MGT-448 Homework 1

Bohan Wang, Jingwei Chen, Ke Wang, Menghe Jin, Xinghai Wang

24 Octobor, 2020

1 Logistic Regression

1.1

(a)The general form of the classifier that corresponds to LR is given by:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \tag{1}$$

$$P(y=1 \mid x;\theta) = h_\theta(x), P(y=0 \mid x;\theta) = 1 - h_\theta(x) \tag{2}$$

$$P(y \mid x;\theta) = (h_\theta(x))^y * (1-h_\theta(x))^{1-y} \tag{3}$$

(b)The likelihood function is given by:

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)};\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)})^{y^{(i)}} * (1-h_\theta(x^{(i)}))^{1-y^{(i)}} \tag{4}$$

And we can get the log-likelihood function, which is

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right) \tag{5}$$

The Newton's Method is

$$\theta = \theta - \frac{f(\theta)}{f'(\theta)} \tag{6}$$

In order to evaluate the maximum likelihood estimate of θ, we have $f'(\theta)=0$, so θ can be given by:

$$\theta = \theta - H^{-1}\frac{\partial l(\theta)}{\partial \theta} \tag{7}$$

$$H_{ij} = \frac{\partial^2 l(\theta)}{\partial\theta_i \partial\theta_j} \tag{8}$$

(c)Classification error is given by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \tag{9}$$

(d) 1)The dependent variables should be binary.
2)The independent variable is linear
3)There are a large number of samples.

1.2

When we use l_2 regularization, the likelihood function is given by:

$$\begin{aligned} L(\theta) &= P(y \mid x, \theta) P(\theta) \\ &= \prod_{i=1}^{N} p(x_i)^{y_i} t(1-p(x_i))^{1-y_i} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{\theta_j^2}{2\sigma^2}) \\ &= \prod_{i=1}^{N} p(x_i)^{y_i} (1-p(x_i))^{1-y_i} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{\theta^T\theta}{2\sigma^2}) \end{aligned} \tag{10}$$

And the log-likelihood function is given by:

$$l(\theta) = \sum_i \left[y_i \ln p\left(x_i\right) + \left(1-y_i\right) \ln\left(1-p\left(x_i\right)\right)\right] + \frac{1}{2\sigma^2}\theta^T\theta \tag{11}$$

So the The Newton's Method is given by:

$$\theta = \theta - H^{-1}\frac{\partial l(\theta)}{\partial \theta} + \frac{1}{2\sigma^2}\theta \tag{12}$$

1.3

The general form of the classifier is given by:

$$P(y^{(i)} = j \mid x^{(i)}; \theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \tag{13}$$

The log-likelihood function is given by:

$$l(\theta) = \sum_{i=1}^{m}\sum_{j=1}^{k}(1\{y^{(i)} = j\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}) \tag{14}$$

The gradient of log-likelihood function is given by:

