EPFL

Statistical Inference and Machine Learning

MGT-448

Homework 1

Logistic Regression, Quadratic Programming & General Linear models for Softmax regression

Students
Student1 Alexis Couturier
Student2 Erick Maraz Zuniga
Student3 Alon Tchelet
Student4 Mustafa Yildirim

Professor
Negar Kiyavash

Abstract

First homework assignment of MGT-448: Statistical Inference and Machine Learning course covering the topics of Logistic Regression, Quadratic Programming, and General Linear models for Softmax regression.

January 31, 2022

1 Logistic Regression

1. (a) The general form of the classifier that corresponds to a logistic regression is the sigmoid function :

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{1}$$

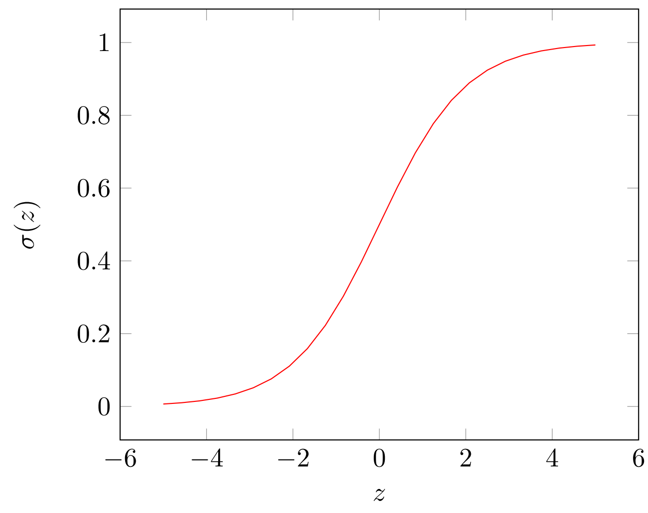


Figure 1: Plot of $\sigma(z)$

In the researcher's case, z will be a weighted sum of her measurements $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$ where θ_0 is the intercept. To simplify notations, we add a 1 in the first position of each measurement vector such that the new weighted sum can be written as $\theta^T x = 1 \cdot \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$. In the following calculations $p(y=1|x,\theta)$ will be referred to as $h_\theta(x)$:

$$h_\theta(x) = P(y=1|x;\theta) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-\theta_0-\theta_1 x_1 ... -\theta_n x_n}} \tag{2}$$

Which can be rewritten as Eq. 3.

$$\log\left(\frac{P(event)}{P(not\ event)}\right) = \log\left(\frac{P(y=1|x;\theta)}{1-P(y=1|x;\theta)}\right) = \theta^T x \tag{3}$$

(b) The likelihood function derived for a binary category Logistic Regression is :

$$L(\theta) = \prod_{i=1}^{n} h_\theta(x_i)^{y_i}(1-h_\theta(x_i))^{1-y_i} \tag{4}$$

We can now define the log-likelihood for the ease of calculation : the derivative is much simpler to compute which will ease the derivation of Newton's Method.

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1-y_i)\log(1-h_\theta(x_i)) \tag{5}$$

Maximizing log-likelihood with Newton's method ones gets the following update rule :

$$\theta_{k+1} = \theta_k + \mathcal{H}^{-1}_{\ell(\theta_k)} \nabla \ell(\theta_k) \tag{6}$$

where $\mathcal{H}_{\ell(\theta)}$ is the hessian of $\ell(\theta)$ and $\nabla\ell(\theta)$ its gradient. Allowing for a maximum number of steps n_{max} and a tolerance tol on the residuals $r_k = ||\theta_{k+1} - \theta_k||_2$, one gets the following algorithm :

```
**Given:** $\nabla\ell(\theta), \mathcal{H}_{\ell(\theta)}, \theta_0, n_{max}, tol$;
$r_0 = tol + 1$;
$k = 0$;
**while** $r_k > tol$ **and** $k < n_{max}$ **do**
    $\theta_{k+1} = \theta_k + \mathcal{H}^{-1}_{\ell(\theta_k)} \nabla\ell(\theta_k)$;
    $r_{k+1} = ||\theta_{k+1} - \theta_k||_2$;
    $k = k + 1$;
**end**
***return*** $\theta_k, r_k, k$;
```

Algorithm 1: Newton's Method

(c) To estimate the classification error of the approach we compare the predictions of the model with the labels. To do so, we first find the weights that gives the maximum likelihood. Then, the results of the classifier equation (Eq. 2) for every x_i are rounded to give a binary result $\in \{0, 1\}$. The results are then compared to the given labels y_i and the percentage of correct result can be calculated.

$$accuracy\% = \frac{correct\ predictions}{overall\ samples} \cdot 100\% = \frac{\sum_{i=1}^{n} h_\theta(x_i) = \begin{cases} 1, & \text{if } \geq 0.5 \\ 0, & \text{if } < 0.5 \end{cases}}{n} \cdot 100\% \tag{7}$$

$$error\% = 100\% - accuracy\% \tag{8}$$

(d) The assumptions made when a logistic regression is used are:

- For the binary case the dependant variable y_i are discrete binary.
- The observations x_i are independent of each other.
- The features do not have linear relationship between them (multicollinearity).
- There is a linear relation between the independent variables x_i and the natural log of the odds, as shown in Eq. 3
- The measurement error ε must be independent.
- There is a large amount of samples such that independent variable is properly represented.

2. By using ℓ_2 regularization the model is less likely to overfit. Introducing a penalty term to the sum of the weights means that the model has to "distribute" its weights optimally, so naturally most of this "ressource" will go to the simple features that explain most of the variance, with complex features getting small or zero weights. Complex models could much better fit the existing samples, but might not predict new samples properly due to overfitting. The regularization is usually parameterized with a λ coefficient to control the complexity of the model.

   Applying a Ridge L2-regularizer, one needs to solve this new problem : $max_\theta\ \ell(\theta) - \lambda||\theta||_2^2$.

   Computing the gradient and hessian of this new term, the update rule in Newton's algorithm becomes : $\theta_{k+1} = \theta_k + (\mathcal{H}_{\ell(\theta_k)} - 2\lambda I)^{-1}(\nabla\ell(\theta_k) - 2\lambda\theta)$

3. Let's now consider a multinomial regression with $J > 2$ discrete categories, D features and N observations. Let x be a $N \times (D+1)$ matrix containing the features and a column of ones for the intercepts, y a $N \times J$ one-hot encoded matrix containing the outcomes and θ a $J \times (D+1)$ matrix containing the regression coefficients.

