Statistical Inference and Machine Learning | Homework 1

Team members : Xia Shengzhao, Chica Linares Andrés, Gloria Dal Santo, Loïc Piccot, Elio Ovide Sanchez

October 26, 2020

Question 1

1.1: Answer

(a) Considering $\mathbf{x} = (x_1, x_2, ..., x_d)$ as the data, $y \in \{0, 1\}$ as the label, θ and b as the model parameters, the general form of logistic regression can be defined as below:

$$P(y = 1 \mid \mathbf{x}; \theta) = h_\theta(\mathbf{x}) = \frac{1}{1 + e^{-(\theta^T \mathbf{x} + b)}}$$

$$P(y = 0 \mid \mathbf{x}; \theta) = 1 - h_\theta(\mathbf{x}) = \frac{e^{-(\theta^T \mathbf{x} + b)}}{1 + e^{-(\theta^T \mathbf{x} + b)}}$$

(b) The likelihood function is defined as:

$$L(\theta) = \prod_{\mathbf{i}=1}^{m} P(y_i \mid \mathbf{x}_i; \theta) = \prod_{\mathbf{i}=1}^{m} (\frac{1}{1 + e^{-(\theta^{\mathbf{T}} \mathbf{x}_i + b)}})^{y_i} (1 - \frac{1}{1 + e^{-(\theta^{\mathbf{T}} \mathbf{x}_i + b)}})^{1 - y_i}$$

and the log-likelihood function is defined as:

$$l(\theta) = \log(L(\theta)) = \sum_{i=1}^{m} y_i \cdot \log \frac{1}{1 + e^{-(\theta^T \mathbf{x}_i + b)}} + (1 - y_i) \cdot \log \frac{e^{-(\theta^T \mathbf{x}_i + b)}}{1 + e^{-(\theta^T \mathbf{x}_i + b)}}$$

The maximum value of log-likelihood function can be obtained when its derivative is equal to zero, i.e. $l'(\theta) = 0$. This equation can be solved by Newton's method, which is written as a functional iteration that converges to the root of the function. In each Newton iteration, we update the parameters as:

$$\theta := \theta - (\frac{\partial}{\partial \theta} \frac{\partial l(\theta)}{\partial \theta^T})^{-1} \frac{\partial l(\theta)}{\partial \theta}$$

Let's consider the update process of the j-th parameter θ_j, we compute its first and second derivative as following:

$$\begin{aligned}
\frac{\partial l(\theta)}{\partial \theta_j} &= \sum_{i=1}^{m} (\frac{y_i}{h_\theta(\mathbf{x}_i)} - \frac{1 - y_i}{1 - h_\theta(\mathbf{x}_i)}) \frac{\partial h_\theta(\mathbf{x}_i)}{\partial \theta_j} \\
&= \sum_{i=1}^{m} (\frac{y_i}{h_\theta(\mathbf{x}_i)} - \frac{1 - y_i}{1 - h_\theta(\mathbf{x}_i)}) h_\theta(\mathbf{x}_i)(1 - h_\theta(\mathbf{x}_i)) \frac{\partial \theta^T \mathbf{x}_i}{\partial \theta_j} \\
&= \sum_{i=1}^{m} (y_i - h_\theta(\mathbf{x}_i)) \, \mathbf{x}_{ij} \\
\frac{\partial^2 l(\theta)}{\partial \theta_k \theta_j} &= \frac{\partial}{\partial \theta_k} \frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} h_\theta(\mathbf{x}_i)(h_\theta(\mathbf{x}_i) - 1) \mathbf{x}_{ik} \mathbf{x}_{ij}
\end{aligned}$$

, where $\mathbf{x}_{ij}$ is the j-th element of data $\mathbf{x}_i$.

In matrix form we end up having the following update rule:

$$\theta := \theta - (\mathbf{X^T W X})^{-1}\mathbf{X^T}(\mathbf{y} - \mathbf{h})$$

, where $\mathbf{X}$ is the matrix of features and has dimension $m \times d$, $\mathbf{W}$ is the $m \times m$ diagonal matrix whose i-th diagonal element is $h_\theta(\mathbf{x}_i)(h_\theta(\mathbf{x}_i) - 1)$, $\mathbf{h}$ is the column vector of predicted values $h_\theta(\mathbf{x_i})$ and $\mathbf{y}$ is the column vector of labels. Both vectors have dimension $m \times 1$.