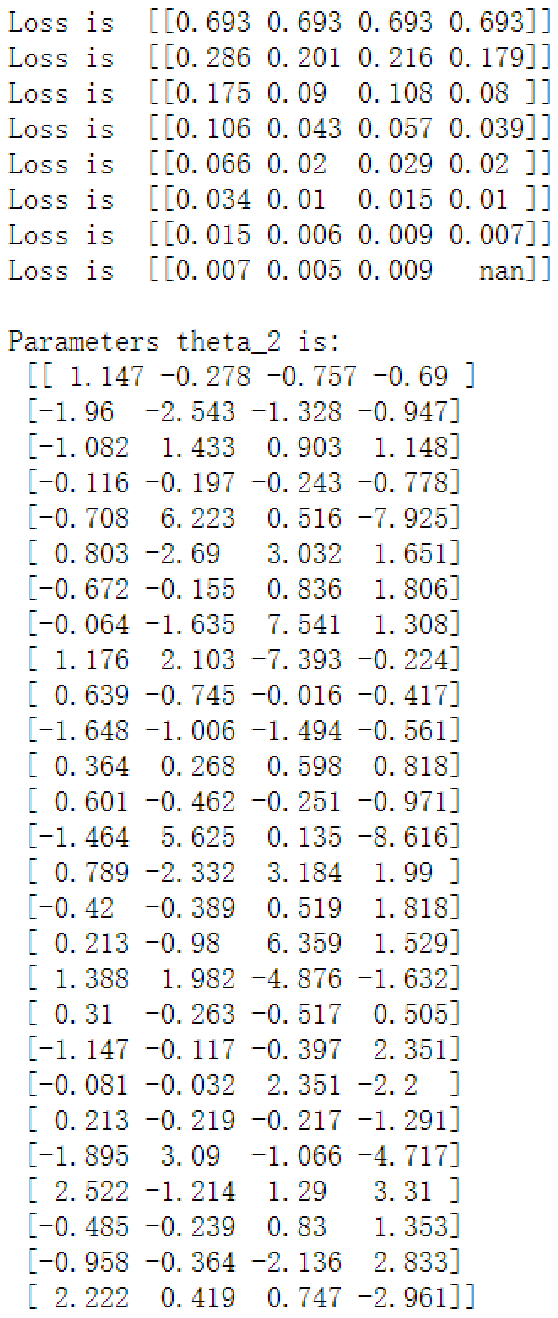
$$\nabla_{\theta_j} l(\theta) = \sum_{i=1}^{m}[x^{(i)}(1\{y^{(i)} = j\} - p(y^{(i)} = j \mid x^{(i)}; \theta))] \tag{15}$$

And our algorithm is:

$$\theta_j := \theta_j - \alpha\nabla_{\theta_j} l(\theta), \forall j = 1, \ldots, k \tag{16}$$

1.4

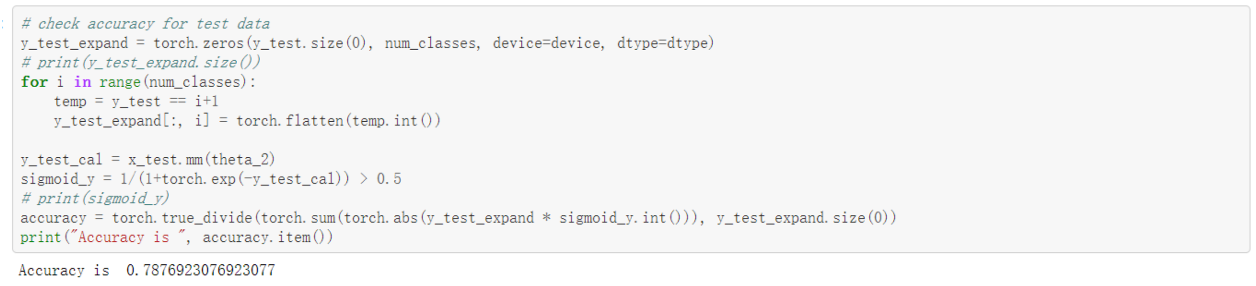
We use the Newton's Method to build the model, and the classification error, model and the accuracy of the model can be seen below.



```
Loss is  [[0.693 0.693 0.693 0.693]]
Loss is  [[0.286 0.201 0.216 0.179]]
Loss is  [[0.175 0.09  0.108 0.08 ]]
Loss is  [[0.106 0.043 0.057 0.039]]
Loss is  [[0.066 0.02  0.029 0.02 ]]
Loss is  [[0.034 0.01  0.015 0.01 ]]
Loss is  [[0.015 0.006 0.009 0.007]]
Loss is  [[0.007 0.005 0.009   nan]]

Parameters theta_2 is:
 [[ 1.147 -0.278 -0.757 -0.69 ]
 [-1.96  -2.543 -1.328 -0.947]
 [-1.082  1.433  0.903  1.148]
 [-0.116 -0.197 -0.243 -0.778]
 [-0.708  6.223  0.516 -7.925]
 [ 0.803 -2.69   3.032  1.651]
 [-0.672 -0.155  0.836  1.806]
 [-0.064 -1.635  7.541  1.308]
 [ 1.176  2.103 -7.393 -0.224]
 [ 0.639 -0.745 -0.016 -0.417]
 [-1.648 -1.006 -1.494 -0.561]
 [ 0.364  0.268  0.598  0.818]
 [ 0.601 -0.462 -0.251 -0.971]
 [-1.464  5.625  0.135 -8.616]
 [ 0.789 -2.332  3.184  1.99 ]
 [-0.42  -0.389  0.519  1.818]
 [ 0.213 -0.98   6.359  1.529]
 [ 1.388  1.982 -4.876 -1.632]
 [ 0.31  -0.263 -0.517  0.505]
 [-1.147 -0.117 -0.397  2.351]
 [-0.081 -0.032  2.351 -2.2  ]
 [ 0.213 -0.219 -0.217 -1.291]
 [-1.895  3.09  -1.066 -4.717]
 [ 2.522 -1.214  1.29   3.31 ]
 [-0.485 -0.239  0.83   1.353]
 [-0.958 -0.364 -2.136  2.833]
 [ 2.222  0.419  0.747 -2.961]]
```