   The probability distribution is now given by (θ_j is the $j-th$ line of θ) :

$$p(y = j|x^{(i)}; \theta) = \frac{e^{x^{(i)}\theta_j^T}}{1 + \sum_{j=1}^{J-1} e^{x^{(i)}\theta_j^T}}, \text{ if } j < J \tag{9}$$

$$p(y = J|x^{(i)}; \theta) = \frac{1}{1 + \sum_{j=1}^{J-1} e^{x^{(i)}\theta_j^T}}, \tag{10}$$

   The log-likelihood function is given by :

$$\ell(\theta) = log\left(\prod_{i=1}^{N}\prod_{j=1}^{J} p(y = j|x^{(i)}; \theta)^{y_{ij}}\right) = \sum_{i=1}^{N} log\left(\frac{e^{x^{(i)}\theta_{j_i}^T}}{1 + \sum_{j=1}^{J-1} e^{x^{(i)}\theta_{j_i}^T}}\right) \tag{11}$$

$$= \sum_{i=1}^{N}\left[x^{(i)}\theta_{j_i}^T - log\left(1 + \sum_{j=1}^{J-1} e^{x^{(i)}\theta_{j_i}^T}\right)\right] \tag{12}$$

The gradient of the log-likelihood function is given by :

$$\frac{\partial \ell(\theta)}{\partial \theta_{k,j}} = \sum_{i=1}^{N} \left[\delta_{j_i,k} x_j^{(i)} - \frac{x_j^{(i)} e^{x^{(i)}\theta_j^T}}{1 + \sum_{j=1}^{J-1} e^{x^{(i)}\theta_j^T}} \right] = \sum_{i=1}^{N} x_j^{(i)} (\delta_{j_i,k} - p(y = k|x^{(i)};\theta)) \tag{13}$$

The hessian of the log likelihood is given by :

$$\frac{\partial^2 \ell(\theta)}{\partial \theta_{k,j} \partial \theta_{m,n}} = \sum_{i=1}^{N} x_j^{(i)} x_n^{(i)} (-p(y = k|x^{(i)};\theta)\delta_{k,m} + p(y = k|x^{(i)};\theta) p(y = m|x^{(i)};\theta)) \tag{14}$$

$$= -\sum_{i=1}^{N} x_j^{(i)} x_j^{(i)} p(y = k|x^{(i)};\theta)[\delta_{k,m} - p(y = m|x^{(i)};\theta)] \tag{15}$$

The objective is now to find the matrix θ that maximises the log-likelihood function. To do so, one can *augment* the problem and define :

$$\tilde{X} = \begin{pmatrix} X & 0 & 0 & \cdots & 0 \\ 0 & X & 0 & \cdots & 0 \\ \vdots & 0 & & \ddots & \vdots \\ 0 & \cdots & & 0 & X \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} W_{1,1} & W_{1,2} & W_{1,3} & \cdots & W_{1,J} \\ W_{2,1} & W_{2,2} & W_{2,3} & \cdots & \\ \vdots & & \vdots & \ddots & \vdots \\ W_{J,1} & \cdots & W_{J,j} & & W_{J,J} \end{pmatrix} \tag{16}$$

The sub-matrices $W_{k,t}$ are $N \times N$ diagonal where $diag(W_{k,t})_i = p(y = k|x^{(i)};\theta) \cdot (\delta_{k,t} - p(y = t|x^{(i)};\theta))$ and $\delta_{k,t}$ is the Kronecker delta.

Based on the calculations of exercise session 3, one can compute the gradient and Hessian for the augmented problem as follows :

$$\nabla \ell(\theta) = \tilde{X}^T (Y - P) \quad \text{and} \quad \mathcal{H}_{\ell(\theta)} = \tilde{X}^T W \tilde{X} \tag{17}$$

Where vectors Y and P are of length $J \times N$ and formed by vertically stacking the columns of y and the N probabilities $p(y = j|x^{(i)};\theta)$ for $j \in [\![1, J]\!]$.

The algorithm for Newton's method is given by :

```
Given: $\nabla \ell(\theta), \mathcal{H}_{\ell(\theta)}, \theta_0, n_{max}, tol$;
$r_0 = tol + 1$;
$k = 0$;
while $r_k > tol$ and $k < n_{max}$ do
    $\theta_{k+1} = \theta_k + (X^T W X)^{-1} X^T (Y - P)$;
    $r_{k+1} = ||\theta_{k+1} - \theta_k||_2$;
    $k = k + 1$;
end
return $\theta_k, r_k, k$;
```

Algorithm 2: Newton's Method for Multinomial Regression

4. Having the algorithm, one can use it on the researcher's data. To gain a better understanding on how the algorithm performs we can plot the loss at each iterations as well as the accuracy every 10 iterations. At each iteration the algorithm learns the *train data* and uses the updated model to predict both the *test* and *train data.* Theta coefficients for the model are to be found in Fig. 8.

   One can notice that the training accuracy increases very quickly at the beginning, it then slightly decreases after 40 iterations for the *test set* : this is probably due to overfitting.

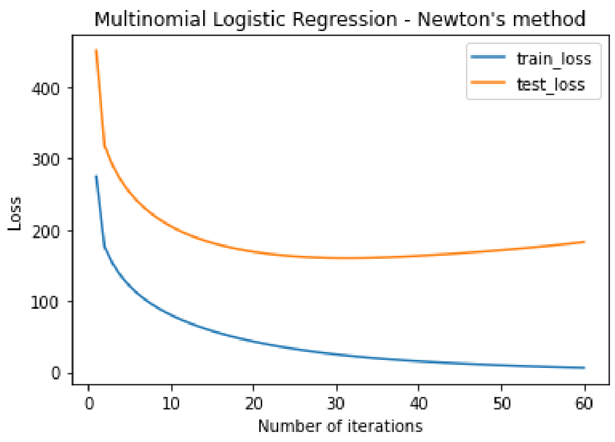


Figure 2: Loss over iterations with $\alpha = 0.001$

```
iter : 10/60 - train_loss = 80.41, train_acc = 0.96, test_loss = 204.68, test_acc = 0.82
iter : 20/60 - train_loss = 43.00, train_acc = 0.98, test_loss = 168.81, test_acc = 0.85
iter : 30/60 - train_loss = 24.98, train_acc = 0.99, test_loss = 159.95, test_acc = 0.86
iter : 40/60 - train_loss = 15.37, train_acc = 1.00, test_loss = 163.05, test_acc = 0.85
iter : 50/60 - train_loss = 9.74, train_acc = 1.00, test_loss = 171.04, test_acc = 0.85
iter : 60/60 - train_loss = 6.07, train_acc = 1.00, test_loss = 182.48, test_acc = 0.84
```