In conclusion, the algorithm can be stated as following:

Algorithm 1: Stochastic Gradient Descent

Input: input parameters: $\mathbf{X}$, $\mathbf{y}$, θ; threshold: ϵ

1 Initialization: $\theta \leftarrow \theta_0$;
2 **for** $\Delta l(\theta) > \epsilon$ **do**
3 $\quad \theta := \theta - (\frac{\partial}{\partial\theta}\frac{\partial l(\theta)}{\partial\theta^T})^{-1}\frac{\partial l(\theta)}{\partial\theta} = \theta - (\mathbf{X^T W X})^{-1}\mathbf{X^T}(\mathbf{y} - \mathbf{h})$
4 **end**

(c) To estimate the classification error we use the Log Loss function, which coincides with the log-likelihood taken with negative sign:

$$H_p = -\sum_{i=1}^{m}\Big(y_i \log\, p_i + (1 - y_i)\log\,(1 - p_i)\Big)$$

The Log-Loss heavily penalizes the classifier when it produces an incorrect classification. For example, if for a particular observation, the classifier assigns a very small probability to the correct class then the corresponding contribution to the Log Loss will be very large. On the other hand, a correct estimation made by the classifier lead to a low contribution to the overall Log Loss. This is why we seek to minimize the Log Loss function, or to maximize the log-likelihood.

(d) As described in the lecture note, there are three main assumptions when logistic regression is used:

- $y|\mathbf{x};\theta \sim$ ExponentialFamily(η), which means given $\mathbf{x}$ and θ, the distribution of y follows some exponential family distribution with parameter η. In this case, the Bernoulli family of distribution for the conditional probability of y given x is assumed.
- Given $\mathbf{x}$, we predict the expected value of $T(y)$ given $\mathbf{x}$. Generally, we will have $T(y) = y$, which means we would like the prediction $h(x)$ output by our learned hypothesis to satisfy $h(x) = E[y \mid x]$.
- The natural parameter η and the inputs x are related linearly: $\eta = \theta^T x$.

1.2: Answer

With l_2 regularization, the log-likelihood function is modified as:

$$l(\theta) = \log(L(\theta)) + \lambda\theta^T\theta$$

Similarly, we consider the update process of the j-th parameter $\theta_j^{(n)}$, its first and second derivative would be changed as below:

$$\frac{\partial l(\theta)}{\partial\theta_j} = \sum_{i=1}^{m}\left(y_i - h_\theta(\mathbf{x}_i)\right)\mathbf{x}_{ij} + 2\lambda\theta_j$$

$$\frac{\partial^2 l(\theta)}{\partial\theta_k\theta_j} = \frac{\partial}{\partial\theta_k}\frac{\partial l(\theta)}{\partial\theta_j} = \sum_{i=1}^{m} h_\theta(\mathbf{x}_i)(h_\theta(\mathbf{x}_i) - 1)\mathbf{x}_{ik}\mathbf{x}_{ij} + 2\lambda\delta_{kj}$$

, where $\delta_{kj} = \begin{cases} 1, j = k \\ 0, j \neq k \end{cases}$ is a dirac delta function.

In matrix form, we have the rule as below:

$$\theta := \theta - (\mathbf{X^T W X} + 2\lambda \mathbf{I})^{-1}(\mathbf{X^T}(\mathbf{y} - \mathbf{h}) + 2\lambda \mathbf{1})$$

, where $\mathbf{X}$ is the $m \times d$ input matrix, $\mathbf{1}$ is an all-ones vector with dimension $d \times 1$, and I is the identity matrix of dimension $d \times d$.

1.3: Answer

Assume that the categorical variables $y \in \{1, 2, ..., K\}$, the general form of multinomial logistic regression is defined as follows:

$$P(y = k \mid \mathbf{x}; \theta) = \frac{e^{\theta_k^T \mathbf{x}}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}}}, \quad k \in \{1, 2..., K-1\}$$
$$P(y = K \mid \mathbf{x}; \theta) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}}}$$

, where the θ_k's are the parameters of the k-th independent binary logistic regression model. The K-th probability $P(y = K \mid \mathbf{x}; \theta)$ is chosen as a pivot.