Figure 1: Newton's Method model



```
# check accuracy for test data
y_test_expand = torch.zeros(y_test.size(0), num_classes, device=device, dtype=dtype)
# print(y_test_expand.size())
for i in range(num_classes):
    temp = y_test == i+1
    y_test_expand[:, i] = torch.flatten(temp.int())

y_test_cal = x_test.mm(theta_2)
sigmoid_y = 1/(1+torch.exp(-y_test_cal)) > 0.5
# print(sigmoid_y)
accuracy = torch.true_divide(torch.sum(torch.abs(y_test_expand * sigmoid_y.int())), y_test_expand.size(0))
print("Accuracy is ", accuracy.item())
```

```
Accuracy is  0.7876923076923077
```

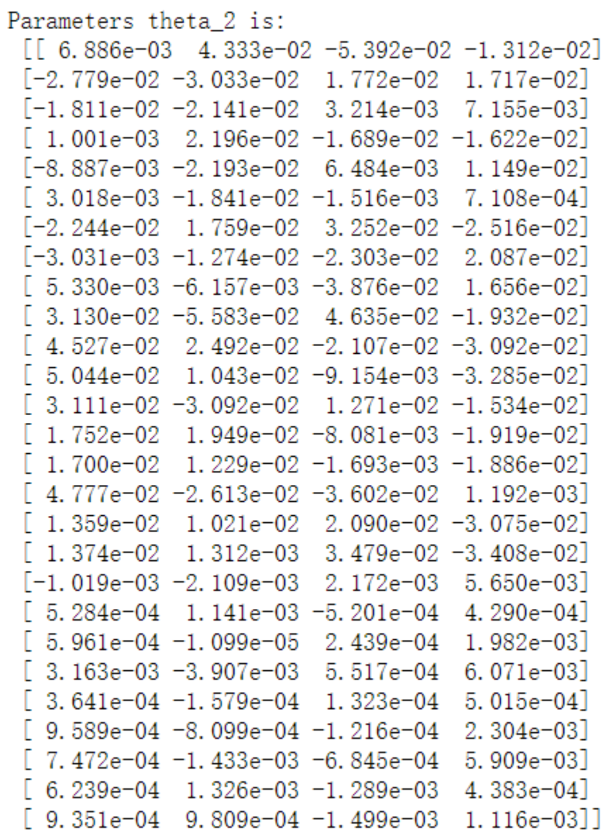
Figure 2: Accuracy of Newton's Method

1.5

According to the 1.3, the gradient of log-likelihood function is:

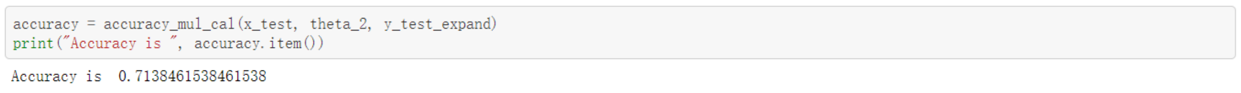
$$\nabla_{\theta_j} l(\theta) = \sum_{i=1}^{m} [x^{(i)}(1\{y^{(i)} = j\} - p(y^{(i)} = j \mid x^{(i)}; \theta))]$$