5. We now want to use the gradient descent method to learn the data. The gradient of the log-likelihood function was computed in question 3 and is given by : $\nabla\ell(\theta) = X^T(Y - P)$ and we will use α as the learning rate.

```
**Given:** $\nabla\ell(\theta), \mathcal{H}_{\ell(\theta)}, \theta_0, n_{max}, tol$;
$r_0 = tol + 1$;
$k = 0$;
**while** $r_k > tol$ **and** $k < n_{max}$ **do**
    $\theta_{k+1} = \theta_k + \alpha\nabla\ell(\theta)$;
    $r_{k+1} = ||\theta_{k+1} - \theta_k||_2$;
    $k = k + 1$;
**end**
***return*** $\theta_k, r_k, k$;
```

Algorithm 3: Gradient Descent Method

To gain a better understanding on how the algorithm performs we can plot the loss at each iterations as well as the accuracy every 100 iterations. At each iteration the algorithm learns the *train data* and uses the updated model to predict both the *test* and *train data*. Theta coefficients for the model are to be found in Fig. 8

The main difference between Newton's method and Gradient descent is that the later does not use the Hessian. This can have advantages such as : freeing the algorithm from computing the hessian, being limited due to conditioning problems and not being attracted by saddle points. Newton's method, on the other hand can be faster to converge than Gradient based algorithm.

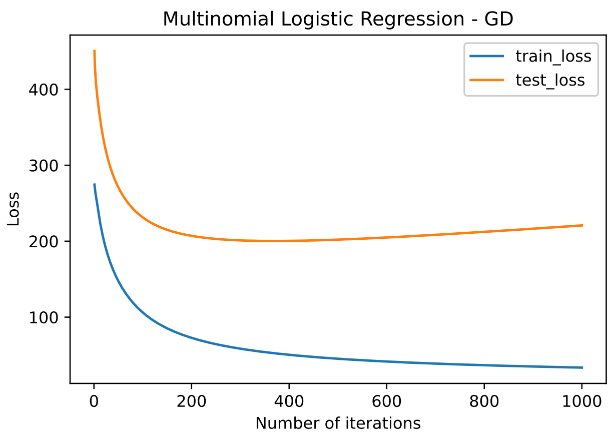


Figure 3: Loss over iterations with $\alpha = 0.0001$

Looking at the graph and the prompt, one can notice that the model makes better predictions every iterations on the *train data* but that the *test loss* increases after 700 iterations. This is due to overfitting.

```
iter : 100/1000 - train_loss = 106.12, train_acc = 0.89, test_loss = 231.33, test_acc = 0.78
iter : 200/1000 - train_loss = 72.75, train_acc = 0.93, test_loss = 207.01, test_acc = 0.81
iter : 300/1000 - train_loss = 58.46, train_acc = 0.93, test_loss = 201.01, test_acc = 0.81
iter : 400/1000 - train_loss = 50.42, train_acc = 0.93, test_loss = 200.38, test_acc = 0.82
iter : 500/1000 - train_loss = 45.22, train_acc = 0.93, test_loss = 202.03, test_acc = 0.82
iter : 600/1000 - train_loss = 41.55, train_acc = 0.93, test_loss = 204.86, test_acc = 0.83
iter : 700/1000 - train_loss = 38.81, train_acc = 0.94, test_loss = 208.37, test_acc = 0.83
iter : 800/1000 - train_loss = 36.69, train_acc = 0.95, test_loss = 212.29, test_acc = 0.82
iter : 900/1000 - train_loss = 34.99, train_acc = 0.95, test_loss = 216.44, test_acc = 0.82
iter : 1000/1000 - train_loss = 33.59, train_acc = 0.95, test_loss = 220.74, test_acc = 0.82
```

6. Implementing the back tracking algorithm for Newton's method ($p_k = \mathcal{H}^{-1}_{\ell(\theta_k)} \nabla \ell(\theta_k)$) resulted in the same losses but with a reduced number of iterations. The Back Tracking line search is implemented as follows (with an optimal choice of ρ and c) :

```
Given: $\bar{\alpha} > 0, \rho, c \in (0,1)$ $\bar{\alpha} = 1, \rho = 0.75, c = 10^{-5}$;
$\alpha \leftarrow \bar{\alpha}$;
$k = 0$;
while $\ell(\theta_k + \alpha p_k) \leq \ell(\theta_k) + c\alpha \nabla \ell(\theta_k)^T p_k$ do
    $\alpha = \rho\alpha$;
    $\theta_{k+1} = \theta_k - \alpha \mathcal{H}^{-1}_{\ell(\theta_k)} \nabla \ell(\theta_k)$;
    $k = k + 1$;
end
return $\theta_k, \alpha$;
```

Algorithm 4: Newton's Method with Back Tracking line Search

A graphical representation helps understand what the code does. Theta coefficients for the model are to be found in Fig. 8

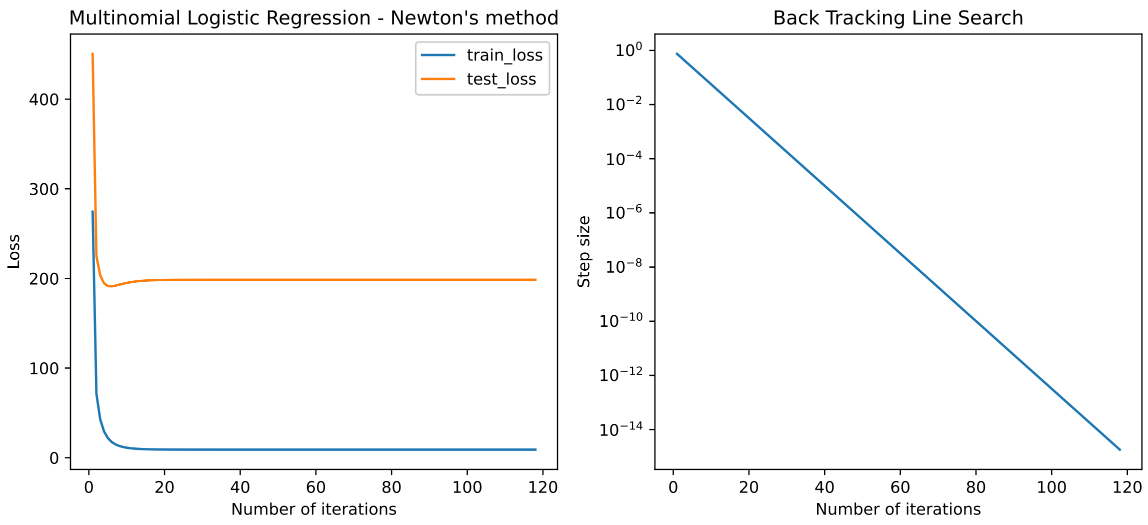


Figure 4: Loss over iterations with α being back tracked

```
iter : 1 - train_loss = 274.49, train_acc = 0.30, test_loss = 450.55, test_acc = 0.42
iter : 2 - train_loss = 250.59, train_acc = 0.50, test_loss = 411.30, test_acc = 0.46
iter : 3 - train_loss = 250.08, train_acc = 0.48, test_loss = 411.62, test_acc = 0.47
```

