INT 305 Assignment 1

(The deadline is 30st of Oct.)

1. Please write down the whole derivation process to obtain the gradient for logistic regression. (30%)

The logistic model is: $z = w^{\mathsf{T}}x$.

And the activation function is: $y = \frac{1}{(1+e^{-z})}$

The loss function is: $\mathcal{L}_{CE}(y,t) = -t\log(y) - (1-t)\log(1-y)$

To optimize the model, we should update the weight w by using gradient descent.

Therefore, by using the chain rule:

$$\frac{\partial \mathcal{L}_{CE}}{\partial w_j} = \frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_j}$$

[Step1] Because $\mathcal{L}_{CE}(y,t) = -t\log(y) - (1-t)\log(1-y)$:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{CE}}{\partial y} &= \frac{\partial[-t\log(y)]}{\partial y} - \frac{\partial[(1-t)\log(1-y)]}{\partial y} \\
&= \left(-t \cdot \frac{1}{y}\right) - \frac{\partial[(1-t)\log(1-y)]}{\partial(1-y)} \cdot \frac{\partial(1-y)}{y} \\
&= \left(-t \cdot \frac{1}{y}\right) - \left[(1-t) \cdot \frac{1}{(1-y)}\right] \cdot (-1) \\
&= \left(-t \cdot \frac{1}{y}\right) - \left[-\frac{(1-t)}{(1-y)}\right] \\
&= \left(-\frac{t}{y} + \frac{1-t}{1-y}\right)
\end{aligned}$$

[Step2] Because $y = \frac{1}{(1+e^{-z})}$:

$$\begin{aligned}
\frac{\partial y}{\partial z} &= \frac{\partial\left[\frac{1}{(1+e^{-z})}\right]}{\partial(1+e^{-z})} \cdot \frac{\partial(1+e^{-z})}{\partial z} \\
&= -(1+e^{-z})^{-2} \cdot (-e^{-z}) \\
&= \frac{e^{-z}}{(1+e^{-z})(1+e^{-z})} \\
&= \frac{1}{(1+e^{-z})} \cdot \frac{(1+e^{-z})-1}{(1+e^{-z})} \Rightarrow y(1-y)
\end{aligned}$$

And $z = w^{\mathsf{T}}x$:

$$\frac{\partial z}{\partial w_j} = \frac{\partial(w_j x_j)}{\partial w_j} = x_j$$

[Step3] Therefore:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{CE}}{\partial w_j} &= \frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_j} \\
&= \left(-\frac{t}{y} + \frac{1-t}{1-y}\right) \cdot y(1-y) \cdot x_j \\
&= [-t(1-y) + (1-t) \cdot y] \cdot x_j \\
&= (-t + ty + y - ty) \cdot x_j \\
&= (y - t) \cdot x_j
\end{aligned}
$$

[Step4] Finally, we can get the gradient of w, then we need to update weight w:

$$
\begin{aligned}
\text{Because: } \mathcal{J} &= \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{CE} \\
\text{thus: } w_j &\leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j} \\
&= w_j - \frac{\alpha}{N} \sum_{i=1}^{N} \left(y^{(i)} - t^{(i)}\right) x_j^{(i)}
\end{aligned}
$$

2. Please write down the whole derivation process to obtain the gradient for multiclass classification with softmax. (40%)

The model function is: $z = Wx$,

$$\Rightarrow z_k = w_k \cdot x$$

The Softmax function is: $y = \mathrm{softmax}(z)$

$$\Rightarrow y_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

The cross-entropy loss function is: $\mathcal{L}_{CE}(y,t) = -\sum_{k=1}^{K} t_k \log y_k$

$$= -t^{\top}(\log y)$$

To optimize the model, we should update the weight w by using gradient descent.

