Liheng Cao

April 17, 2021

1

(a)

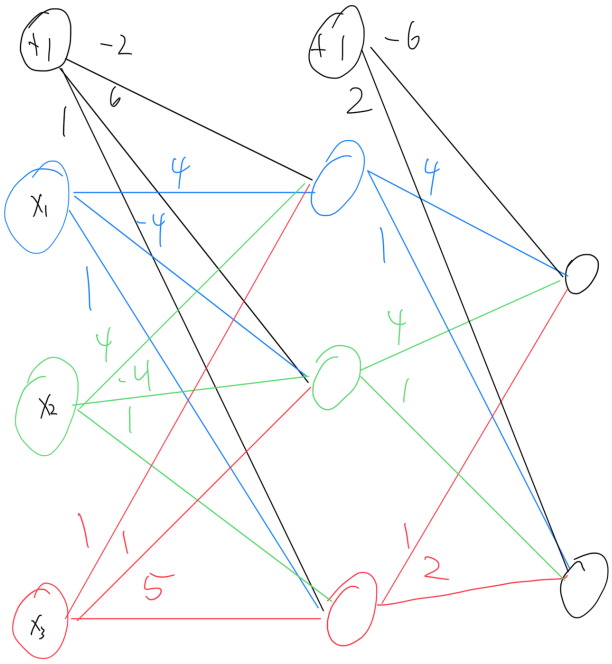

Figure 1: Drawing of Neural Network

(b)

$$z^{(i+1)} = W^{(i)}a^{(i)} + b^{(i)}, a^{(i)} = \sigma\left(z^{(i)}\right)$$

The first a is x.

$$h_{W,b}(\mathbf{x}) = \sigma\left(W^{(2)}\sigma\left(W^{(1)}x + b^{(1)}\right) + b^{(2)}\right)$$

$$= \sigma\left(\begin{bmatrix}4 & 4 & 1\\ 1 & 1 & 2\end{bmatrix}\sigma\left(\begin{bmatrix}4 & 4 & 1\\ -4 & -4 & 1\\ 1 & 1 & -5\end{bmatrix}\begin{bmatrix}1\\2\\3\end{bmatrix} + \begin{bmatrix}-2\\6\\1\end{bmatrix}\right) + \begin{bmatrix}-6\\2\end{bmatrix}\right)$$

$$= \begin{bmatrix}0.1406\ldots\\0.9547\ldots\end{bmatrix}$$

(c)

- $\delta_1^{(3)} = \dfrac{\partial J}{\partial z_1^{(3)}} = -\left(y_1 - \sigma\left(z_1^3\right)\right)\sigma'\left(z_1^3\right) \approx -\left(1 - 0.1406\right)\left(0.1406 \cdot \left(1 - 0.1406\right)\right) \approx -0.1038$
- $a_1^{(2)} = \sigma\left(\begin{bmatrix}4 & 4 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T + \begin{bmatrix}-2\end{bmatrix}\right) \approx 0.9999$

  $\dfrac{\partial J}{\partial W_{11}^{(2)}} = \delta_1^{(l+1)} a_1^{(2)} = -0.1038 \cdot 0.9999 \approx -0.1038$
- $\delta_1^{(2)} \to \sum_{i=1}^{s_{l+1}} \delta_i^{(l+1)} W_{ij}^{(l)} \sigma'\left(z_j^{(l)}\right) = \sum_{i=1}^{2} \delta_i^3 W_{i1}^2 \sigma'\left(z_1^2\right) = \delta_1^3 W_{11}^2 \sigma'\left(z_1^2\right) + \delta_2^3 W_{21}^2 \sigma'\left(z_1^2\right)$

  $\delta_1^3 = -0.1038$

  $W_{11}^2 = 4$

  $\sigma'\left(z_1^2\right) = \sigma'\left(\begin{bmatrix}4 & 4 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T + \begin{bmatrix}-2\end{bmatrix}\right) = \sigma'\left(13\right) = \sigma(13) \cdot \left(1 - \sigma(13)\right) \approx 0$

  $\delta_2^3 = -\left(0 - 0.9547\right)\left(0.9547 \cdot \left(1 - 0.9547\right)\right) \approx 0.0413$

  $W_{21}^2 = 1$

  $\sigma'\left(z_1^2\right) = \sigma'\left(\begin{bmatrix}-4 & -4 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T + \begin{bmatrix}6\end{bmatrix}\right) = \sigma'(-3) = 0.0474 \cdot \left(1 - 0.0474\right) = 0.0452$