7. We now want to use the stochastic gradient descent method to learn the data. In this case, to reduce computational cost, the gradient is only computed for a batch of 1, 16, 32 randomly chosen data points instead of the whole data set. We will use α as the learning rate.

```
Given: $\nabla\ell(\theta), \mathcal{H}_{\ell(\theta)}, \theta_0, n_{max}, tol$;
$r_0 = tol + 1$;
$k = 0$;
while $r_k > tol$ and $k < n_{max}$ do
    $\theta_{k+1} = \theta_k + \alpha\nabla_i\ell(\theta)$;
    $r_{k+1} = ||\theta_{k+1} - \theta_k||_2$;
    $k = k + 1$;
end
return $\theta_k, r_k, k$;
```

Algorithm 5: Stochastic Gradient Descent Method

To have a graphical representation of how the stochastic gradient descent performs, a loss plot and prompt for each batch size are represented bellow. Theta coefficients for the models are to be found in Fig. 9.

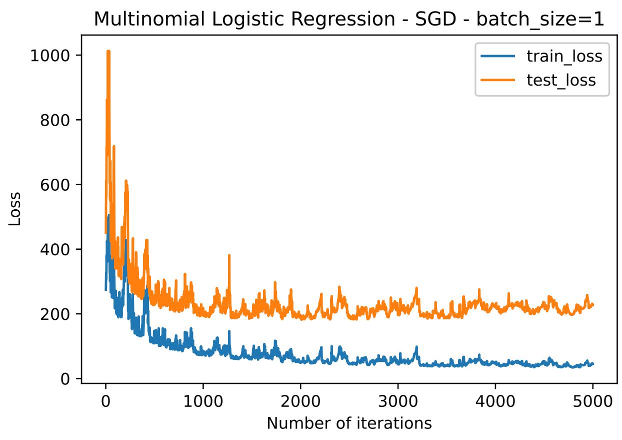


Figure 5: Loss over iterations with $\alpha = 0.003$

```
iter : 500/5000 - train_loss = 118.59, train_acc = 0.82, test_loss = 236.37, test_acc = 0.71
iter : 1000/5000 - train_loss = 79.73, train_acc = 0.92, test_loss = 209.19, test_acc = 0.79
iter : 1500/5000 - train_loss = 63.87, train_acc = 0.93, test_loss = 202.70, test_acc = 0.81
iter : 2000/5000 - train_loss = 75.18, train_acc = 0.91, test_loss = 240.41, test_acc = 0.77
iter : 2500/5000 - train_loss = 49.55, train_acc = 0.95, test_loss = 197.42, test_acc = 0.79
iter : 3000/5000 - train_loss = 41.84, train_acc = 0.95, test_loss = 193.99, test_acc = 0.82
iter : 3500/5000 - train_loss = 38.99, train_acc = 0.95, test_loss = 197.26, test_acc = 0.83
iter : 4000/5000 - train_loss = 49.82, train_acc = 0.92, test_loss = 233.95, test_acc = 0.78
iter : 4500/5000 - train_loss = 40.25, train_acc = 0.94, test_loss = 208.89, test_acc = 0.83
iter : 5000/5000 - train_loss = 34.54, train_acc = 0.96, test_loss = 199.83, test_acc = 0.83
```

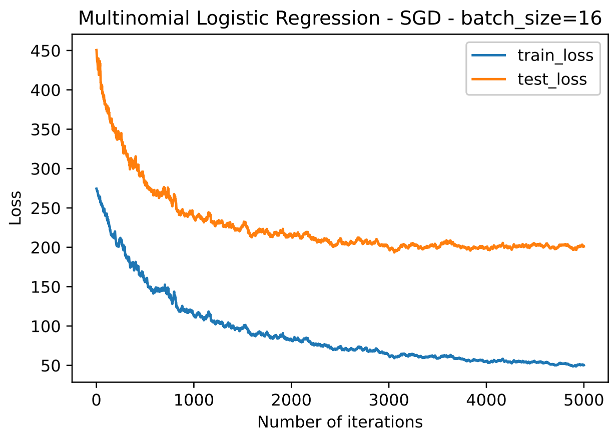


Figure 6: Loss over iterations with $\alpha = 0.0001$

```
iter : 500/5000 - train_loss = 160.98, train_acc = 0.82, test_loss = 283.44, test_acc = 0.76
iter : 1000/5000 - train_loss = 109.40, train_acc = 0.92, test_loss = 234.53, test_acc = 0.77
iter : 1500/5000 - train_loss = 93.51, train_acc = 0.91, test_loss = 219.76, test_acc = 0.78
iter : 2000/5000 - train_loss = 77.55, train_acc = 0.93, test_loss = 204.21, test_acc = 0.81
iter : 2500/5000 - train_loss = 69.40, train_acc = 0.93, test_loss = 203.01, test_acc = 0.81
iter : 3000/5000 - train_loss = 66.24, train_acc = 0.93, test_loss = 205.37, test_acc = 0.80
iter : 3500/5000 - train_loss = 60.23, train_acc = 0.94, test_loss = 202.01, test_acc = 0.81
iter : 4000/5000 - train_loss = 56.82, train_acc = 0.93, test_loss = 200.95, test_acc = 0.81
iter : 4500/5000 - train_loss = 51.32, train_acc = 0.94, test_loss = 197.97, test_acc = 0.82
iter : 5000/5000 - train_loss = 50.18, train_acc = 0.94, test_loss = 200.78, test_acc = 0.82
```

