1. Polynomial Regression

$x_1 \to y$
$x_2 \to y$
$x_1 \cdot x_2 \to y$
$x_1^2 \to y$
$x_2^2 \to y$

$\theta_0 \cdot x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot x_1 \cdot x_2 + \theta_4 \cdot x_1^2 + \theta_5 \cdot x_2^2$

$y_{pred} = \sum_{i=0}^{5} \theta_i \cdot \text{feat}_i(x_1, x_2)$

2) Design matrix

$$\phi = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

3) $w = (\phi^T \phi)^{-1} \phi^T t$

$$= \begin{bmatrix} 3.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 0.0000 \\ 1.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

4) Prove for the LMSE regression that the penalty term quotient when λ is set: $\frac{\text{Effectivelambda}}{\text{Lambda}}$ (Lambda to zero)

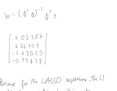
---

2) Neural Networks

$x = [1 \ 1 \ 1 \ 1 \ 1]^T \quad t = (1, 0)^T$

$$W^{(1)} = \begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.1 & 0.2 & 0.1 \end{bmatrix} \qquad b^{(1)} = (1)$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix} \qquad b^{(2)} = (1)$$



cross entropy $= -\sum_i t_i \log(y_i)$

$= -\sum_i t_i \log(y_i) \cdot x_i$

cross entropy: 1.313

$1^{st} \to$ Forwardpy of $x^{(1)}$

$z^{(1)} = W^{(1)} x + b^{(1)} = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}$

$h^{(1)} = \text{ReLU}(z^{(1)}) = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}$

$z^{(2)} = W^{(2)} h^{(1)} + b^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y^{(1)} = \text{Softmax}(z^{(2)}) = \frac{e^{z_i^{(2)}}}{\sum_j e^{z_j^{(2)}}} = \begin{bmatrix} \frac{e^1}{e^1 + e^0} \\ \frac{e^0}{e^1 + e^0} \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix}$

$y^{(1)} = \begin{bmatrix} 0.269 \\ 0.731 \end{bmatrix}$

$3^{rd} \to$ Calculate δ_i's

$\delta^{(2)} = \frac{\partial L^{(1)}}{\partial z^{(2)}} = \frac{\partial L^{(1)}}{\partial y^{(1)}} \cdot \begin{bmatrix} \frac{\partial y}{\partial z_1} \\ \frac{\partial y}{\partial z_2} \end{bmatrix}$

$= \begin{bmatrix} 0.269 - t_1 \\ 0.731 - t_2 \end{bmatrix} = \begin{bmatrix} 0.269 \\ -0.269 \end{bmatrix} = \begin{bmatrix} 0.731 \\ 0.731 \end{bmatrix}$

$\delta^{(1)} = \underbrace{\frac{\partial z^{(2)}}{\partial h^{(1)}}}_{W_2} \cdot \delta^{(2)} \cdot \underbrace{\frac{\partial h^{(1)}}{\partial z^{(1)}}}_{\text{derivative activation function}} = \begin{bmatrix} 0.00 & 0 \\ 2.00 & 0 \end{bmatrix}^T \begin{bmatrix} 0.269 \\ -0.269 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$= \begin{bmatrix} 0.538 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.538 \\ 0 \end{bmatrix}$

$3^{rd} \to$ Update W_2 and b_2

$W^{(2)} = W^{(2)} - \eta \frac{\partial L}{\partial W^{(2)}} =$

$\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)} \left(\frac{\partial z^{(2)}}{\partial W^{(2)}} \right)^T = \delta^{(2)} \cdot (x^{(1)})^T =$

$= \begin{bmatrix} 0.538 & 0.538 & 0.538 & 0.538 & 0.538 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$W^{(2)} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} - 0.1 \begin{bmatrix} 0.538 & 0.538 & 0.538 & 0.538 & 0.538 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$= \begin{bmatrix} -0.0538 & -0.0538 & -0.0538 & -0.0538 & -0.0538 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$b^{(2)} = b^{(2)} - \eta \frac{\partial L}{\partial b^{(2)}}$

$\frac{\partial L}{\partial b^{(2)}} = \delta^{(2)} \cdot \frac{\partial z^{(2)}}{\partial b^{(2)}} = \delta^{(2)} = \begin{bmatrix} 0.538 \\ 0 \end{bmatrix}$

$b^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} 0.538 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.0538 \\ 0 \end{bmatrix}$

$W^{(1)} = W^{(1)} - \eta \frac{\partial L}{\partial W^{(1)}} =$

$\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)} \left(\frac{\partial z^{(1)}}{\partial W^{(1)}} \right)^T = \delta^{(1)} \cdot (x^{(1)})^T =$

$= \begin{bmatrix} 0.538 \\ 0.538 \end{bmatrix} \begin{bmatrix} 0.1 & 0.5 \end{bmatrix}$

$= \begin{bmatrix} 0.0538 & 0.2688 \\ 0.0538 & 0.2688 \end{bmatrix}$

$W^{(1)} = W^{(1)} - 0.1 \boxed{} = \begin{bmatrix} 0.0946 & 0.1462 \\ 0.1946 & 0.0462 \end{bmatrix}$

$b^{(1)} = b^{(1)} - \eta \frac{\partial L}{\partial b^{(1)}}$

$\frac{\partial L}{\partial b^{(1)}} = \delta^{(1)} \cdot \frac{\partial z^{(1)}}{\partial b^{(1)}} = \delta^{(1)} = \begin{bmatrix} 0.538 \\ 0.538 \end{bmatrix}$

$b^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} 0.538 \\ 0.538 \end{bmatrix} = \begin{bmatrix} -0.0538 \\ -0.0538 \end{bmatrix}$

cross entropy after update: 0.037
(before it was 1.313)

Sophisticated experiments

1. split the data
2. compare the accuracy on the test set of logistic regression

a) Change the hidden_layer_sizes

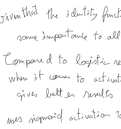
$\underbrace{\text{hidden_layer_sizes}}_{\text{hidden layer}} = (\underbrace{10}_{\text{neurons}}, \underbrace{4}_{\text{neurons}})$

output layer

run the test accuracy and check the parameters

initial accuracy
↳ with logistic regression: 0.95
↳ with hidden_layer_sizes(4): 0.7008

Best parameters (that we tried)
hidden_layer_sizes = (100) ⇒ accuracy: 0.9655
output_size



This is the loss curve, that shows the difference between the output and the target. Because it goes down to zero, the accuracy gets better with more iterations.

Also, the best parameter for the hidden layer size was (100,) because it had our larger layer that prevents underfitting, by being able to learn complex patterns

b)

use if the accuracy is still good with the same parameters (a)
with a different activation function
⇓
IDENTITY FUNCTION

The accuracy lowered from 0.9655 to 0.9588.
Given that the identity function gives the same importance to all parameters.
Compared to logistic regression function when it comes to activating, the latter gives better results because it uses sigmoid activation to output probabilities directly.

In conclusion, logistic regression is better for classification because it calculates the probabilities, it doesn't just automatically give the same importance to all parameters.