  $\delta_1^{(2)} = (\ldots) \cdot 0 + 0.0413 \cdot 1 \cdot 0.0452 = 0.0018$
- $a_1^1 = 1$

  $\dfrac{\partial J}{\partial W_{11}^{(1)}} = \delta_1^{(1)} a_1^1 = 0.0452$

2

1. Forward propagation:

$$
\begin{aligned}
a^{(1)} &= (1,0)^T \\
z^{(2)} &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \\
a^{(2)} &= \begin{bmatrix} 0.8808 \\ 0.8808 \end{bmatrix} \\
z^{(3)} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 0.8808 \\ 0.8808 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} 3.64 \end{bmatrix} \\
a^{(3)} &= \begin{bmatrix} 0.9745 \end{bmatrix}
\end{aligned}
$$

   Backward propagation:

$$
\begin{aligned}
\delta^{(3)} &= \frac{\partial J}{\partial b^{(2)}} = -\left(1 - 0.9745\right) 0.9745 \left(1 - 0.9745\right) = [-0.000634] \\
\delta^{(2)} &= \frac{\partial J}{\partial b^{(1)}} = \begin{bmatrix} \delta_1^3 \cdot W_{11}^{(2)} \cdot \sigma'\left(z_1^{(2)}\right) \\ \delta_1^{(3)} \cdot W_{12}^{(2)} \cdot \sigma'\left(z_2^{(2)}\right) \end{bmatrix} = \begin{bmatrix} -0.000634 \cdot 1 \cdot (0.8808 \cdot (1 - 0.8808)) \\ -0.000634 \cdot 2 \cdot (0.08808 \cdot (1 - 0.8808)) \end{bmatrix} = \begin{bmatrix} -0.000067 \\ -0.00013 \end{bmatrix} \\
\frac{\partial J}{\partial W^{(2)}} &= \begin{bmatrix} \delta_1^{(3)} a_1^{(2)} & \delta_1^{(3)} a_2^{(2)} \end{bmatrix} = \begin{bmatrix} -0.00063 \cdot 0.8808 & -0.00063 \cdot 0.8808 \end{bmatrix} = \begin{bmatrix} -0.00055 & -0.00055 \end{bmatrix} \\
\frac{\partial J}{\partial W^{(1)}} &= \begin{bmatrix} \delta_1^{(2)} \cdot a_1^{(1)} & \delta_1^{(2)} \cdot a_2^{(1)} \\ \delta_2^{(2)} \cdot a_1^{(1)} & \delta_2^{(2)} \cdot a_2^{(1)} \end{bmatrix} = \begin{bmatrix} -0.000067 \cdot 1 & -0.000067 \cdot 0 \\ -0.00013 \cdot 1 & -0.00013 \cdot 0 \end{bmatrix} = \begin{bmatrix} -0.000067 & 0 \\ -0.00013 & 0 \end{bmatrix}
\end{aligned}
$$

2. Forward propagation:

$$
\begin{aligned}
a^{(1)} &= (0,1)^T \\
z^{(2)} &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \\
a^{(2)} &= \begin{bmatrix} .9525 \\ .9525 \end{bmatrix} \\
z^{(3)} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} .9525 \\ .9525 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} 3.8575 \end{bmatrix} \\
a^{(3)} &= \begin{bmatrix} .9793 \end{bmatrix}
\end{aligned}
$$

   Backward Propagation:

$$
\begin{aligned}
\delta^{(3)} &= \frac{\partial J}{\partial b^{(2)}} = -\left(0 - a^{(3)}\right)\left(a^{(3)}\left(1 - a^{(3)}\right)\right) = [0.01985] \\
\delta^{(2)} &= \frac{\partial J}{\partial b^{(1)}} = \begin{bmatrix} \delta^{(3)} W_{11}^{(2)} \sigma'\begin{pmatrix} z_1^{(2)} \end{pmatrix} \\ \delta^{(3)} W_{12}^{(2)} \sigma'\begin{pmatrix} z_2^{(2)} \end{pmatrix} \end{bmatrix} = \begin{bmatrix} 0.01985 \cdot 1 \cdot 0.9525 \cdot (1 - 0.9525) \\ 0.01985 \cdot 2 \cdot 0.09525 \cdot (1 - 0.9525) \end{bmatrix} = \begin{bmatrix} 0.000898 \\ 0.001796 \end{bmatrix} \\
\frac{\partial J}{\partial W^{(2)}} &= \begin{bmatrix} \delta^{(3)} \cdot a_1^{(2)} & \delta^{(3)} \cdot a_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0.01985 \cdot 0.9525 & 0.01985 \cdot 0.09525 \end{bmatrix} = \begin{bmatrix} 0.01891 & 0.01891 \end{bmatrix} \\
\frac{\partial J}{\partial W^{(1)}} &= \begin{bmatrix} \delta_1^{(2)} \cdot a_1^{(1)} & \delta_1^{(2)} \cdot a_2^{(1)} \\ \delta_2^{(2)} \cdot a_1^{(1)} & \delta_2^{(2)} \cdot a_2^{(1)} \end{bmatrix} = \begin{bmatrix} 0.000898 \cdot 0 & 0.000898 \cdot 1 \\ 0.001796 \cdot 0 & 0.001796 \cdot 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.000898 \\ 0 & 0.001796 \end{bmatrix}
\end{aligned}
$$