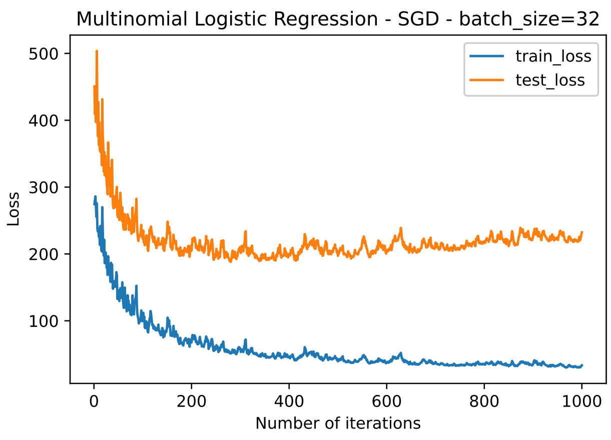


Figure 7: Loss over iterations with $\alpha = 0.0007$

```
iter : 100/1000 - train_loss = 100.69, train_acc = 0.90, test_loss = 230.50, test_acc = 0.77
iter : 200/1000 - train_loss = 67.66, train_acc = 0.93, test_loss = 202.78, test_acc = 0.83
iter : 300/1000 - train_loss = 51.03, train_acc = 0.94, test_loss = 194.49, test_acc = 0.81
iter : 400/1000 - train_loss = 47.73, train_acc = 0.93, test_loss = 203.71, test_acc = 0.82
iter : 500/1000 - train_loss = 45.46, train_acc = 0.94, test_loss = 209.54, test_acc = 0.83
iter : 600/1000 - train_loss = 37.10, train_acc = 0.95, test_loss = 203.57, test_acc = 0.82
iter : 700/1000 - train_loss = 40.58, train_acc = 0.94, test_loss = 217.14, test_acc = 0.84
iter : 800/1000 - train_loss = 34.94, train_acc = 0.96, test_loss = 215.89, test_acc = 0.83
iter : 900/1000 - train_loss = 33.21, train_acc = 0.95, test_loss = 219.27, test_acc = 0.83
iter : 1000/1000 - train_loss = 31.63, train_acc = 0.96, test_loss = 225.03, test_acc = 0.83
```

2 Quadratic Programming

Let $f_\varepsilon : \left\{ \begin{array}{l} \mathbb{R}\times\mathbb{R} \longrightarrow \mathbb{R} \\ (x_1,x_2) \mapsto \frac{1}{2}(x_1^2 + 2(1-\varepsilon)x_1x_2 + x_2^2) \end{array} \right.$ *with* $\varepsilon > 0$. This function is of class $\mathcal{C}^2(\mathbb{R}\times\mathbb{R},\mathbb{R})$ as it is polynomial.

The first partial derivatives are given by :

$$\frac{\partial f_\varepsilon}{\partial x_1}(x_1,x_2) = x_1 + (1-\varepsilon)x_2 \qquad \text{and} \qquad \frac{\partial f_\varepsilon}{\partial x_2}(x_1,x_2) = x_2 + (1-\varepsilon)x_1 \tag{18}$$

The second partial derivatives are given by (note that partial derivatives commute for $\mathcal{C}^2$ functions according to *Schwarz's theorem*) :

$$\frac{\partial^2 f_\varepsilon}{\partial x_1^2}(x_1,x_2) = 1 \qquad \text{and} \qquad \frac{\partial^2 f_\varepsilon}{\partial x_2^2}(x_1,x_2) = 1 \qquad \text{and} \qquad \frac{\partial^2 f_\varepsilon}{\partial x_1 \partial x_2} = \frac{\partial^2 f_\varepsilon}{\partial x_2 \partial x_1} = 1-\varepsilon \tag{19}$$

The hessian of f_ε is : $\mathcal{H}(f_\varepsilon) = \left(\frac{\partial^2 f_\varepsilon}{\partial x_i \partial x_j}\right)_{(i,j)\in[\![1;2]\!]^2} = \begin{pmatrix} 1 & 1-\varepsilon \\ 1-\varepsilon & 1 \end{pmatrix}$ and $\mathcal{H}(f_\varepsilon)^{-1} = \frac{1}{\varepsilon(2-\varepsilon)}\begin{pmatrix} 1 & \varepsilon-1 \\ \varepsilon-1 & 1 \end{pmatrix}$

The conditioning of the Hessian is defined as : $\kappa(\mathcal{H}(f_\varepsilon)) = ||\mathcal{H}(f_\varepsilon)|| \cdot ||\mathcal{H}(f_\varepsilon)^{-1}||$ where $||\cdot||$ is any matrix-norm.

Using the Frobenius norm one finds : $\kappa(\mathcal{H}(f_\varepsilon)) = \frac{2(1+(1-\varepsilon)^2)}{|\varepsilon(2-\varepsilon)|}$

Using $\lambda_{max}/\lambda_{min}$ as a definition for the conditioning and assuming $0 < \varepsilon < 1$, one finds : $\kappa(\mathcal{H}(f_\varepsilon)) = \frac{2-\varepsilon}{\varepsilon}$

One can notice that $\kappa(\mathcal{H}(f_\varepsilon)) \underset{\varepsilon\to 0}{\longrightarrow} +\infty$ in both cases which makes sense as $\mathcal{H}(f_\varepsilon) \underset{\varepsilon\to 0}{\longrightarrow} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \notin \mathcal{GL}_2(\mathbb{R})$.

3 General Linear models for Softmax regression

1) Let's consider the distribution given by : $\begin{cases} \forall i \in [\![1,k]\!], & p(y=i;\phi)=\phi_i \\ \forall i \in [\![1,k]\!], & \phi_i > 0 \\ \sum_{i=1}^{k}\phi_i = 1 \end{cases}$

Then,

$$\begin{aligned} p(y|\phi) = \prod_{i=1}^{k}\phi_i^{y_i} &= exp\Big(\sum_{i=1}^{k} y_i log(\phi_i)\Big) \\ &= exp\Big(\sum_{i=1}^{k-1} y_i log(\phi_i) + \Big(M - \sum_{i=1}^{k-1} y_i\Big) log\Big(1-\sum_{l=1}^{k-1}\phi_l\Big)\Big) \text{ with } M = \sum_{i=1}^{k} y_i \\ &= exp\Big(\sum_{i=1}^{k-1} y_i log\Big(\frac{\phi_i}{1-\sum_{l=1}^{k-1}\phi_l}\Big) y_i + Mlog\Big(1-\sum_{l=1}^{k-1}\phi_l\Big)\Big) \\ &= exp\Big(\Big\langle log\Big(\frac{\phi}{\phi_k}\Big)\Big| y\Big\rangle + Mlog\Big(1-\sum_{l=1}^{k-1}\phi_l\Big)\Big) \end{aligned}$$

Which gives : $p(y|\eta) = b(y)exp(\langle\eta|T(y)\rangle - a(\eta))$ with :

$$\eta_i = log\Big(\frac{\phi_i}{1-\sum_{l=1}^{k-1}\phi_l}\Big) = log\Big(\frac{\phi_i}{\phi_k}\Big), \text{ for } i \in [\![1,k]\!] \tag{20}$$

$$T(y) = y \tag{21}$$

$$a(\eta) = -Mlog\Big(1-\sum_{l=1}^{k-1}\phi_l\Big) = Mlog\Big(\sum_{i=1}^{k} e^{\eta_i}\Big) \tag{22}$$

$$b(y) = 1 \tag{23}$$

$$\tag{24}$$

2) From the previous question, we have :

$$\forall i \in [\![1,k]\!],\ e^{\eta_i} = \frac{\phi_i}{\phi_k} \qquad \text{and} \qquad \sum_{i=1}^{k} e^{\eta_i} = \frac{1}{\phi_k}\underbrace{\sum_{i=1}^{k}\phi_i}_{=1} \tag{25}$$