The model and the accuracy of the model can be seen below.



```
Parameters theta_2 is:
 [[ 6.886e-03  4.333e-02 -5.392e-02 -1.312e-02]
 [-2.779e-02 -3.033e-02  1.772e-02  1.717e-02]
 [-1.811e-02 -2.141e-02  3.214e-03  7.155e-03]
 [ 1.001e-03  2.196e-02 -1.689e-02 -1.622e-02]
 [-8.887e-03 -2.193e-02  6.484e-03  1.149e-02]
 [ 3.018e-03 -1.841e-02 -1.516e-03  7.108e-04]
 [-2.244e-02  1.759e-02  3.252e-02 -2.516e-02]
 [-3.031e-03 -1.274e-02 -2.303e-02  2.087e-02]
 [ 5.330e-03 -6.157e-03 -3.876e-02  1.656e-02]
 [ 3.130e-02 -5.583e-02  4.635e-02 -1.932e-02]
 [ 4.527e-02  2.492e-02 -2.107e-02 -3.092e-02]
 [ 5.044e-02  1.043e-02 -9.154e-03 -3.285e-02]
 [ 3.111e-02 -3.092e-02  1.271e-02 -1.534e-02]
 [ 1.752e-02  1.949e-02 -8.081e-03 -1.919e-02]
 [ 1.700e-02  1.229e-02 -1.693e-03 -1.886e-02]
 [ 4.777e-02 -2.613e-02 -3.602e-02  1.192e-03]
 [ 1.359e-02  1.021e-02  2.090e-02 -3.075e-02]
 [ 1.374e-02  1.312e-03  3.479e-02 -3.408e-02]
 [-1.019e-03 -2.109e-03  2.172e-03  5.650e-03]
 [ 5.284e-04  1.141e-03 -5.201e-04  4.290e-04]
 [ 5.961e-04 -1.099e-05  2.439e-04  1.982e-03]
 [ 3.163e-03 -3.907e-03  5.517e-04  6.071e-03]
 [ 3.641e-04 -1.579e-04  1.323e-04  5.015e-04]
 [ 9.589e-04 -8.099e-04 -1.216e-04  2.304e-03]
 [ 7.472e-04 -1.433e-03 -6.845e-04  5.909e-03]
 [ 6.239e-04  1.326e-03 -1.289e-03  4.383e-04]
 [ 9.351e-04  9.809e-04 -1.499e-03  1.116e-03]]
```

Figure 3: Gradient descent model



```
accuracy = accuracy_mul_cal(x_test, theta_2, y_test_expand)
print("Accuracy is ", accuracy.item())
```

```
Accuracy is  0.7138461538461538
```

Figure 4: Accuracy of Gradient descent

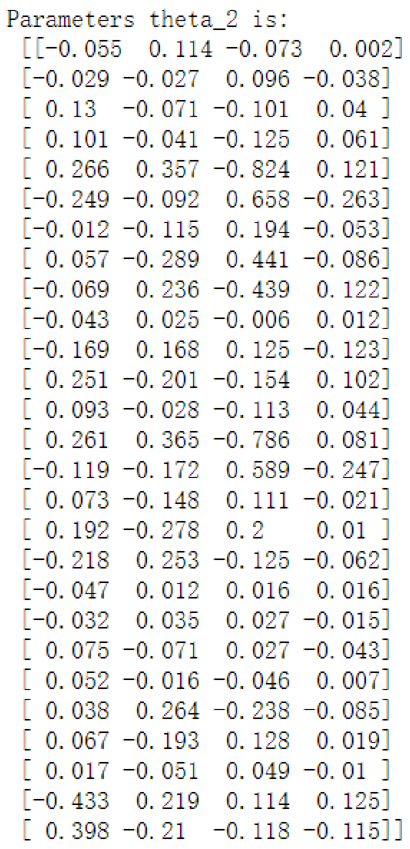
Using Newton's method would cause one problem. If we assume every θ_j is $\theta_j - \psi$, we can get:

$$
\begin{aligned}
p\left(y^{(i)} = j \mid x^{(i)}; \theta\right) &= \frac{e^{(\theta_j - \psi)^T x^{(i)}}}{\sum_{l=1}^{k} e^{(\theta_l - \psi)^T x^{(i)}}} \\
&= \frac{e^{\theta_j^T x^{(i)}} e^{-\psi^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}} e^{-\psi^T x^{(i)}}} \\
&= \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}
\end{aligned}
\tag{17}
$$