The likelihood function would be:

$$L(\theta) = \prod_{i=1}^{m} P(y_i \mid \mathbf{x}_i; \theta) = \prod_{i=1}^{m} \prod_{k=1}^{K-1} \left(\frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \right)^{1\{y_i = k\}}$$

where $1\{y = k\} = \begin{cases} 1, & y = k \\ 0, & y \neq k \end{cases}$ denotes the indicator function.

The log-likelihood can be derived as follows:

$$l(\theta) = log(L(\theta)) = \sum_{i=1}^{m} \sum_{k=1}^{K-1} 1\{y_i = k\} \cdot \log \left(\frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \right)$$

To obtain the update algorithm, we first derive the partial derivative of $l(\theta)$ with respect to parameter θ_k. Notice that the elements of the sum over k can take two different values depending on whether we are in the case $y_i = k$ or not:

$$\begin{aligned}
\frac{\partial l(\theta)}{\partial \theta_k} &= \sum_{i=1}^{m} \left[1\{y_i = k\} \left(\mathbf{x}_i \Big(1 - \frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \Big) \right) + 1\{y_i \neq k\} \left(- \mathbf{x}_i \frac{e^{\theta_k^T \mathbf{x_i}}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \right) \right] \\
&= \sum_{i=1}^{m} \mathbf{x}_i \cdot \left(1\{y_i = k\} - \frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \cdot \Big(1\{y_i = k\} + 1\{y_i \neq k\} \Big) \right) \\
&= \sum_{i=1}^{m} \mathbf{x}_i \cdot \left(1\{y_i = k\} - \frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \right) \\
&= \sum_{i=1}^{m} \mathbf{x}_i \cdot \Big(1\{y_i = k\} - p(y_i = k \mid \mathbf{x}_i; \theta) \Big)
\end{aligned}$$

Defining P_{ik} as the probability $P_{ik} = p(y_i = k \mid \mathbf{x}_i; \theta) = \frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}}$, we can construct a matrix $\mathbf{P}$ of dimension $m \times (K-1)$ where the i-th, k-th entry is P_{ik}. The matrix form of the gradient $\nabla_{\hat{\theta}} l(\theta)$ can then be derived as follows:

$$\nabla_{\hat{\theta}} l(\theta) = \tilde{\mathbf{X}}^T (\hat{\mathbf{y}} - \tilde{\mathbf{p}})$$

, where the column vector $\tilde{\mathbf{p}}$ of length $m \cdot (K-1)$ is defined by vertically concatenating the transpose of each row of matrix $\mathbf{P}$, and the vector $\hat{\mathbf{y}} \in \mathbb{R}^{m \cdot (K-1)}$ is defined by concatenating the indicator functions

of each sample. These vectors, together with vector $\hat{\theta}$ of the parameters, are intuitively demonstrated as follows:

$$\hat{\theta} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{K-1} \end{pmatrix} \theta_k = \begin{pmatrix} \theta_{k1} \\ \theta_{k2} \\ \vdots \\ \theta_{kd} \end{pmatrix}; \tilde{\mathbf{p}} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_{K-1} \end{pmatrix} \mathbf{p}_k = \begin{pmatrix} P_{1k} \\ P_{2k} \\ \vdots \\ P_{mk} \end{pmatrix}; \hat{\mathbf{y}} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{K-1} \end{pmatrix} \mathbf{y}_k = \begin{pmatrix} 1\{y_1 = k\} \\ 1\{y_2 = k\} \\ \vdots \\ 1\{y_m = k\} \end{pmatrix}$$

The matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{(m\cdot(K-1))\times(d\cdot(K-1))}$ is a block diagonal matrix whose main-diagonal blocks are the matrix $\mathbf{X} \in \mathbb{R}^{m\times d}$ corresponding to the given features:

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{X} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{X} \end{pmatrix}$$

To obtain the update algorithm according to Newthon's method, we derived the Hessian matrix as following:

If $j \neq k$:

$$\begin{aligned} \frac{\partial^2 l(\theta)}{\partial\theta_j \partial\theta_k} &= \sum_{i=1}^{n} \mathbf{x}_i^2 \frac{e^{\theta_\mathbf{j}^\mathbf{T}\mathbf{x_i}} e^{\theta_\mathbf{k}^\mathbf{T}\mathbf{x_i}}}{\left(1 + \sum_{l=1}^{K-1} e^{\theta_l^T \mathbf{x}_i}\right)^2} \\ &= \sum_{i=1}^{n} \mathbf{x}_i^2 \cdot p(y_i = j \mid \mathbf{x}_i; \theta) \cdot p(y_i = k \mid \mathbf{x}_i; \theta) \end{aligned}$$