From which one gets : $\forall i \in [\![1,k]\!],\ \phi_i = \frac{e^{\eta_i}}{\sum_{l=1}^{k} e^{\eta_l}}$

3) Let's consider the distribution given by : $\forall i \in [\![1,k]\!],\ p(y=i|x;\theta) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$

Then,

$$L(\theta;x) = \prod_{l=1}^{n}\prod_{i=1}^{k} p(y=i|x^{(l)};\theta) = \prod_{l=1}^{n}\prod_{i=1}^{k}\frac{e^{\theta_i^T x^{(l)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(l)}}} = \frac{e^{\sum_{l=1}^{n}\sum_{i=1}^{k}\theta_i^T x^{(l)}}}{\prod_{l=1}^{n}\sum_{j=1}^{k} e^{\theta_j^T x^{(l)}}} \tag{26}$$

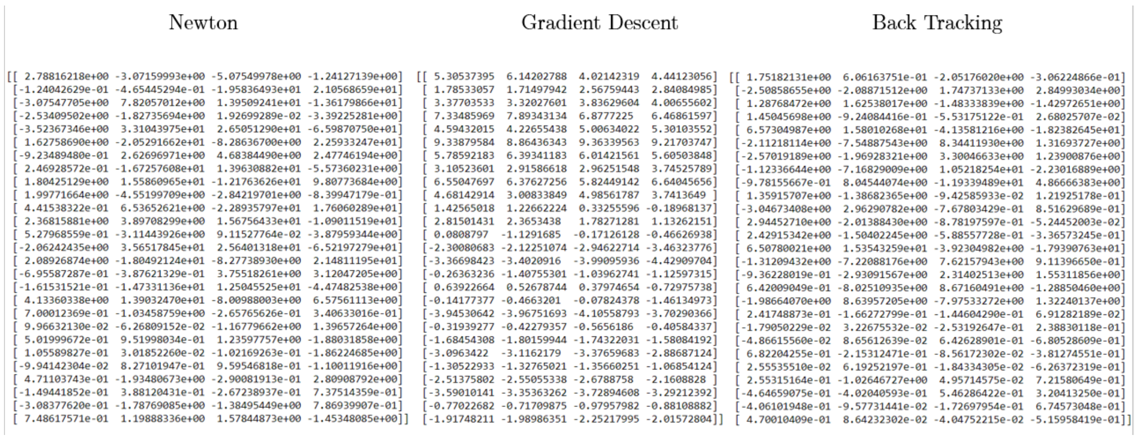
Taking the logarithm, one gets :

$$l(\theta;x) = log(L(\theta)) = \sum_{l=1}^{n}\sum_{i=1}^{k}\theta_i^T x^{(l)} - \sum_{l=1}^{n} log\Big(\sum_{j=1}^{k} e^{\theta_j^T x^{(l)}}\Big) \tag{27}$$

4) The gradient of the log-likelihood function is given by :

$$\nabla l(\theta;x) = \Big(\frac{\partial l(\theta;x)}{\partial \theta_{l,m}}\Big)_{(l,m)\in[\![1,k]\!]^2} = \Big(\sum_{l=1}^{n} x_m^{(l)} - \frac{x_m^{(l)} e^{\theta_{l,m} x_m^{(l)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(l)}}}\Big)_{(l,m)\in[\![1,k]\!]^2} \tag{28}$$

4 APPENDIX : Theta Coefficients for Exercise 1



Newton

```
[[ 2.78816218e+00 -3.07159993e+00 -5.07549978e+00 -1.24127139e+00]
 [-1.24042629e-01 -4.65445294e-01 -1.95836493e+01  2.10568659e+01]
 [-3.07547705e+00  7.82057012e+00  1.39509241e+01 -1.36179866e+01]
 [-2.53409502e+00 -1.82735694e+00  1.92699289e-02 -3.39225281e+00]
 [-3.52367346e+00  3.31043975e+01  2.65051290e+01 -6.59870750e+01]
 [ 1.62758690e+00 -2.05291662e+01 -8.28636700e+00  2.25933247e+01]
 [-9.23489480e-01  2.62696971e+00  4.68384490e+00  2.47746194e+00]
 [ 2.46928572e-01 -1.67257608e+01  1.39630882e+01 -5.57360231e+00]
 [ 1.80425129e+00  1.55860965e+01 -1.21763626e+01  9.80773684e+00]
 [ 1.99771664e+00 -4.55199709e+00 -2.84219701e+00 -8.39947179e-01]
 [ 4.41538322e-01  6.53652621e+00 -2.28935797e+01  1.76060289e+01]
 [ 2.36815881e+00  3.89708299e+00  1.56756433e+01 -1.09011519e+01]
 [ 5.27968559e-01 -3.11443926e+00  9.11527764e-02 -3.87959344e+00]
 [-2.06242435e+00  3.56517845e+01  2.56401318e+01 -6.52197279e+01]
 [ 2.08926874e+00 -1.80492124e+01 -8.27738930e+00  2.14811195e+01]
 [-6.95587287e-01 -3.87621329e-01  3.75518261e+00  3.12047205e+00]
 [-1.61531521e-01 -1.47331136e+01  1.25045525e+01 -4.47482538e+00]
 [ 4.13360338e+00  1.39032470e+01 -8.00988003e+00  6.57561113e+00]
 [ 7.00012369e-01 -1.03458759e+00 -2.65765626e-01  3.40633016e-01]
 [ 9.96632130e-02 -6.26809152e-02 -1.16779662e+00  1.39657264e+00]
 [ 5.01999672e-01  9.51998034e-01  1.23597757e+00 -1.88031858e+00]
 [ 1.05589827e-01  3.01852260e-02 -1.02169263e-01 -1.86224685e+00]
 [-9.94142304e-02  8.27101947e-01  9.59546818e-01 -1.10011916e+00]
 [ 4.71103743e-01 -1.93480673e+00 -2.90081913e-01  2.80908792e+00]
 [-1.49441852e-01  3.88120431e-01 -2.67238937e-01  7.37514359e-01]
 [-3.08377620e-01 -1.78769085e+00 -1.38495449e+00  7.86939907e-01]
 [ 7.48617571e-01  1.19888336e+00  1.57844873e+00 -1.45348085e+00]]
```

