Ex 8 Timo Marks 14.06.22

1) With the gradient descent algorithm we want to optimize some parameters e.g. the weights in our learning algorithm. We start with the initialization of the weights (could be random, could be gaussian, could be all 1) $\Rightarrow \vec{w}^{(0)}$

Then we define any loss function $L(\vec{w})$ which gives an estimate about how good our weights lead to the desired output $\vec{y}$. We want to minimize our loss, so minimize $L(\vec{w})$, we look for the steepest descent $\vec{\nabla}_{\vec{w}} L(\vec{w})\big|_{\vec{w}^{(k)}}$ to hopefully get to the global minimum. $\rightarrow$ iteration k

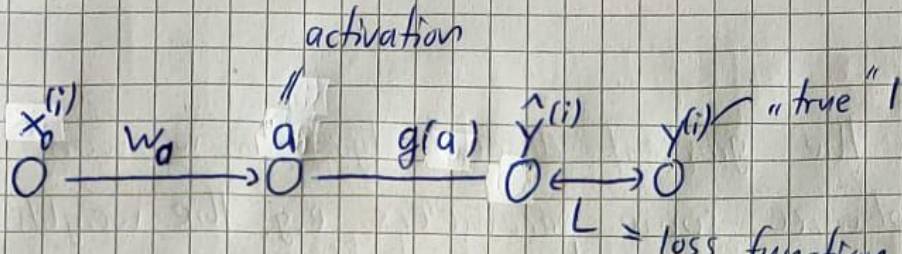
We update our weights proportional to the gradient by going a step with the step size $\eta$ in this direction. $\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \cdot \vec{\nabla}_{\vec{w}} L(\vec{w})\big|_{\vec{w}^{(k)}}$

We do this until convergence, which could mean $\vec{\nabla}_{\vec{w}} L(\vec{w})\big|_{\vec{w}^{(k)}} < C$, with C as any threshold for the change in the loss function. Pseudocode:

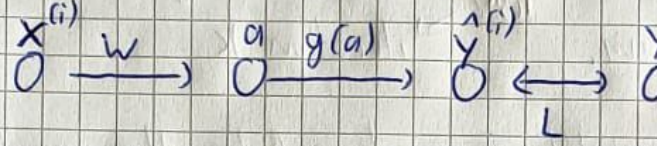
```
weights = function_initial_weights ()

  while iteration < max_iteration or delta_L > threshold
  do {
     delta_L = compute_gradient (weights, loss_function)
     weights = weights - step_size · delta_L
     iteration = iteration + 1
  }
```

2)




In general our logistic regression model uses the function $g(a) = \frac{1}{1+e^{-a}}$ which is in this case $\hat{y}^{(i)} = g(w_0 \cdot x_0^{(i)}) = \frac{1}{1+e^{-w_0 x_0^{(i)}}}$. We can omit the index 0 because we only have one input and one weight, but this indicates that in general we have D-dimensional vectors.




Cross-entropy loss function:

$$L = -\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)} \cdot \log\left(\hat{y}^{(n)}\right) + \left(1-y^{(n)}\right) \cdot \log\left(1-\hat{y}^{(n)}\right)\right]$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)} \cdot \log\left(\frac{1}{1+e^{-wx^{(n)}}}\right) + \left(1-y^{(n)}\right) \cdot \log\left(1-\frac{1}{1+e^{-wx^{(n)}}}\right)\right]$$

Gradient loss function

$$\frac{\partial}{\partial w}\log\left(\frac{1}{1+e^{-wx^{(n)}}}\right) = \left(1+e^{-wx^{(n)}}\right) \cdot \frac{\partial}{\partial w}\left(\frac{1}{1+e^{-wx^{(n)}}}\right)$$

$$= \left(1+e^{-wx^{(n)}}\right) \cdot \frac{-1}{\left(1+e^{-wx^{(n)}}\right)^2} \cdot \left(-x^{(n)}\right) \cdot e^{-wx^{(n)}}$$

$$= \frac{x^{(n)} e^{-wx^{(n)}}}{1+e^{-wx^{(n)}}} = \hat{y}^{(n)} e^{-wx^{(n)}} \cdot x^{(n)}$$