If $j = k$:

$$\begin{aligned} \frac{\partial^2 l(\theta)}{\partial\theta_j \partial\theta_k} &= \sum_{i=1}^{n} -\mathbf{x}_i^2 \left(\frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=i}^{K-1} e^{\theta_l^T \mathbf{x}_i}} \left(1 - \frac{e^{\theta_k^T \mathbf{x}_i}}{1 + \sum_{l=i}^{K-1} e^{\theta_l^T \mathbf{x}_i}}\right) \right) \\ &= \sum_{i=1}^{n} -\mathbf{x}_i^2 \cdot p(y_i = k \mid \mathbf{x}_i; \theta)(1 - p(y_i = k \mid \mathbf{x}_i; \theta)) \end{aligned}$$

In matrix form it can be expressed as follows:

$$\mathbf{H} = -\tilde{\mathbf{X}}^T \tilde{\mathbf{W}} \tilde{\mathbf{X}}$$

, where $\tilde{\mathbf{W}} \in \mathbb{R}^{(m\cdot(K-1))\times(m\cdot(K-1))}$ can be sub-divided into diagonal matrices $\mathbf{W}_{k,j} \in \mathbb{R}^{m\times m}$ whose i-th diagonal element is $P_{ik}(\delta_{kj} - P_{ij})$:

$$\tilde{\mathbf{W}} = \begin{pmatrix} \mathbf{W_{1,1}} & \mathbf{W_{1,2}} & \cdots & \mathbf{W_{1,K-1}} \\ \mathbf{W_{2,1}} & \mathbf{W_{2,2}} & \cdots & \mathbf{W_{2,K-1}} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W_{K-1,1}} & \mathbf{W_{K-1,2}} & \cdots & \mathbf{W_{K,K-1}} \end{pmatrix}.$$

Finally, applying the Newton's method we get the following update rule:

$$\hat{\theta} := \hat{\theta} + \left(\tilde{\mathbf{X}}^T \tilde{\mathbf{W}} \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T (\hat{\mathbf{y}} - \tilde{\mathbf{p}})$$

1.4: Answer

The model is implemented by python with the scikit-learn library, and the logistic regression function of the scikit-learn library is called to realize the classification.

In this function, the tolerance for stopping criteria is set at "10^{-4}", the solver is set as "newton-cg" for using the Newton's method. The penalty is set as "none", which means the loss does not contain penalty function. The training process can be illustrated as following:

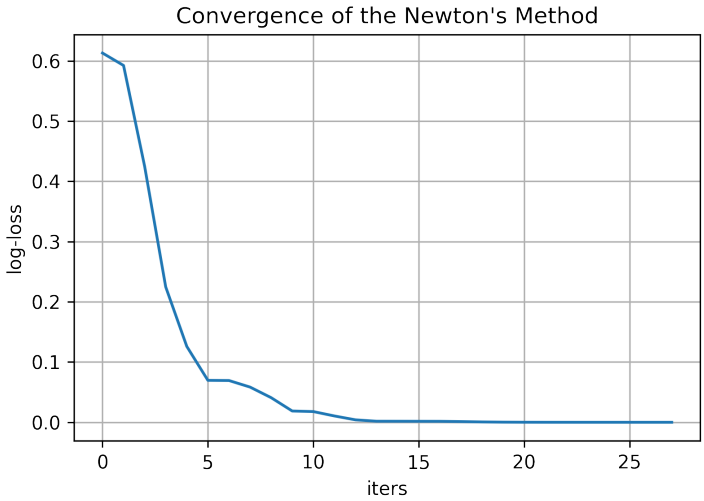

Figure 1: Convergence of the Newton Method

After fitting the model on the test data, we can see:

- Accuracy obtained on the training data is **1.0** with a loss of $\mathbf{9.07 \cdot 10^{-08}}$.
- Accuracy obtained on the testing data is **0.828** with a loss of **3.08**.