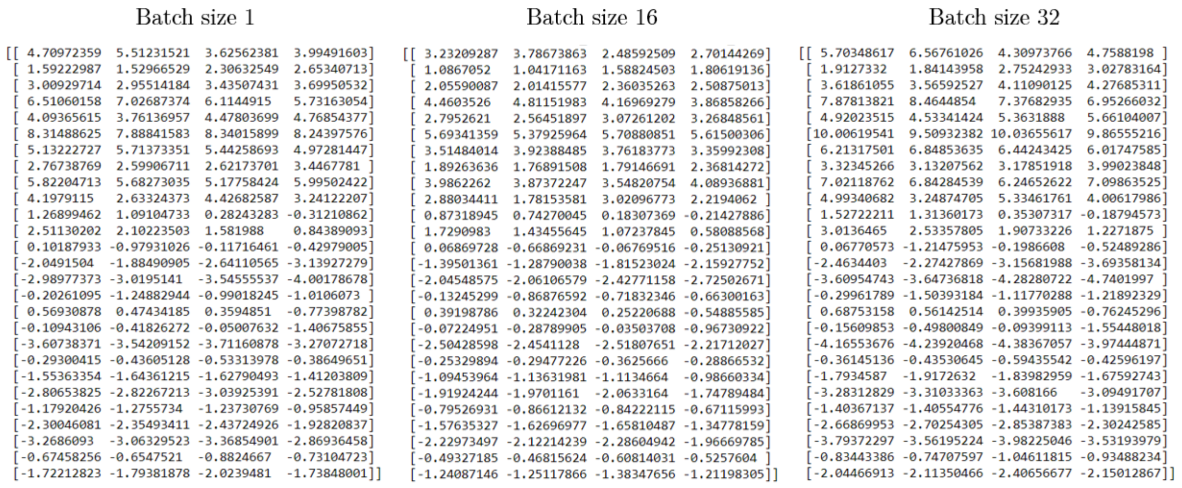
Gradient Descent

```
[[ 5.30537395  6.14202788  4.02142319  4.44123056]
 [ 1.78533057  1.71497942  2.56759443  2.84084985]
 [ 3.37703533  3.32027601  3.83629604  4.00655602]
 [ 7.33485969  7.89343134  6.8777225   6.46861597]
 [ 4.59432015  4.22655438  5.00634022  5.30103552]
 [ 9.33879584  8.86436343  9.36339563  9.21703747]
 [ 5.78592183  6.39341183  6.01421561  5.60503848]
 [ 3.10523601  2.91586618  2.96251548  3.74525789]
 [ 6.55047697  6.37627256  5.82449142  6.64045656]
 [ 4.68142914  3.00833849  4.98561787  3.7413649 ]
 [ 1.42565018  1.22662224  0.33255596 -0.18968137]
 [ 2.81501431  2.3653438   1.78271281  1.13262151]
 [ 0.0808797  -1.1291685  -0.17126128 -0.46626938]
 [-2.30080683 -2.12251074 -2.94622714 -3.46323776]
 [-3.36698423 -3.4020916  -3.99095936 -4.42909704]
 [-0.26363236 -1.40755301 -1.03962741 -1.12597315]
 [ 0.63922664  0.52678744  0.37974654 -0.72975738]
 [-0.14177377 -0.4663201  -0.07824378 -1.46134973]
 [-3.94530642 -3.96751693 -4.10558793 -3.70290366]
 [-0.31939277 -0.42279357 -0.5656186  -0.40584337]
 [-1.68454308 -1.80159944 -1.74322031 -1.58084192]
 [-3.0963422  -3.1162179  -3.37659683 -2.88687124]
 [-1.30522933 -1.32765021 -1.35660251 -1.06854124]
 [-2.51375802 -2.55055338 -2.6788758  -2.1608828 ]
 [-3.59010141 -3.35363262 -3.72894608 -3.29212392]
 [-0.77022682 -0.71709875 -0.97957982 -0.88108882]
 [-1.91748211 -1.98986351 -2.25217995 -2.01572804]]
```

Back Tracking

```
[[ 1.75182131e+00  6.06163751e-01 -2.05176020e+00 -3.06224866e-01]
 [-2.50858655e+00 -2.08871512e+00  1.74737133e+00  2.84993034e+00]
 [ 1.28768472e+00  1.62538017e+00 -1.48333839e+00 -1.42972651e+00]
 [ 1.45045698e+00 -9.24084416e-01 -5.53175122e-01  2.68025707e-02]
 [ 6.57304987e+00  1.58010268e+01 -4.13581216e+00 -1.82382645e+01]
 [-2.11218114e+00 -7.54887543e+00  8.34411930e+00  1.31693727e+00]
 [-2.57019189e+00 -1.96928321e+00  3.30046633e+00  1.23900876e+00]
 [-1.12336644e+00 -7.16829009e+00  1.05218254e+01 -2.23016889e+00]
 [-9.78155667e-01  8.04544074e+00 -1.19339489e+01  4.86666383e+00]
 [ 1.35915707e+00 -1.38682365e+00 -9.42585933e-02  1.21925178e-01]
 [-3.04673408e+00  2.96290782e+00 -7.67803429e-01  8.51629689e-01]
 [ 2.94452710e+00 -2.01388430e+00 -8.78197597e-01 -5.24452003e-02]
 [ 2.42915342e+00 -1.50402245e+00 -5.88557728e-01 -3.36573245e-01]
 [ 6.50780021e+00  1.53543259e+01 -3.92304982e+00 -1.79390763e+01]
 [-1.31209432e+00 -7.22088176e+00  7.62157943e+00  9.11396650e-01]
 [-9.36228019e-01 -2.93091567e+00  2.31402513e+00  1.55311856e+00]
 [ 6.42009049e-01 -8.02510935e+00  8.67160491e+00 -1.28850460e+00]
 [-1.98664070e+00  8.63957205e+00 -7.97533272e+00  1.32240137e+00]
 [ 2.41748873e-01 -1.66272799e-01 -1.44604290e-01  6.91282189e-02]
 [-1.79050229e-02  3.22675532e-02 -2.53192647e-01  2.38830118e-01]
 [-4.86615560e-02  8.65612639e-02  6.42628901e-01 -6.80528609e-01]
 [ 6.82204255e-01 -2.15312471e-01 -8.56172302e-02 -3.81274551e-01]
 [ 2.55535510e-02  6.19252197e-01 -1.84334305e-02 -6.26372319e-01]
 [ 2.55315164e-01 -1.02646727e+00  4.95714575e-02  7.21580649e-01]
 [-4.64659075e-01 -4.02040593e-01  5.46286422e-01  3.20413250e-01]
 [-4.06101948e-01 -9.57731441e-02 -1.72697954e-01  6.74573048e-01]
 [ 4.70010409e-01  8.64232302e-02 -4.04752215e-02 -5.15958419e-01]]
```

