-p_adj))) + y.dot(np.log(p_adj) )
        nll_sequence[i] = nll
        # Update & check for convergence
        if i>1:
            if not converged and abs(nll_sequence[-1]-nll_sequence[-2])<.000001:
                converged = True
                converged_k = i+1
            elif not converged and np.isnan(nll_sequence[-1]):
                converged = True
                converged_k = i+1

        if not converged:
            s = p*(1-p)
            S = np.diag(s)
            arb_small = np.ones_like(s, dtype='float64')*.000001
            z = h + np.divide((y-p), s, out=arb_small, where=s!=0)
            Xt = np.transpose(X)
            XtS = Xt.dot(S)
            XtSX = XtS.dot(X)
            inverse_of_XtSX = np.linalg.inv(XtSX)
            inverse_of_XtSX_Xt = inverse_of_XtSX.dot(Xt)
            inverse_of_XtSX_XtS = inverse_of_XtSX_Xt.dot(S)
            w = inverse_of_XtSX_XtS.dot(z)
        else:
            nll_sequence[i] = nll
```

```
    nll_sequence = np_ffill(nll_sequence,axis=0)
    return w, nll_sequence
```

Answer Q6C: Constrained logistic regression with MM algorithm with $\beta^{(0)}$ from problem set 1. Python code below [4]:

```
import numpy as np
from scipy.optimize import minimize_scalar

def logistic(x):
    """ Compute logistic
    """
    return 1 / (1 + np.exp(-x))

def logistic_loss(w, X, y):
    """ Compute logistic loss
    """
    n = X.shape[0]
    z = np.dot(X, w)
    loss = np.sum(np.log(1 + np.exp(-y * z))) / n
    return loss

def logistic_grad(w, X, y):
    """ Compute score (gradient) for logistic Loss
    """
    n = X.shape[0]
    z = np.dot(X, w)
    g = np.dot(X.T, (-y * logistic(-y * z))) / n
    return g

def logistic_majorize(w, X, y, z, L):
    """ MM (Majorize-Minimization) loistic loss approximation
    for logistic regression
    """
    n = X.shape[0]
    mm_loss = logistic_loss(z, X, y) + np.dot(logistic_grad(z, X, y), w - z) + (L /
                                                 2) * np.sum((w - z)**2)
    return mm_loss

def logistic_MM(X, y, w=None, max_iter=100, tol=1e-4, L=0.1):
    """ Implement IRLS algorithm for logistic regression

    References:
        [1] Bishop, Pattern Recognition & Machine Learning (2006)

    :X: Design matrix
    :y: Boolean vector of responses
    :w: Initial weight coefficient vector
    :return: w, nll_sequence
    """
    # Set initial weight matrix
    if w is None:
        w = np.array([0]* X.shape[1], dtype='float64')
    nll_sequence = np.zeros(max_iter)