1.5: Answer

The gradient descent method is based on the following updating rule:

$$\theta := \theta - \alpha \nabla l(\theta)$$

In our case, the cost function $l(\theta)$ coincides with the negative log-likelihood. As derived above, its gradient is:

$$\frac{\partial l(\theta)}{\partial \theta_k} = \sum_{i=1}^{m} \mathbf{x}_i \cdot \left(1\{y_i = k\} - p(y_i = k \mid \mathbf{x}_i; \theta) \right)$$

And it's matrix form would be:

$$\nabla_{\hat{\theta}} l(\theta) = \tilde{\mathbf{X}}^T (\hat{\mathbf{y}} - \tilde{\mathbf{p}})$$

, where $\mathbf{X}$, $\hat{\theta}$, $\hat{\mathbf{y}}$, $\tilde{p}$ are defined identically as 1.3: Answer.

The algorithm could be descried as follows:

Algorithm 2: Gradient Descent

Input: input parameters: $\tilde{\mathbf{X}}$, $\hat{\mathbf{y}}$, $\mathbf{p}$, θ; learning rate: α

Initialization: $\theta \leftarrow \theta_0$;
while *not converge* **do**
  $\theta := \theta + \alpha \nabla_{\hat{\theta}} l(\theta) = \theta + \alpha \tilde{\mathbf{X}}^T (\hat{\mathbf{y}} - \tilde{\mathbf{p}})$
end

Similar to the previous question, we follow the same methodology and implemented the gradient descent functions on python with scikit-learn library. The training process can be illustrated as following:

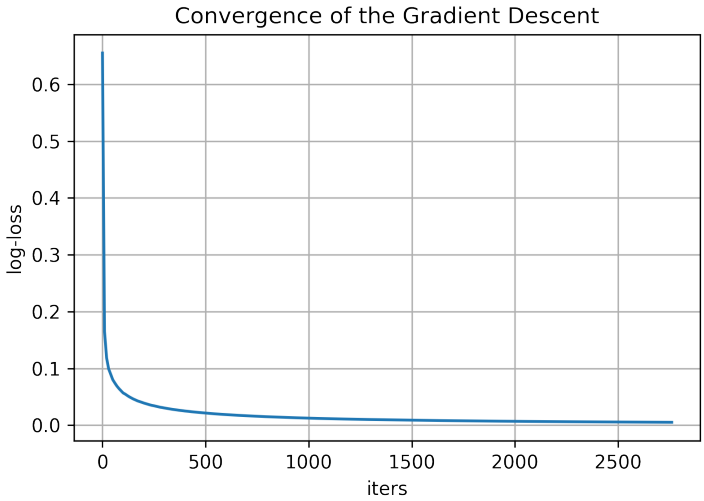

Figure 2: Convergence of the Gradient Descent

After fitting the model on the test data, we can see:

- Accuracy obtained on the training data is **1.0** with a loss of **0.005**.
- Accuracy obtained on the testing data is **0.831** with a loss of **1.086**.

This algorithm is likely to take more steps until convergence since it uses only the information of the first derivative. While the Newton's method, which involves also the second derivative, converges faster and requires fewer iterations.

1.6: Answer

With the back tracking line search method, given a continuously differentiable function, a position $\mathbf{x}$ and a search direction ρ the aim is to find $\alpha > 0$ (known as step size) that reduces $\mathbf{f}(\mathbf{x} + \alpha\rho)$ respect to $\mathbf{f}(\mathbf{x})$. Once an improved starting point is obtained, a new direction is set and another subsequent line search will be performed. The process itself starts with a large estimate of α and it shrinks it iteratively up to the point in which the objective function decrease the expected value.

Applying this method with the log-loss function as our objective function we can decrease significantly the number of iterations, as it can be intuitively deduct from figure 3. However we have encountered several difficulties in implementing this algorithm that prevent us from reporting any meaningful result.