Update To do the update step, we just average the respective terms, multiply by the learning rate, and subtract.

$$\begin{aligned} W^{(1)} &= W^{(1)} - \frac{\alpha}{2}\left(\begin{bmatrix} -0.000067 & 0 \\ -0.00013 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0.000898 \\ 0 & 0.001796 \end{bmatrix}\right) \\ &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \frac{0.2}{2}\begin{bmatrix} -0.000067 & 0.000898 \\ -0.00013 & 0.001796 \end{bmatrix} \\ &= \begin{bmatrix} 1.00001 & 1.99991 \\ 3.00001 & 3.99982 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} b^{(1)} &= b^{(1)} - \frac{\alpha}{2}\left(\begin{bmatrix} -0.000067 \\ -0.00013 \end{bmatrix} + \begin{bmatrix} 0.000898 \\ 0.001796 \end{bmatrix}\right) \\ &= \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{0.2}{2}\begin{bmatrix} 0.000831 \\ 0.001666 \end{bmatrix} \\ &= \begin{bmatrix} 0.999917 \\ -1.00017 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} W^{(2)} &= W^{(2)} - \frac{\alpha}{2}\left(\begin{bmatrix} -0.00055 & -0.00055 \end{bmatrix} + \begin{bmatrix} 0.01891 & 0.01891 \end{bmatrix}\right) \\ &= \begin{bmatrix} 1 & 2 \end{bmatrix} - \frac{0.2}{2}\begin{bmatrix} 0.01836 & 0.01836 \end{bmatrix} \\ &= \begin{bmatrix} 0.99814 & 1.99814 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} b^{(2)} &= b^{(2)} - \frac{\alpha}{2}\left([-0.000634] + [0.01985]\right) \\ &= [1] - \frac{0.2}{2}\left(0.019216[]\right) \\ &= [0.980784] \end{aligned}$$

4

Overfitting will be a problem for small training sets and models with a large number of parameters.

7

(a)

```
for l in range(len(nn_structure) - 1, 0, -1):
    # W[l] += -alpha * (1.0/N * tri_W[l]) # default
    W[l] += -alpha * (1.0/N * tri_W[l] + lamb/2*W[l]) # regularization
    b[l] += -alpha * (1.0/N * tri_b[l])
```

Figure 2: Changes for regularization

I also added lamb as an argument, of course.

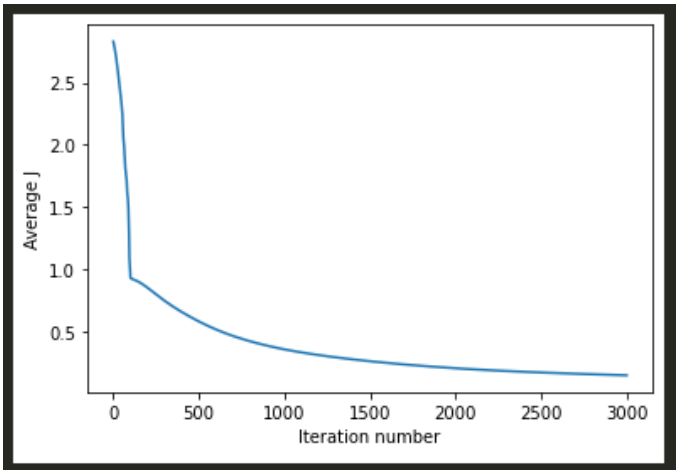

Figure 3: Error v. Iteration

The accuracy was 94.297% when lamb=1.