    for i in range(max_iter):
        z = w.copy()
```

```
        res = minimize_scalar(lambda x: logistic_majorize(w, X, y, z, L*x), bounds=(
                                                      0, 1), method='bounded')
        w = z - (1/L) * logistic_grad(z, X, y) / res.x
        nll = logistic_loss(w, X, y)
        nll_sequence[i] = nll
        if np.linalg.norm(w - z) < tol:
            break
    return w, nll_sequence
```

Answer Q6D: Plots of negative-log-likelihood for each iteration for IRLS and MM algorithm:

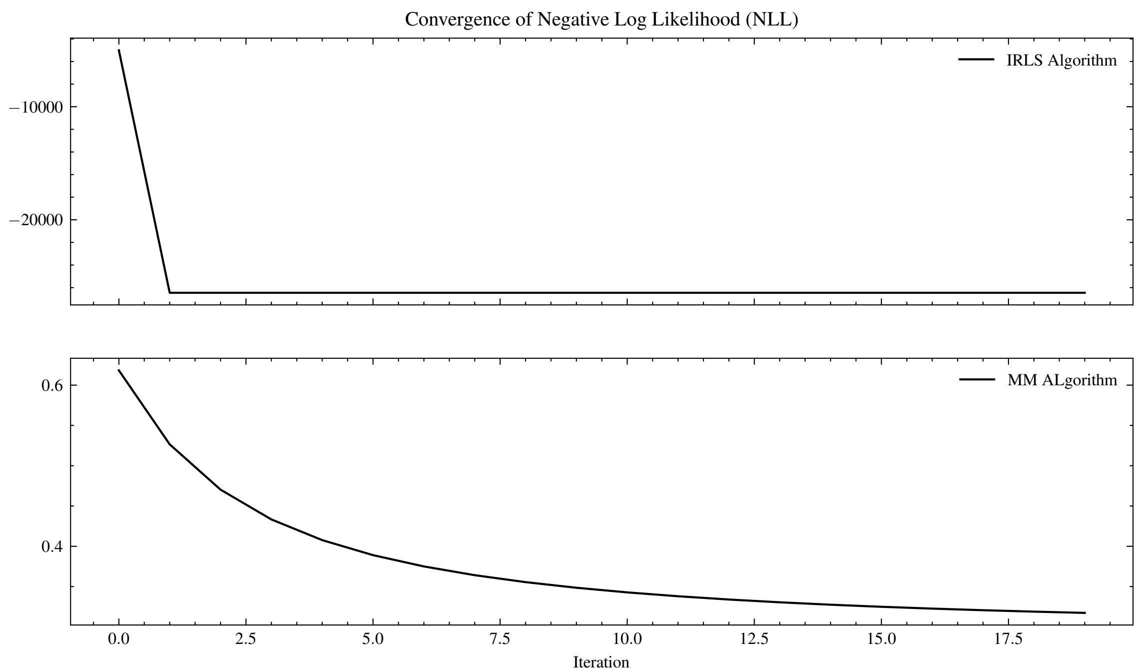

Figure 1: Convergence of NLL for IRLS and MM algorithm

Answer Q6E: Logistic regression solution for IRLS and MM algorithm shown in Fig. 2 below:

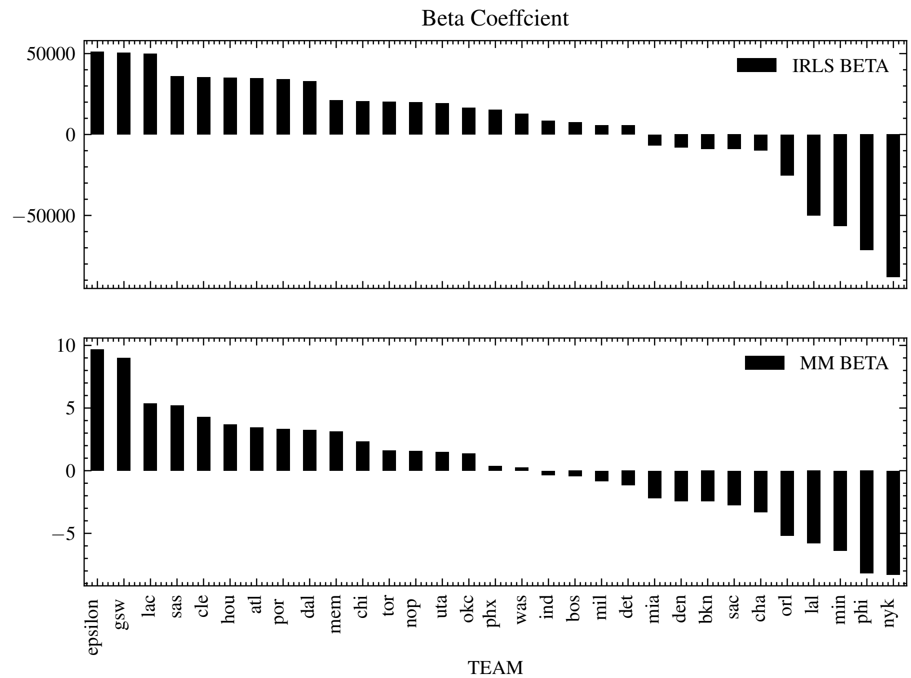

Figure 2: Beta Coefficient Plots for IRLS and MM algorithm

Comparison of rankings obtained here with rankings: The IRLS has a similar convergence to our original solution shown in problem set one, it also has a fast convergence. This corresponds with the theoretical convergence shown in 3 steps due to the matrix becoming singular as shown in Bishop and Nasrabadi, 2006 [5]. Our MM algorithm is essentially producing an approximation by utilizing a proxy function and thus the beta coefficient weights are slightly different, but would likely converge after a greater number of iterations. This is similar to ML training that we would see in industry, it is not sufficient just to have a large number of iterations as we have to have a balance between over-fitting our training data due to the Bias-Variance trade-off.

Comment on skill-coefficient vector compared to final NBA team rankings: Our coefficients seem to fairly accurately capture the top-performing (by wins) and worse performing teams (by wins) in this logistic regression framework. Intuitively this makes sense that our parameters accurately reflect the skill of the teams.

Problem 7: classification

Answer Q7A: Generally, for a multiclass (multinomial) logistic regression with IID data we have:

$$\begin{aligned}
P(\boldsymbol{Y}=0|\boldsymbol{x},\theta) &= \frac{e^{\boldsymbol{x}^T\theta_0}}{1+\sum_i e^{\boldsymbol{x}^T\theta_i}} \\
P(\boldsymbol{Y}=1|\boldsymbol{x},\theta) &= \frac{e^{\boldsymbol{x}^T\theta_0}}{1+\sum_i e^{\boldsymbol{x}^T\theta_i}} \\
&\vdots \\
P(\boldsymbol{Y}=M-1|\boldsymbol{x},\theta) &= \frac{e^{\boldsymbol{x}^T\theta_0}}{1+\sum_i e^{\boldsymbol{x}^T\theta_i}}