Therefore, by using the chain rule:

$$\frac{\partial \mathcal{L}_{CE}}{\partial w_k} = \frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_k}$$

[Step1] Because $\mathcal{L}_{CE} = -t^{\top}(\log y)$:

$$\frac{\partial \mathcal{L}_{CE}}{\partial y} = \frac{\partial[-t^{\top}(\log y)]}{\partial y}$$

- Since after one-hot encoding, for classification task with K classes, t_k is a K-dimensional vector.
- Only the position of the correct class is 1, and the rest is 0. So, here are two cases: for correct class T, $t_T = 1$, $t_{m \neq T} = 0$.
- According to the function $\mathcal{L}_{CE}(y,t) = -\sum_{k=1}^{K} t_k \log y_k$, we only need to consider the condition of t_T, because the rest condition will get 0. For the same reason, we only need to consider y_T as well.

Therefore, $\mathcal{L}_{CE} = -t_T(\log y_T) = -\log y_T$:

$$\frac{\partial \mathcal{L}_{CE}}{\partial y_T} = \frac{\partial(-\log y_T)}{\partial y_T} = -\frac{1}{y_T}$$

[Step2] Because $y_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$:

$$\frac{\partial y}{\partial z_k} = \frac{\partial y_T}{\partial z_k} = \frac{\partial y_k}{\partial z_k} = \frac{\partial \left(\frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}\right)}{\partial z_k}$$

There are also two cases:

(1) While $T = k$:

$$\frac{\partial y_T}{\partial z_k} = \frac{\partial y_k}{\partial z_k} = \frac{\partial \left(\frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}\right)}{\partial z_k} = \frac{e^{z_k} \cdot \sum_{k'} e^{z_{k'}} - e^{z_k} \cdot \frac{\partial(\sum_{k'} e^{z_{k'}})}{\partial z_k}}{(\Sigma_{k'} e^{z_{k'}})^2}$$

Because z_k is one of $z_{k'}$, so:

$$\frac{\partial(\sum_{k'} e^{z_{k'}})}{\partial z_k} = \frac{\partial(e^{z_1+\cdots+e^{z_k}+\cdots+e^{z_n}})}{\partial z_k} = e^{z_k}$$

Thus:

$$\frac{\partial y_T}{\partial z_k} = \frac{\partial y_k}{\partial z_k} = \frac{e^{z_k} \cdot \sum_{k'} e^{z_{k'}} - e^{z_k} \cdot e^{z_k}}{(\sum_{k'} e^{z_{k'}})^2} = \frac{e^{z_k}(\sum_{k'} e^{z_{k'}} - e^{z_k})}{\sum_{k'} e^{z_{k'}} \cdot \sum_{k'} e^{z_{k'}}} = y_k(1 - y_k)$$

(2) While $T \neq k$:

$$\frac{\partial y_T}{\partial z_k} = \frac{\partial \left(\frac{e^{z_T}}{\sum_{k'} e^{z_{k'}}}\right)}{\partial z_k} = \frac{0 - e^{z_T} \cdot e^{z_k}}{(\sum_{k'} e^{z_{k'}})^2} = \frac{-e^{z_T} \cdot e^{z_k}}{\sum_{k'} e^{z_{k'}} \cdot \sum_{k'} e^{z_{k'}}} = -y_T \cdot y_k$$

[Step3] Because $z_k = w_k \cdot x$:

$$\frac{\partial z_k}{\partial w_k} = \frac{\partial(w_k \cdot x)}{\partial w_k} = x$$

[Step4] Because we have two cases of $\frac{\partial y}{\partial z_k}$, so we should consider them both when calculate $\frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z_k}$.

(1) While $T = k \; or \; t_k = 1$:

$$\frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z_k} = \frac{\partial \mathcal{L}_{CE}}{\partial z_k} \quad = \frac{\partial \mathcal{L}_{CE}}{\partial y_T} \cdot \frac{\partial y_T}{\partial z_k}$$
$$= -\frac{1}{y_k} \cdot y_k(1-y_k)$$
$$= y_k - 1$$
$$= y_k - t_k$$

(2) While $T \neq k \; or \; t_k = 0$:

$$\frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z_k} = \frac{\partial \mathcal{L}_{CE}}{\partial z_k} \quad = \frac{\partial \mathcal{L}_{CE}}{\partial y_T} \cdot \frac{\partial y_T}{\partial z_k}$$
$$= -\frac{1}{y_T} \cdot (-y_T \cdot y_k)$$
$$= y_k$$
$$= y_k - 0$$
$$= y_k - t_k$$

Therefore, the result of $\frac{\partial \mathcal{L}_{CE}}{\partial z_k}$ are both $y_k - t_k$.