Figure 8: Theta coefficients for Newton's Method, Gradient Descent and Back Tracking



Batch size 1

```
[[ 4.70972359  5.51231521  3.62562381  3.99491603]
 [ 1.59222987  1.52966529  2.30632549  2.65340713]
 [ 3.00929714  2.95514184  3.43507431  3.69950532]
 [ 6.51060158  7.02687374  6.1144915   5.73163054]
 [ 4.09365615  3.76136957  4.47803699  4.76854377]
 [ 8.31488625  7.88841583  8.34015899  8.24397576]
 [ 5.13222727  5.71373351  5.44258693  4.97281447]
 [ 2.76738769  2.59906711  2.62173701  3.4467781 ]
 [ 5.82204713  5.68273035  5.17758424  5.99502422]
 [ 4.1979115   2.63324373  4.42682587  3.24122207]
 [ 1.26899462  1.09104733  0.28243283 -0.31210862]
 [ 2.51130202  2.10223503  1.581988    0.84389093]
 [ 0.10187933 -0.97931026 -0.11716461 -0.42979005]
 [-2.0491504  -1.88490905 -2.64110565 -3.13927279]
 [-2.98977373 -3.0195141  -3.54555537 -4.00178678]
 [-0.20261095 -1.24882944 -0.99018245 -1.0106073 ]
 [ 0.56930878  0.47434185  0.3594851  -0.77398782]
 [-0.10943106 -0.41826272 -0.05007632 -1.40675855]
 [-3.60738371 -3.54209152 -3.71160878 -3.27072718]
 [-0.29300415 -0.43605128 -0.53313978 -0.38649651]
 [-1.55363354 -1.64361215 -1.62790493 -1.41203809]
 [-2.80653825 -2.82267213 -3.03925391 -2.52781808]
 [-1.17920426 -1.2755734  -1.23730769 -0.95857449]
 [-2.30046081 -2.35493411 -2.43724926 -1.92820837]
 [-3.2686093  -3.06329523 -3.36854901 -2.86936458]
 [-0.67458256 -0.6547521  -0.8824667  -0.73104723]
 [-1.72212823 -1.79381878 -2.0239481  -1.73848001]]
```

Batch size 16

```
[[ 3.23209287  3.78673863  2.48592509  2.70144269]
 [ 1.0867052   1.04171163  1.58824503  1.80619136]
 [ 2.05590087  2.01415577  2.36035263  2.50875013]
 [ 4.4603526   4.81151983  4.16969279  3.86858266]
 [ 2.7952621   2.56451897  3.07261202  3.26848561]
 [ 5.69341359  5.37925964  5.70880851  5.61500306]
 [ 3.51484014  3.92388485  3.76183773  3.35992308]
 [ 1.89263636  1.76891508  1.79146691  2.36814272]
 [ 3.9862262   3.87372247  3.54820754  4.08936881]
 [ 2.88034411  1.78153581  3.02096773  2.2194062 ]
 [ 0.87318945  0.74270045  0.18307369 -0.21427886]
 [ 1.7290983   1.43455645  1.07237845  0.58088568]
 [ 0.06869728 -0.66869231 -0.06769516 -0.25130921]
 [-1.39501361 -1.28790038 -1.81523024 -2.15927752]
 [-2.04548575 -2.06106579 -2.42771158 -2.72502671]
 [-0.13245299 -0.86876592 -0.71832346 -0.66300163]
 [ 0.39198786  0.32242304  0.25220688 -0.54885585]
 [-0.07224951 -0.28789905 -0.03503708 -0.96730922]
 [-2.50428598 -2.4541128  -2.51807651 -2.21712027]
 [-0.25329894 -0.29477226 -0.3625666  -0.28866532]
 [-1.09453964 -1.13631981 -1.1134664  -0.98660334]
 [-1.91924244 -1.9701161  -2.0633164  -1.74789484]
 [-0.79526931 -0.86612132 -0.84222115 -0.67115993]
 [-1.57635327 -1.62696977 -1.65810487 -1.34778159]
 [-2.22973497 -2.12214239 -2.28604942 -1.96669785]
 [-0.49327185 -0.46815624 -0.60814031 -0.5257604 ]
 [-1.24087146 -1.25117866 -1.38347656 -1.21198305]]
```

Batch size 32

```
[[ 5.70348617  6.56761026  4.30973766  4.7588198 ]
 [ 1.9127332   1.84143958  2.75242933  3.02783164]
 [ 3.61861055  3.56592527  4.11090125  4.27685311]
 [ 7.87813821  8.4644854   7.37682935  6.95266032]
 [ 4.92023515  4.53341424  5.3631888   5.66104007]
 [10.00619541  9.50932382 10.03655617  9.86555216]
 [ 6.21317501  6.84853635  6.44243425  6.01747585]
 [ 3.32345266  3.13207562  3.17851918  3.99023848]
 [ 7.02118762  6.84284539  6.24652622  7.09863525]
 [ 4.99340682  3.24874705  5.33461761  4.00617986]
 [ 1.52722211  1.31360173  0.35307317 -0.18794573]
 [ 3.0136465   2.53357805  1.90733226  1.2271875 ]
 [ 0.06770573 -1.21475953 -0.1986608  -0.52489286]
 [-2.4634403  -2.27427869 -3.15681988 -3.69358134]
 [-3.60954743 -3.64736818 -4.28280722 -4.7401997 ]
 [-0.29961789 -1.50393184 -1.11770288 -1.21892329]
 [ 0.68753158  0.56142514  0.39935905 -0.76245296]
 [-0.15609853 -0.49800849 -0.09399113 -1.55448018]
 [-4.16553676 -4.23920468 -4.38367057 -3.97444871]
 [-0.36145136 -0.43530645 -0.59435542 -0.42596197]
 [-1.7934587  -1.9172632  -1.83982959 -1.67592743]
 [-3.28312829 -3.31033363 -3.608166   -3.09491707]
 [-1.40367137 -1.40554776 -1.44310173 -1.13915845]
 [-2.66869953 -2.70254305 -2.85387383 -2.30242585]
 [-3.79372297 -3.56195224 -3.98225046 -3.53193979]
 [-0.83443386 -0.74707597 -1.04611815 -0.93488234]
 [-2.04466913 -2.11350466 -2.40656677 -2.15012867]]
```

Figure 9: Theta coefficients for Stochastic Gradient Descent