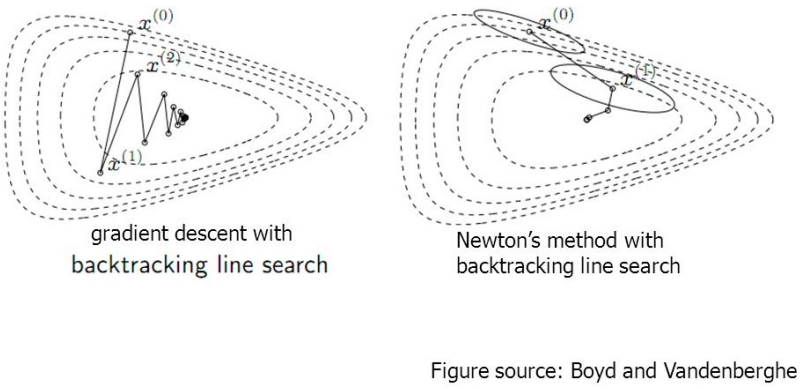

Figure 3: Visual representation of backtracking line search

1.7: Answer

In stochastic gradient descent, we sweep through the training set, for each training example, we compute its gradient and update the parameters according to that. This allows us to speed up the algorithm since we can take a step without computing the summation over the entire set of measurements. The algorithm could be described as follows:

Algorithm 3: Stochastic Gradient Descent

Input: input parameters: $\mathbf{X}$, $\mathbf{y}$, θ; learning rate: α

Initialization: $\theta \leftarrow \theta_0$;
foreach $\mathbf{x}_i$ **do**
  $\theta := \theta - \alpha \frac{\partial l_i(\theta)}{\partial \theta} = \theta - \alpha \cdot \mathbf{x}_i \cdot (1\{y_i = k\} - p(y_i = k \mid \mathbf{x}_i; \theta))$
end

Futher more, it could be adapted to mini-batch stochastic gradient descent, which computes the gradient of a subset (mini-batch) of the training examples. It is illustrated as following:

Algorithm 4: Mini-batch Stochastic Gradient Descent

Input: input parameters: $\mathbf{X}$, $\mathbf{y}$, θ; learning rate: α; batch size: b

Initialization: $\theta \leftarrow \theta_0$;
foreach *mini-batch* **do**
  $\theta := \theta - \alpha \cdot \frac{\sum_{i=1}^{b} \partial l_i(\theta)}{\partial \theta} = \theta - \alpha \cdot \sum_{i=1}^{b} \mathbf{x}_i \cdot (1\{y_i = k\} - p(y_i = k \mid \mathbf{x}_i; \theta))$
end

We fixed the maximum iterations to 200, after fitting the model on the test data, we obtained the following results:

Table 1: Performance of SGD with Different Batch Size

Model	Batch Size	Training Error	Training Accuracy	Testing Error	Testing Accuracy
SGD	1	0.192	0.990	3.733	0.828
	16	0.181	0.995	3.425	0.846
	32	0.373	0.990	3.410	0.862

From the table above, we can see that for what concerns the testing error and testing accuracy, we get better results as we increase the batch size. On the other hand, we found that the batch size 16 achieves the lowest training error. The training accuracy remains at a constant value around 0.9 in either case.

Question 2 : Quadratic Programming

2: Answer

First of all, we need to compute the Hessian of $f(x) = \frac{1}{2}(x_1^2 + 2(1-\epsilon)x_1x_2 + x_2^2)$

$$H = \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 1 & 1-\epsilon \\ 1-\epsilon & 1 \end{pmatrix}$$

Then we can search the eigenvalues by resolving $Hv = \lambda v \rightarrow |H - \lambda I| = 0$

$$det \begin{vmatrix} 1-\lambda & 1-\epsilon \\ 1-\epsilon & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - (1-\epsilon)^2 = 0$$

$$\implies 1 - \lambda = \begin{cases} 1-\epsilon \\ -1+\epsilon \end{cases} \implies \begin{cases} \lambda_1 = \epsilon \\ \lambda_2 = 2-\epsilon \end{cases}$$

The eigenvalues have to be positive so $0 < \epsilon < 2$.
The condition number k is given by the rate between the eigenvalues

$$k = \frac{\lambda_{max}}{\lambda_{min}}$$

where λ_{max} and λ_{min} are attributed depending on the value of ϵ. The two possible cases are the following:

$$\lambda_1 > \lambda_2 \iff \epsilon > 2 - \epsilon \Rightarrow \epsilon > 1 \qquad \Rightarrow \lambda_{max} = \lambda_1 \ , \ \lambda_{min} = \lambda_2$$

$$\lambda_2 > \lambda_1 \iff 2 - \epsilon > \epsilon \Rightarrow \epsilon < 1 \qquad \Rightarrow \lambda_{max} = \lambda_2 \ , \ \lambda_{min} = \lambda_1$$