In other words, if $(\theta_1, \theta_2, \ldots, \theta_k)$ is the solution of $l(\theta)$, $(\theta_1 - \psi, \theta_2 - \psi, \ldots, \theta_k - \psi)$ also is the solution of $l(\theta)$. Since $l(\theta)$ is a convex function, we won't face the problem of local optimal solutions by using gradient descent. But the Hessian matrix is singular or irreversible, which will directly lead to numerical calculation problems when using Newton's method to optimize.

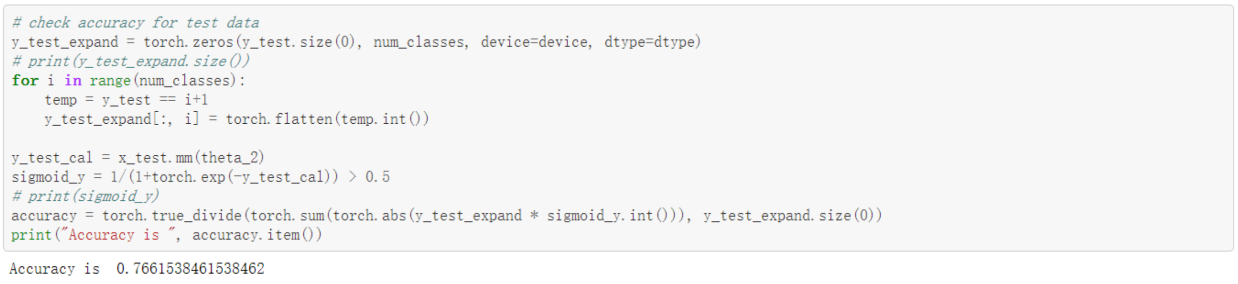
1.6

The model and the accuracy of the model can be seen below.



```
Parameters theta_2 is:
 [[-0.055  0.114 -0.073  0.002]
 [-0.029 -0.027  0.096 -0.038]
 [ 0.13  -0.071 -0.101  0.04 ]
 [ 0.101 -0.041 -0.125  0.061]
 [ 0.266  0.357 -0.824  0.121]
 [-0.249 -0.092  0.658 -0.263]
 [-0.012 -0.115  0.194 -0.053]
 [ 0.057 -0.289  0.441 -0.086]
 [-0.069  0.236 -0.439  0.122]
 [-0.043  0.025 -0.006  0.012]
 [-0.169  0.168  0.125 -0.123]
 [ 0.251 -0.201 -0.154  0.102]
 [ 0.093 -0.028 -0.113  0.044]
 [ 0.261  0.365 -0.786  0.081]
 [-0.119 -0.172  0.589 -0.247]
 [ 0.073 -0.148  0.111 -0.021]
 [ 0.192 -0.278  0.2    0.01 ]
 [-0.218  0.253 -0.125 -0.062]
 [-0.047  0.012  0.016  0.016]
 [-0.032  0.035  0.027 -0.015]
 [ 0.075 -0.071  0.027 -0.043]
 [ 0.052 -0.016 -0.046  0.007]
 [ 0.038  0.264 -0.238 -0.085]
 [ 0.067 -0.193  0.128  0.019]
 [ 0.017 -0.051  0.049 -0.01 ]
 [-0.433  0.219  0.114  0.125]
 [ 0.398 -0.21  -0.118 -0.115]]
```

Figure 5: Backtrack line search for Newton's Method model



```
# check accuracy for test data
y_test_expand = torch.zeros(y_test.size(0), num_classes, device=device, dtype=dtype)
# print(y_test_expand.size())
for i in range(num_classes):
    temp = y_test == i+1
    y_test_expand[:, i] = torch.flatten(temp.int())

y_test_cal = x_test.mm(theta_2)
sigmoid_y = 1/(1+torch.exp(-y_test_cal)) > 0.5
# print(sigmoid_y)
accuracy = torch.true_divide(torch.sum(torch.abs(y_test_expand * sigmoid_y.int())), y_test_expand.size(0))
print("Accuracy is ", accuracy.item())
```

```
Accuracy is  0.7661538461538462
```