Second term:

$$\frac{\partial}{\partial w}\log\left(1-\frac{1}{1+e^{-wx^{(n)}}}\right)=\frac{\partial}{\partial w}\log\left(\frac{e^{-wx^{(n)}}}{1+e^{-wx^{(n)}}}\right)$$

↓ chain rule + quotient rule

$$=\frac{1+e^{-wx^{(n)}}}{e^{-wx^{(n)}}}\cdot\frac{-x^{(n)}e^{-wx^{(n)}}\cdot\left(1+e^{-wx^{(n)}}\right)-e^{-wx^{(n)}}\left(-x^{(n)}\right)e^{-wx^{(n)}}}{\left(1+e^{-wx^{(n)}}\right)^2}$$

$$=\frac{-x^{(n)}\left(1+e^{-wx^{(n)}}\right)+x^{(n)}e^{-wx^{(n)}}}{1+e^{-wx^{(n)}}}$$

$$=\frac{-x^{(n)}}{1+e^{-wx^{(n)}}}=-\hat{y}^{(n)}x^{(n)}$$

Also notice: $\hat{y}^{(n)}=\frac{1}{1+e^{-wx^{(n)}}} \Leftrightarrow 1=\hat{y}^{(n)}+\hat{y}^{(n)}e^{-wx^{(n)}}$

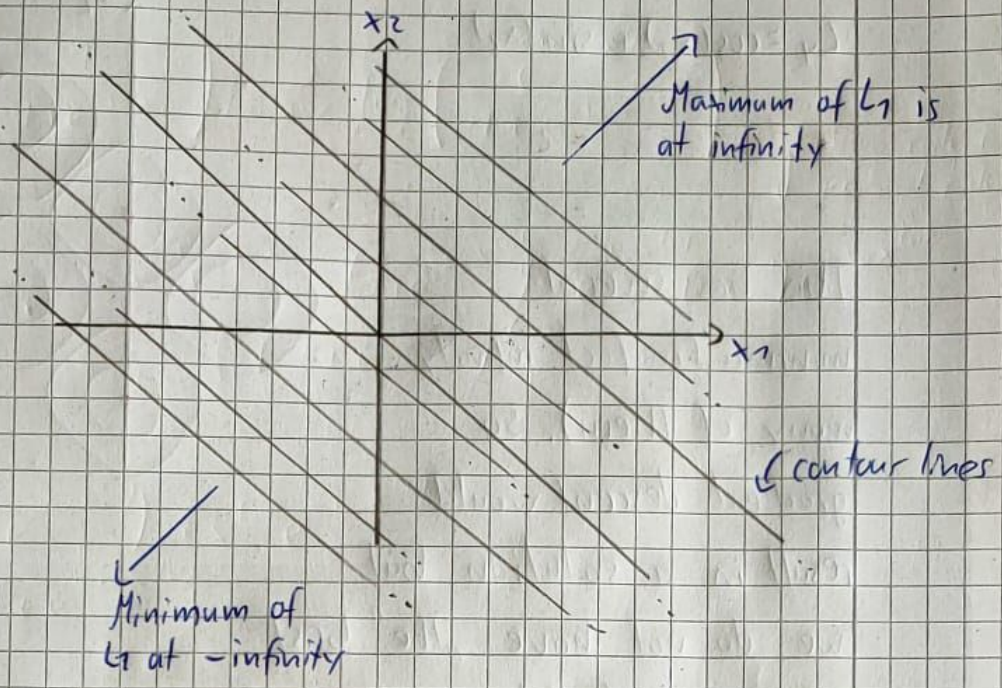
Together:

$$\frac{\partial}{\partial w}L=-\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}\left(\hat{y}^{(n)}e^{-wx^{(n)}}x^{(n)}\right)+\left(1-y^{(n)}\right)\cdot\hat{y}^{(n)}\cdot\left(-x^{(n)}\right)\right]$$

$$=-\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}\left\{\underbrace{\hat{y}^{(n)}e^{-wx^{(n)}}+\hat{y}^{(n)}}_{=1}\right\}-\hat{y}^{(n)}\right]x^{(n)}$$