[Step5] Gradient descent updates can be derived for each row of w:

$$\begin{aligned}\frac{\partial \mathcal{L}_{CE}}{\partial w_k} &= \frac{\partial \mathcal{L}_{CE}}{\partial y} \cdot \frac{\partial y}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_k} \\ &= \frac{\partial \mathcal{L}_{CE}}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_k} \\ &= (y_k - t_k) \cdot x\end{aligned}$$

Now, we can get the gradient of w, then we need to update weight w:

$$\begin{aligned}\text{Because: } \mathcal{J} &= \frac{1}{N}\sum_{k=1}^{N} \mathcal{L}_{CE} \\ \text{thus: } w_k &\leftarrow w_k - \alpha \frac{\partial \mathcal{J}}{\partial w_k} \\ &= w_k - \frac{\alpha}{N}\sum_{i=1}^{N}\left(y_k^{(i)} - t_k^{(i)}\right) x^{(i)}\end{aligned}$$

3. **Please compare the SVM loss and Softmax loss for multiclass classification, please explain which one is better? (30%)**

Result: The Softmax loss is better for multiclass classification.

	Cell 1	Cell 2	Cell 3
Example 1	10	-2	3
Example 2	10	9	9
Example 3	10	-100	-100

For example, there is a multiclass classification task with 3 cells. And we sample 3 training examples, their scores are shown above (cell 1 is label).

Then we calculate SVM loss and Softmax loss for these three examples respectively.

The formula of Softmax loss is:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

The formula of SVM loss is:

$$L_i = \sum_{j \neq y_i} \max\left(0, s_j - s_{y_i} + 1\right)$$

The results are shown below:

	Cell 1	Cell 2	Cell 3	**SVM**	**Softmax**
Example 1	10	-2	3	0	0.4E-3
Example 2	10	9	9	0	0.24
Example 3	10	-100	-100	0	0

It can be seen from the results that SVM loss cannot reflect the degree of model optimization precisely. When using SVM Loss to optimize the model, we may only find the local optimal solution. Because after obtaining a solution, SVM Loss will become 0.

However, Softmax loss doesn't have that problem, it is a good reflection of the current model. Moreover, even if we find a solution, we can continue to optimize until we find the optimal solution.

Therefore, the Softmax loss is better than SVM loss for multiclass classification.

<Python process of Q3>

```python
class SoftmaxLoss(object):
    def expect(n):
        L0 = []
        for i in n:
            L0.append(np.exp(i))
        return L0

    def correct(n):
        L0 = []
        for i in n:
            x = np.log10(i)
            if x == 0:
                L0.append(np.log10(i))
            else:
                L0.append(-np.log10(i))
        return L0

    def normalize(n):
        L0 = []
        for i in n:
            L0.append(i / np.sum(n))
        return L0


class SVMLoss(object):
    def formula(n):
        return max(0, n[1] - n[0] + 1) + max(0, n[2] - n[0] + 1)


if __name__ == "__main__":
    L = [[10, -2, 3],
         [10, 9, 9],
         [10, -100, -100]]
    L1 = []
    L2 = []
    for i in L:
        ex = SoftmaxLoss.expect(i)
        no = SoftmaxLoss.normalize(ex)
        co = SoftmaxLoss.correct(no)
        L1.append(co[0])
        fo = SVMLoss.formula(i)
        L2.append(fo)
    print(f"The Softmax loss is {L1}")
    print(f"The SVM loss is {L2}")
```

Output:

The Softmax loss is [0.0003985108096053745, 0.23948939633540703, 0.0]

The SVM loss is [0, 0, 0]