Then,

$$k = \begin{cases} \frac{2-\epsilon}{\epsilon} & if\ 0 < \epsilon < 1 \\ \frac{\epsilon}{2-\epsilon} & if\ 1 < \epsilon < 2 \end{cases} \quad .$$

Focusing now on the case in which $\epsilon \to 0$, we have that

$$\text{if } \epsilon \to 0 \text{ then } \epsilon < 1 \Rightarrow k = \frac{\lambda_2}{\lambda_1} = \frac{2-\epsilon}{\epsilon}$$

$$\Rightarrow k = \lim_{\epsilon \to 0} \frac{2-\epsilon}{\epsilon} \to \infty$$

When $\epsilon \to 0$ we have $k >> 1$, meaning that the gradient descent converges very slowly.
For optimal cost function, we must choose $\epsilon = 1 \Rightarrow k = 1$.

Question 3: General Linear models for Softmax regression

3.1: Answer

We start by defining the vectors of indicator functions $T : \{1, ..., k\} \to \mathbb{R}^{k-1}$ given by $T(y)_i = 1\{y = i\}$, as follows:

$$T(1) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ ... \\ 0 \\ 0 \end{pmatrix} \ , \ T(2) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ ... \\ 0 \\ 0 \end{pmatrix} \ , \ ... \ , \ T(k-1) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ ... \\ 0 \\ 1 \end{pmatrix}$$

In the following steps we take the advantage of the fact that knowing any $k-1$ parameters we can express ϕ_k as $\phi_k = p(y = k, \phi) = 1 - \sum_{i=1}^{k-1} \phi_i$. We can then obtain the probability of y in the following way:

$$\begin{aligned} p(y; \phi) = \prod_{i=1}^{k} \phi_i^{1\{y=i\}} &= \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_k^{1\{y=k\}} \\ &= \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_k^{1-\sum_{i=1}^{k-1} 1\{y=i\}} \\ &= \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_k^{1-\sum_{i=1}^{k-1} (T(y))_i}. \end{aligned}$$

In the following we use the fact that $p(y;\phi)$ can be expressed in exponential form as $e^{\log(p(y;\phi))}$ and that the logarithm of a product is the sum of the logarithm of each term in the product.

$$\begin{aligned}
p(y;\phi) &= exp(log(\phi_1^{(T(y))_1}\phi_2^{(T(y))_2}...\phi_k^{1-\sum_{i=1}^{k-1}(T(y))_i})) \\
&= exp((T(y))_1 log(\phi_1) + (T(y))_2 log(\phi_2) + ... + (1 - \sum_{i=1}^{k-1}(T(y))_i) log(\phi_k)) \\
&= exp((T(y))_1 log(\phi_1) + (T(y))_2 log(\phi_2) + ... + log(\phi_k)) - (T(y))_1 log(\phi_k) - ... - (T(y))_{k-1} log(\phi_k)) \\
&= exp((T(y))_1 log(\frac{\phi_1}{\phi_k}) + (T(y))_2 log(\frac{\phi_2}{\phi_k}) + ... + (T(y))_{k-1} log(\frac{\phi_{k-1}}{\phi_k}) + log(\phi_k))
\end{aligned}$$

Finally, we can assign the function $a(\eta)$ and $b(\eta)$ in the following way in order to show that the distribution belongs to an exponential family,

$$\begin{aligned}
\eta &= \begin{pmatrix} log(\phi_1/\phi_k) \\ log(\phi_2/\phi_k) \\ ... \\ log(\phi_{k-1}/\phi_k) \end{pmatrix} \\
a(\eta) &= -log(\phi_k) \\
b(y) &= 1
\end{aligned}$$

In this way we obtain $p(y;\phi) = b(y)exp(\eta^T T(y) - a(\eta))$.