$$=\frac{1}{N}\sum_{n=1}^{N}\left(\hat{y}^{(n)}-y^{(n)}\right)x^{(n)}$$

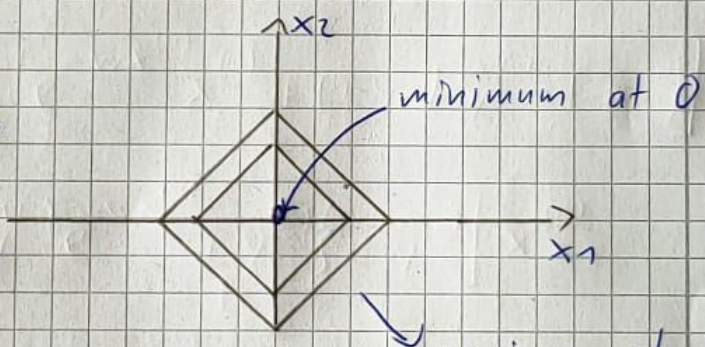
3) $L_1 = x_1 + x_2$




This is not suitable as a loss function because when we would use gradient descent on it, we wouldn't get to a distinct minimum. We do not actually get to the global minimum, just like in $L = x^2$ because the minimum sits at infinity

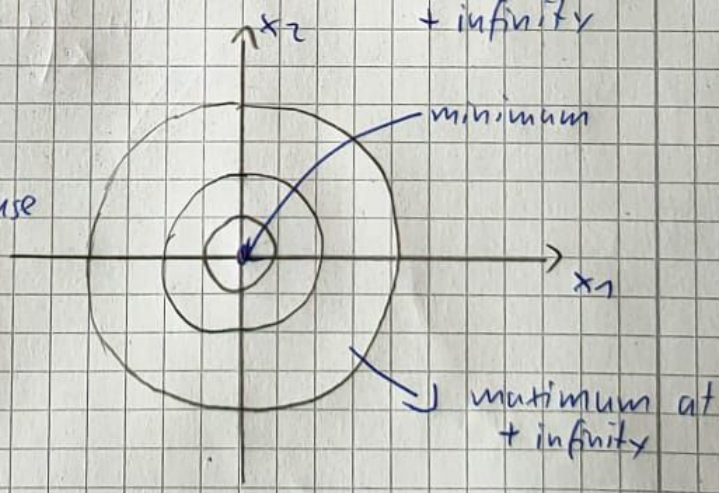
$L_2 = |x_1| + |x_2|$

This is suitable because the gradient descent would lead to the global minimum at $x_1 = x_2 = 0$



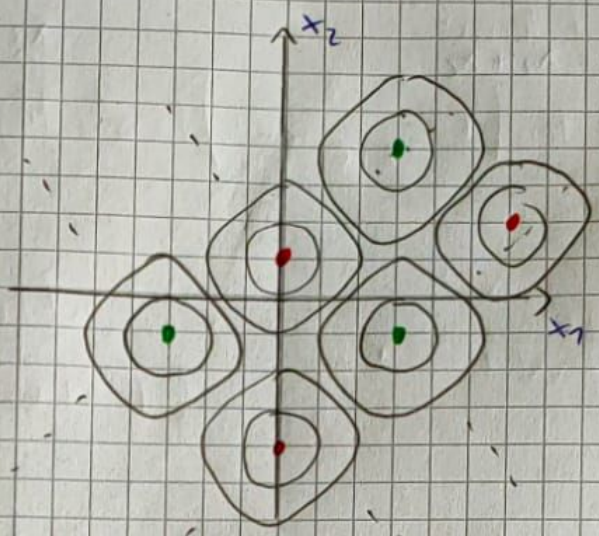

$L_3 = x_1^2 + x_2^2$

This is also suitable because just like $L_2$ we have a distinct global minimum at $x_1 = x_2 = 0$.



$L_4 = \cos(x_1) + \sin(x_2)$

- Maximum = 2
- Minimum = -2




This has several minima (in fact infinity many of them). So gradient descent would result in a solution but we do not have „the" best solution. Preferably use $L_2$ or $L_3$.