Università della Svizzera italiana

Institute of Computing CI

Machine Learning **2021**

Student: Naga Venkata Sai Jitin Jami

Solution for Assignment 1

1. Polynomial Regression

1.1. Output vector

$$X \in \mathbb{R}^{n \times k} \tag{1}$$

$$\implies y \in \mathbb{R}^{n \times 1} \tag{2}$$

1.2. Forward pass

$$W \in \mathbb{R}^{k \times 1} \tag{3}$$

$$b \in \mathbb{R} \tag{4}$$

$$y_i = W^T \cdot X_i + b \tag{5}$$

$$y = W^T \cdot X + b \tag{6}$$

1.3. Mean Squared Error

$$E = \frac{1}{n} \sum_{i=1}^{n} (t_i - y_i)^2 \tag{7}$$

$$E = \frac{1}{n} \sum_{i=1}^{n} (t_i - (W^T \cdot X_i + b))^2 \tag{8}$$

1.4. Derivative w.r.t weights

$$\frac{dE}{dW} = \frac{2}{n} \sum_{i=1}^{n} X_i^T ((W^T \cdot X_i + b) - t_i) \tag{9}$$

1.5. Weight update

$$W_{t+1} = W_t - \eta \frac{dE}{dW_t} \tag{10}$$

1.6. Perceptron

Follow is the code snippet for to solve the regression problem stated with a single perceptron layer:

```
import numpy as np
X = np.array([[6,1,-11],[-5,6,7],[-5,-11,1],
              [-11,3,5],[-5,-2,1],[5,-6,-7],
              [2,0,2],[4,-9,-11],[9,-7,-9],[-8,4,-8]])
t = np.array([0,1,0,0,1,1,1,0,0,1])
w = np.array([-0.1,-0.3,-0.2])
b = 2
y = np.dot(X,w) + b
for i in range(len(X)):
    activation = np.dot(X[i],w) + b
    prediction_for_sample = 1
    if activation<0:
        prediction_for_sample = 0
    err = t[i] - prediction_for_sample
    print(err)
    w = w + 0.02*X[i]*err
    b = b + 0.02*err
```

The final weights:

w1	-0.16
w2	0.16
w3	0.3
b	1.9

Table 1: Final weights

1.7. Backpropogation and Gradient Descent

The non-linear nature of the MLP infrastructure makes updating of the variables a slightly complex procedure. We use the idea of backpropogation to calculate the partial derivatives of all the variables (weight matrices and bias vectors) with the help of a basic rule of partial differentiation: the chain rule. Once we have the partial derivatives, we use them to update the variables with gradient descent optimization technique. The idea behind gradient descent is, in a minimizing problem (as is the case here, we are trying to minimize the error) we find the direction of descent with the help of gradient (calculated by backpropogation) and move in that direction to reach the minima.

2. Simple Machine Learning Framework

The solution to this section can be found in the python file in the same folder by the name $framework.py$.

3. MNIST

Upon using the MLP infrastructure set up in the previous section to solve the MNIST classification problem, we get this (figure) final plot as the solution.

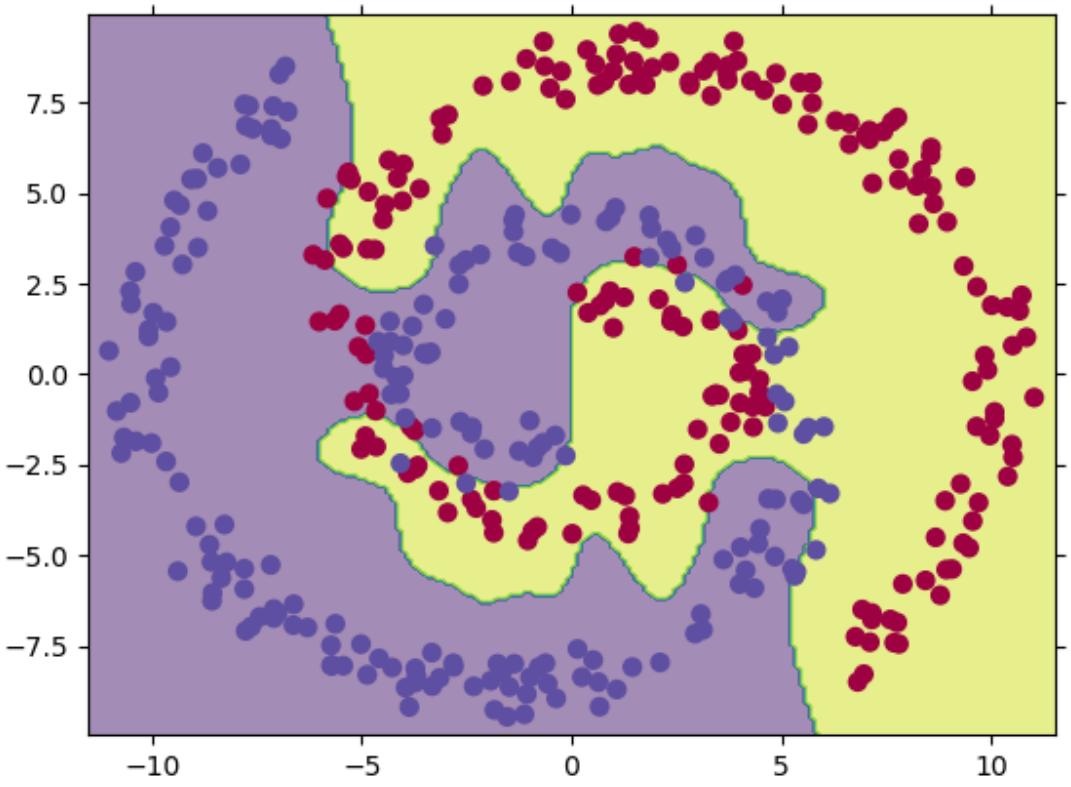

Figure 1: Final plot

4. References

- https://eli.thegreenplace.net/2018/backpropagation-through-a-fully-connected-layer/
- https://sgugger.github.io/a-simple-neural-net-in-numpy.html