\end{aligned} \tag{22}$$

and thus by the multiplication rule we have:

$$P(\boldsymbol{Y}=M|\boldsymbol{x},\theta) = \prod_{i=1}^{M-1}\left(\frac{e^{\boldsymbol{x}^T\theta_i}}{1+\sum_i e^{\boldsymbol{x}^T\theta_j}}\right) \tag{23}$$

Answer Q7B: Not sure - this question is not clear not enough time.

Answer Q7C: Not sure - this question is not clear not enough time.

Answer Q7D: Consider $y = [\boldsymbol{Y}_{i1}, \boldsymbol{Y}_{i2}, \cdots, \boldsymbol{Y}_{i(M-1)},]$ is the multinomial trial for dice roll i with $\boldsymbol{Y}_{ij} = 1$ when the response indicates the face of the roll of the dice. Additionally $\boldsymbol{x}_i$ is the vector of explanatory variables and θ is the parameters vector. For simplicity we first consider the log-likelihood for single observation i as follows:

$$\begin{aligned}
\log L(\theta|(\boldsymbol{y}_i) &= \log\left[\prod_{m=1}^{M}\pi_m\right] \\
&= \sum_{m=1}^{M-1} y_{ij}\log\pi_m + \big(1-\sum_{m=1}^{M-1} y_{ij}\big)\log\left[1-\sum_{m=1}^{M-1}\pi_m\right] \\
&= \sum_{m=1}^{M-1} y_{ij}\log\frac{\pi_m}{1-\sum_{m=1}^{M-1}\pi_m} + \log\left[1-\sum_{m=1}^{M-1}\pi_m\right]
\end{aligned} \tag{24}$$

By definition for our probabilities we have $\pi_M = \sum_{m=1}^{M-1}\pi_m$ and $\sum_{m=1}^{M} y_{nm} = 1$. Also note that we have N independent observations of dice rolls, then the *negative log-likelihood* (NLL) for the M-class multinomial logistic regression is given as follows:

$$\log L(\theta|(\boldsymbol{y}_{i=1}^{n})) = \sum_{i=1}^{n}\sum_{i=1}^{M-1} -y_{im} + \log\left(1+\sum_{m'}^{M-1} e^{\theta_{m'}^T \boldsymbol{x}_i}\right) \text{ as required.} \quad \square \tag{25}$$

Answer Q7E: First define the likelihood function as follows:

$$\begin{aligned}
\log L(\theta|(\boldsymbol{y}_{i=1}^{n})) &= \sum_{m=1}^{M} y_{nm} \log(\pi_{nm}) \\
\implies \nabla_{\theta_m} \log L(\theta|(\boldsymbol{y}_{i=1}^{n})) &= \sum_{m=1}^{M} y_{nm} \frac{1}{\pi_{nm}} \frac{\partial \pi_{nm}}{\partial t_j} \frac{\partial t_j}{\partial \boldsymbol{\theta}_j} \\
&= \sum_{m=1}^{M} y_{nm} y_{nm} \frac{1}{\pi_{nm}} \pi_{nm}(\delta_{mj} - \pi_{nj}) \boldsymbol{x}_n \\
&= (y_{nj} - \pi_{nj}) \boldsymbol{x}_n.
\end{aligned} \tag{26}$$

As before, given $\sum_{m=1}^{M} y_{nm} = 1$ then we have:

$$\implies \nabla_{\theta_m} \log L(\theta|(\boldsymbol{y}_{i=1}^{n})) = -\sum_{i=1}^{n} \boldsymbol{x}_i (y_{im} - \pi_m), \text{ as required.} \quad \square \tag{27}$$

Answer Q7F: We follow a similar derivation to the highly cited paper by Böhning, 1992 [2] and other texts which used the Kroneker product $\otimes$ to simplify the notation.

Definition 7.1 (Kronecker product)**.** *If $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{B}$ is a $p \times q$ matrix, then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is the $pm \times qn$ block matrix:*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \dots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix}$$

Without loss of generality let us reformulate our answer to Q7E as follows for the log-likelihood instead of the negative log-likelihood using Kronecker products:

$$\implies \nabla_{\theta_m} \log L(\theta|(\boldsymbol{y}_{i=1}^{n})) = \sum_{i=1}^{n} (\boldsymbol{\pi}_i - \boldsymbol{y}_i) \otimes \boldsymbol{x}_i. \tag{28}$$

For $j \in 1, 2, \dots, M-1$ we can also derive the following derivative (the Jacobian of the softmax):

$$\frac{\partial \pi_m}{\partial t_j} = \pi_m(\delta_{mj} - \pi_j) \tag{29}$$

Now our Hessian can be shown to be the following:

$$\begin{aligned}
\nabla^2_\theta \log L(\theta|(\boldsymbol{y}^n_{i=1})) &= \sum_{i=1}^{N} \boldsymbol{x}_n \nabla^T_{\theta_k}(\pi_{ij} - y_{ij}) \\
&= \sum_{i=1}^{N} \pi_{ij}(\delta_{kj} - \pi_{ik})\boldsymbol{x}_i\boldsymbol{x}_i^T \\
&\equiv \sum_{i=1}^{N} (\mathrm{diag}(\boldsymbol{\pi}_i) - \boldsymbol{\pi}_i\boldsymbol{\pi}_i^T) \otimes \boldsymbol{x}_i\boldsymbol{x}_i^T \\
\implies \nabla^2_\theta \log L(\theta|(\boldsymbol{y}^n_{i=1})) &= \sum_{i=1}^{N} \boldsymbol{X}_i^T \Sigma_{\boldsymbol{y}_i} \boldsymbol{X}_i, \text{ as required.} \quad \square
\end{aligned} \tag{30}$$

References

[1] X.-D. Zhang. "A matrix algebra approach to artificial intelligence". In: (2020).

[2] D. Böhning. "Multinomial logistic regression algorithm". In: *Annals of the institute of Statistical Mathematics* 44.1 (1992), pp. 197–200.

[3] M. A. Erdogdu, R. Hosseinzadeh, and S. Zhang. "Convergence of Langevin Monte Carlo in chi-squared and Rényi divergence". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 8151–8175.

[4] G. van den Burg. *Algorithms for Multiclass Classification and Regularized Regression*. Tech. rep. 2018.

[5] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.