(b)

```
# Relu
def f(z):
    return np.maximum(0, z)


def f_deriv(z):
    return np.where(z < 0, 0, 1)
```

Figure 4: Changes for ReLu

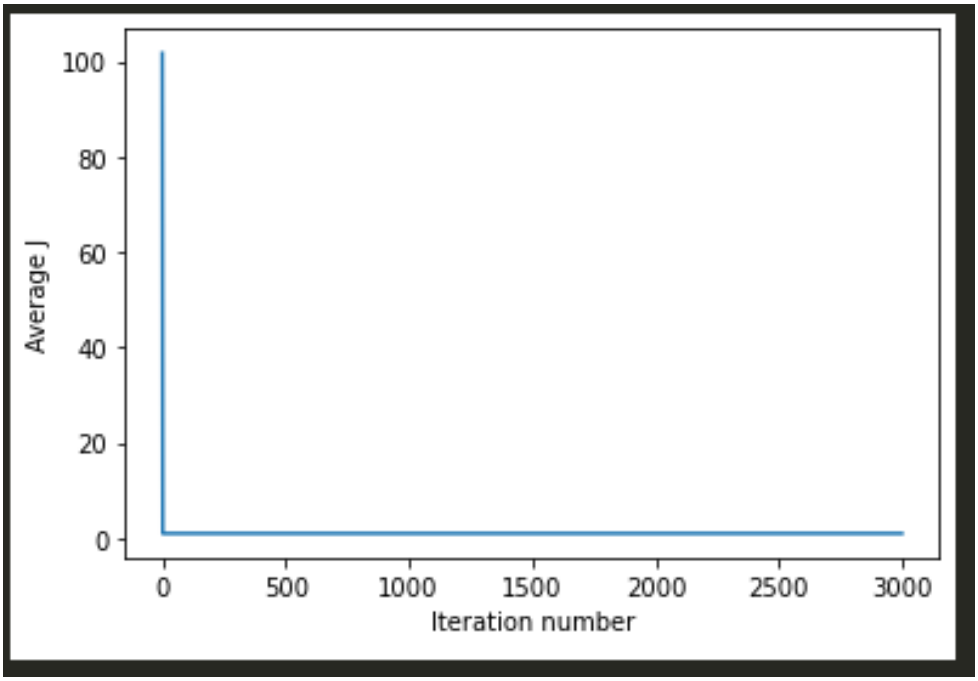

Figure 5: Error v. Iteration

The accuracy with the ReLu function was 9.75%.

(c)

```
# tanh
def f(z):
    p = np.exp(z)
    n = np.exp(-z)
    return (p-n)/(p+n)


def f_deriv(z):
    a = f(z)
    return 1-a*a
```

Figure 6: Changes for tanh

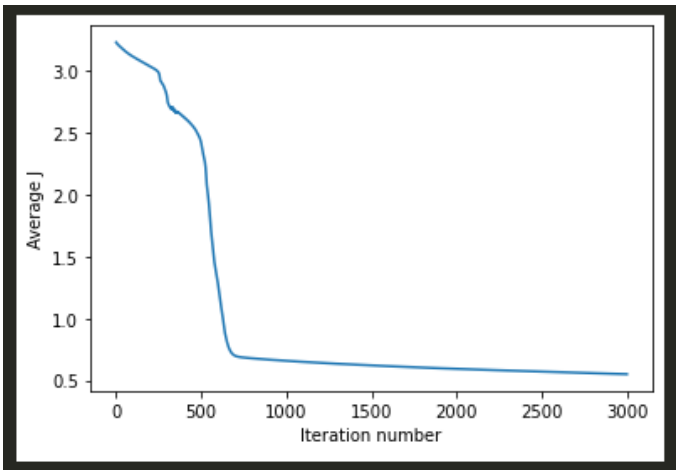

Figure 7: Error v. Iteration

The accuracy with the tanh function was 88.873%.

(d) ELU results in overflow.

```
# ELU with alpha=1
def f(z):
    return np.where(z<=0, np.exp(z)-1, z)


def f_deriv(z):
    return np.where(z<0, np.exp(z), 1)
```

Figure 8: Changes

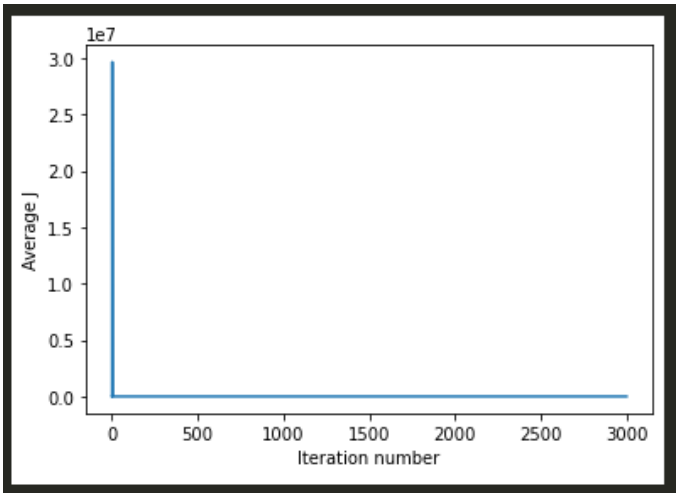

Figure 9: Error v. Iteration

The accuracy with the ELU function (with overflow) was 9.04%.

