Neural Network Training

Train a Neural network in Tensorflow

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.laters import Dense

model = Sequention ([
  Dense(units=25, activation='sigmoid'),
  Dense(units=15, activation='sigmoid'),
  Dense(units=1, activation='sigmoid'),
])

from tensorflow.keras.losses import BinaryCrossentropy

model.compile(loss=BinaryCrossentropy())
model.fit(X, Y, epochs=100)
```

- **Epoch:** number of steps (interartions) in Gradient Descent.

- **Step 1:** Define the model by specifying the layers and their configurations in `TensorFlow`.
- **Step 2:** Compile the model by selecting a loss function, specifically `BinaryCrossentropy` `loss` function.
- **Step 3:** Use the `fit` function to train the model on the dataset `(X, Y)` while specifying the number of epochs for the training process.

Training Details

Model Training Step

1. **Specify how to compute output given input x and parameters w, b (define model):** $f_{\vec{w},b}(\vec{x}) = ?$
2. **Specify `loss` and `cost`):**

$$L(f_{\vec{w},b}(\vec{x}), y)$$

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=0}^{m} L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})$$

- Logistic loss (compare **prediction** vs **target**) is known as **Binary cross entropy**

3. **Train on data to minimize $J(\vec{w}, b)$ (Gradient Descent)**

$$\text{repeat} \begin{cases} w_j^{[l]} = w_j^{[l]} - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b) \\ b^{[l]} = b^{[l]} - \alpha \frac{\partial}{\partial b} J(\vec{w}, b) \end{cases}$$

- Compute derivatives for gradient descent using **“backpropagation”**

Example:

	Logistic Regression	Neural network
1	`f_x = sigmoid(np.dot(w, x) + b)`	`model = Sequential([ Dense(), Dense(), Dense()])`
2	logistic loss: `loss = -y*np.log(f_x) - (1-y)*log(1-f_x)` mean squared error: `loss = -y*np.log(f_x) - (1-y)*log(1-f_x)`	binary cross entropy: `model.compile(loss=BinaryCrossentropy())` mean squared error: `model.compile(loss=MeanSquaredError())`
3	`w = w - alpha*dj_dw` `b = b - alpha*dj_db`	`model.fit(X, Y, epochs=100)`