Figure 6: Accuracy of backtrack line search for Newton's Method

1.7

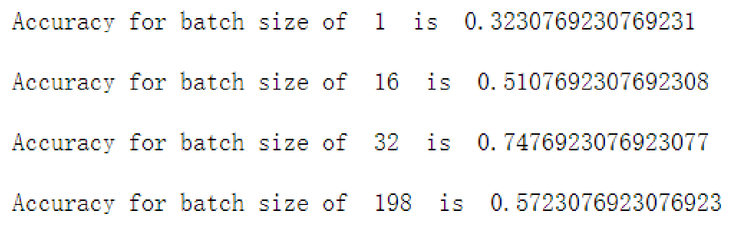
Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). the algorithm of stochastic gradient descent can be presented as follows:

$$\theta_j = \theta_j + (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} \tag{18}$$

In general, the mini-batch gradient descent algorithm can be given as below:

$$\theta_j := \theta_j - \alpha \frac{1}{n} \sum_{k=i}^{n} \left(h_\theta \left(x^{(k)} \right) - y^{(k)} \right) x_j^{(k)} \tag{19}$$

Where $n = 1, \ldots, m$.



```
Accuracy for batch size of  1  is  0.3230769230769231

Accuracy for batch size of  16  is  0.5107692307692308

Accuracy for batch size of  32  is  0.7476923076923077

Accuracy for batch size of  198  is  0.5723076923076923
```

Figure 7: Accuracy of stochastic gradient descent

2. Quadratic Programming

$$f(x) = \frac{1}{2}(x_1^2 + 2(1-\varepsilon)x_1 x_2 + x_2^2)$$

$$H = \nabla^2 f(x) = \begin{bmatrix} 1 & (1-\varepsilon) \\ (1-\varepsilon) & 1 \end{bmatrix}$$

when $H \cdot x = \lambda \cdot x$

$\Rightarrow (H - \lambda I)x = 0$

$\Rightarrow \det(H - \lambda I) = 0 \Rightarrow \left| \begin{bmatrix} (1-\lambda) & (1-\varepsilon) \\ (1-\varepsilon) & (1-\lambda) \end{bmatrix} \right| = 0$

$\Rightarrow \lambda_1 = \varepsilon, \quad \lambda_2 = 2 - \varepsilon$

$$K(H) = \frac{\lambda_{max}(H)}{\lambda_{min}(H)}$$

1° $\varepsilon \leq 1$

$$K(H) = \frac{2-\varepsilon}{\varepsilon} = \frac{2}{\varepsilon} - 1$$

2° $\varepsilon > 1$

$$K(H) = \frac{\varepsilon}{2-\varepsilon} = \frac{1}{\frac{2}{\varepsilon} - 1}$$

3° when $\varepsilon \to 0$

$$K(H) = \frac{2}{\varepsilon} - 1 \to +\infty$$

3. General Linear models for Softmax regression

1. $P(y=i;\phi)=\phi_i$, $\sum_{i=1}^{k}\phi_i=1$, $T(y)_i = 1\{y=i\}$ er

$$\Rightarrow P(y;\phi) = \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\cdots\phi_k^{1\{y=k\}}$$

$$= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\cdots\phi_k^{1-\sum_{i=1}^{k-1}1\{y=i\}}$$

$$= \phi_1^{(T(y))_1}\phi_2^{(T(y))_2}\cdots\phi_k^{1-\sum_{i=1}^{k-1}(T(y))_i}$$

$$= \exp\left((T(y))_1\log(\phi_1)+(T(y))_2\log(\phi_2)+\cdots+ + \left(1-\sum_{i=1}^{k-1}(T(y))_i\right)\log(\phi_k)\right)$$

$$= \exp\left((T(y))_1\log\left(\frac{\phi_1}{\phi_k}\right)+(T(y))_2\log\left(\frac{\phi_2}{\phi_k}\right) + \cdots + (T(y))_{k-1}\log\left(\frac{\phi_{k-1}}{\phi_k}\right)+\log(\phi_k)\right)$$

$$= b(y)\exp(\eta^T T(y) - a(\eta))$$

$$\left(\eta=\begin{bmatrix}\log\left(\frac{\phi_1}{\phi_k}\right)\\ \log\left(\frac{\phi_2}{\phi_k}\right)\\ \vdots \\ \log\left(\frac{\phi_{k-1}}{\phi_k}\right)\end{bmatrix},\quad a(\eta)=-\log(\phi_k),\quad b(y)=1\right)$$