(e)

```
W, b, avg_cost_func = train_nn(nn_structure, X_train, y_v_train, 6000)
```

Figure 10: Changes

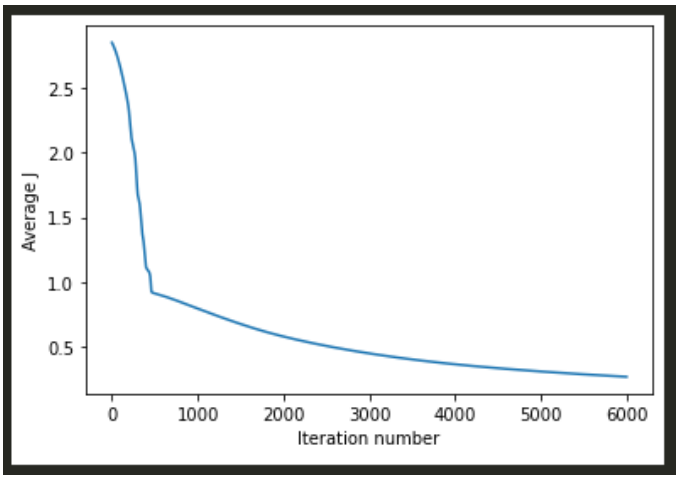

Figure 11: Erorr v. Iteration

Increasing the iterations to 6000 resulted in an accuracy of 92%.

(f)

```
nn_structure = [64, 30, 30, 10]
```

Figure 12: Changes

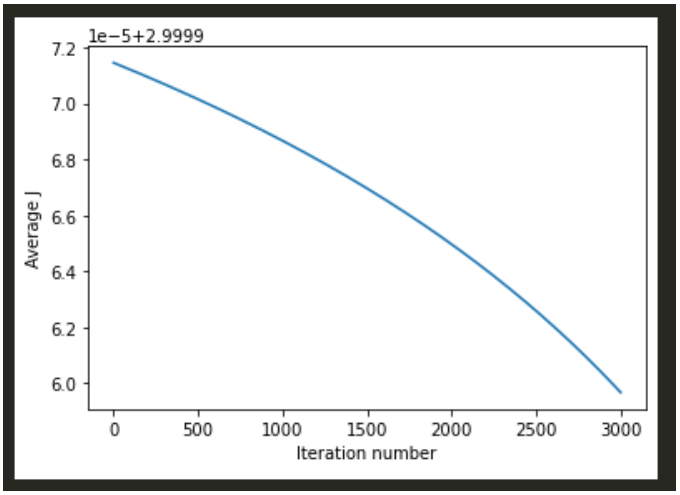

Figure 13: Error v. Iteration

```
nn_structure = [64, 10, 10, 10]
```

Figure 14: Changes

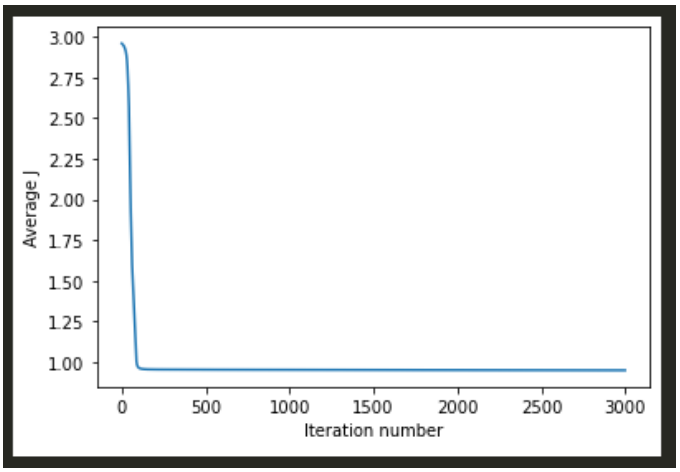

Figure 15: Error v. Iteration

Neither of the hidden layer changes had high accuracy: 5% and 9.3%, respectively. The highest accuracy of 94 % came from regularization with lamb=1.