3.2: Answer

To find the response function of the model we can start from the expression of the link function and then we invert it to get ϕ_i:

$$\begin{aligned}
\eta_i = \log\frac{\phi_i}{\phi_k} \Leftrightarrow e^{\eta_i} &= \frac{\phi_i}{\phi_k} \\
\phi_i &= \phi_k e^{\eta_i} \\
\sum_{i=1}^{k}\phi_i &= \phi_k \sum_{i=1}^{k} e^{\eta_i}
\end{aligned}$$

We already know that $\sum_{i=1}^{k}\phi_i = 1$, hence $\phi_k = \frac{1}{\sum_{i=1}^{k} e^{\eta_i}}$
We finally get the following response function,

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} .$$

3.3: Answer

Recalling the formula for the likelihood function,

$$L(\theta) = p(y|X;\theta) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta)$$

we can derive the log-likelihood function:

$$\begin{aligned}
log(L(\theta)) &= \log\left(\prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta)\right) \\
&= \sum_{i=1}^{n} \log\Big(p(y^{(i)}|x^{(i)};\theta)\Big) \\
&= \sum_{i=1}^{n} \log\left(\prod_{j=1}^{k}\left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right)^{T(y^{(i)})_j}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k} \log\left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right)^{T(y^{(i)})_j}.
\end{aligned}$$

3.4: **Answer**

To compute the gradient of the log-likelihood we will first find the derivative of the probability ϕ_i:

$$\begin{aligned}
\frac{\partial}{\partial\theta_p}\left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right) &= \frac{\partial}{\partial\theta_p^T x^{(i)}}\left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right)\cdot\frac{\partial}{\partial\theta_p}\left(\theta_p^T x^{(i)}\right) \\
&= \frac{\frac{\partial}{\partial\theta_p^T x^{(i)}}\left(e^{\theta_j^T x^{(i)}}\right)\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}} - e^{\theta_j^T x^{(i)}}\cdot\frac{\partial}{\partial\theta_p^T x^{(i)}}\left(\sum_{l=1}^{k} e^{\theta_j^T x^{(i)}}\right)}{\left(\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}\right)^2}\cdot\frac{\partial}{\partial\theta_p^T}\left(\theta_p^T x^{(i)}\right) \\
&= \frac{1\{p=j\}e^{\theta_j^T x^{(i)}}\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}} - e^{\theta_j^T x^{(i)}}\cdot e^{\theta_p^T x^{(i)}}}{\left(\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}\right)^2}\cdot x^{(i)} \\
&= \left(\frac{1\{p=j\}e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} - \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\cdot\frac{e^{\theta_p^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right)\cdot x^{(i)} \\
&= (1\{p=j\}p(y=j|x;\theta) - p(y=j|x;\theta)\cdot(y=p|x;\theta))\cdot x^{(i)} \\
&= p(y=j|x;\theta)\cdot(1\{p=j\} - p(y=p|x;\theta))\,x^{(i)}
\end{aligned}$$

Moreover,

$$\begin{aligned}
\frac{\partial}{\partial\theta_p}\sum_{j=1}^{k}\log p(y=j|x;\theta) &= \sum_{j=1}^{k}\frac{\partial}{\partial\theta_p}(\log p(y=j|x;\theta)) \\
&= \sum_{j=1}^{k}\frac{1}{p(y=j|x;\theta)}\cdot\frac{\partial p(y=j|x;\theta)}{\partial\theta_p} \\
&= \sum_{j=1}^{k}\frac{1}{p(y=j|x;\theta)}\cdot p(y=j|x;\theta)(1\{p=j\} - p(y=p|x;\theta))x^{(i)} \\
&= \sum_{j=1}^{k}(1\{p=j\} - p(y=p|x;\theta))x^{(i)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial\theta_p}\left(\sum_{j=1}^{k}\log(p(y=j|x;\theta))^{T(y^{(i)})_j}\right) &= \frac{\partial}{\partial\theta_p}\sum_{j=1}^{k} T(y^{(i)})_j\log p(y=j|x;\theta) \\
&= T(y^{(i)})_j\sum_{j=1}^{k}(1\{p=j\} - p(y=p|x;\theta))x^{(i)} \\
&= x^{(i)}\Big(1\{y^{(i)}=p\} - p(y=p|x;\theta)\Big)
\end{aligned}$$

Finally we obtain

$$\begin{aligned}\frac{\partial \ell(\theta)}{\partial \theta_p} &= \frac{\partial}{\partial \theta_p}\left(\sum_{i=1}^{n}\sum_{j=1}^{k}\log(p(y=j|x;\theta))^{T(y^{(i)})_j}\right) \\ &= \sum_{i=1}^{n} x^{(i)}(1\{y^{(i)}=p\} - p(y=p|x;\theta))\end{aligned}$$