2. $\because \eta=\begin{bmatrix}\log\left(\frac{\phi_1}{\phi_k}\right)\\ \vdots \\ \log\left(\frac{\phi_{k-1}}{\phi_k}\right)\end{bmatrix}$

$\therefore \eta_i = \log\frac{\phi_i}{\phi_k}$

$\therefore e^{\eta_i} = \frac{\phi_i}{\phi_k} \Rightarrow \phi_i = e^{\eta_i}\cdot\phi_k$

$\therefore \phi_k\cdot e^{\eta_i} = \phi_i$

$\therefore \phi_k\cdot\sum_{i=1}^{k} e^{\eta_i} = \sum_{i=1}^{k}\phi_i = 1$

$\therefore \phi_k = \frac{1}{\sum_{i=1}^{k} e^{\eta_i}}$

$\therefore \phi_i = e^{\eta_i} \cdot \phi_k = \dfrac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}$

$\because$ In general linear model, natural parameter (η_i) is $\theta_i^T x$ $(\forall i = 1, \cdots, k-1)$

$\therefore \eta_i = \theta_i^T x \quad (\forall i = 1, \cdots, k-1)$

$\therefore P(y=i \mid x; \theta) = \phi_i = \dfrac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} = \dfrac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$

3. $$L(\theta) = P(y \mid x; \theta) = \prod_{i=1}^{n} P(y_i \mid x^{(i)}; \theta)$$

$$\Rightarrow l(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log P(y_i \mid x^{(i)}; \theta)$$

$$= \sum_{i=1}^{n} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_i^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y_i = l\}}$$

$$= \sum_{i=1}^{n} \sum_{l=1}^{k} \log \left(\frac{e^{\theta_i^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y_i = l\}}$$

$$= \sum_{i=1}^{n} \sum_{l=1}^{k} 1\{y_i = l\} \log \left(\frac{e^{\theta_i^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)$$

4. $$l(\theta) = \sum_{i=1}^{n} \sum_{l=1}^{k} 1\{y_i = l\} \log \left(\frac{e^{\theta_i^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)$$

$$\frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} 1\{y_i = l\} \cdot \frac{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}{e^{\theta_i^T x^{(i)}}} \cdot \left(\frac{x^{(i)} \cdot \left(e^{\theta_i^T x^{(i)}} \cdot \sum_{j=1}^{k} e^{\theta_j^T \cdot x^{(i)}} - (e^{\theta_i^T \cdot x^{(i)}})^2\right)}{\left(\sum_{j=1}^{k} e^{\theta_j^T \cdot x^{(i)}}\right)^2} \right)$$

$$= \sum_{i=1}^{n} 1\{y_i = l\} \cdot \frac{\sum_{j=1}^{k} e^{\theta_j^T \cdot x^{(i)}}}{e^{\theta_i^T \cdot x^{(i)}}} \cdot x^{(i)} \cdot \frac{e^{\theta_i^T x^{(i)}} \cdot \sum_{j=1}^{k} e^{\theta_j^T \cdot x^{(i)}} - e^{\theta_i^T x^{(i)}} \cdot e^{\theta_i^T x^{(i)}}}{\left(\sum_{j=1}^{k} e^{\theta_j^T \cdot x^{(i)}}\right)^2}$$

$$= \sum_{i=1}^{n} \left[1\{y_i = l\} \cdot \frac{\sum_{j=1}^{k} e^{\theta_j^T \cdot x^{(i)}} - e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \cdot x^{(i)} \right]$$

$$= \sum_{i=1}^{n} \left[1\{y_i = l\} \cdot x^{(i)} \cdot \left(1 - \frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right) \right]$$