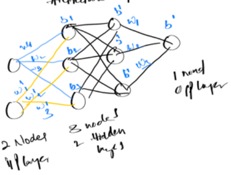
Now lets apply our Pattern to make the fun example Predict custom data set Person weight, Person height

defendnt value

Heart decease (1 or 0)

Architecture Of Our Neural Network



Build every thing

Implimentation :-

Step 1: Define how many layer we wanna work with like only I/P is excluded so layer = 3

Step 2: Now number of nodes that you need for each layer nodes = [2, 3, 3, 1]

Step 3: Initialize the weight and biases at Random

W^L = np.random.randn(n[L] * n[L-1])

b^L = np.random.randn(n[L] * 1)

Step 4: data prep key points make everything 2D matrix

Step 5: Activation function we are using sigmoid activation which is $\frac{1}{1+e^{-x}}$

Step 6: Start with feed forward function

```
→ we start with our I/P layer A0
→ Z = W1 @ A0 + b(n)
  A1 = sigmoid (z)
  Z^n = W n @ A(n-1) + b(n)
  A^n = sigmoid (Z^n)
```

Step 7: Now Calculation of loss / cost

```
loss = -(y * np.log(y_hat)) + (1-y) * np.log(1-y)
m = y_hat.reshape(-1).shape[0]
summed_loss = (1/m) * np.sum(losses, axis=1)
```

Step 8: Yea fun Begins

Back Propagation :-

```
we start with
∂C_∂Z3 = 1/m * (A3 - Y) → Check intuition
assert ∂C_∂Z3.shape == (n[3], m)

∂Z3_dW3 = A2
assert dZ3_dW.shape == (n[2], m)
∂C_dW3 = ∂C_dZ3 @ ∂Z3_dW3.T

∂Z3_A2 = W3
∂C_∂A2 = W3.T @ ∂C_∂Z3
```

Step 9: back prop by 2

```
∂A2_∂Z2 = A2 * (1 - A2) → essential derivative sigmoid function
∂C_∂Z2 = prop * ∂A2_∂Z2

∂Z2_W2 = A1
∂C_dW2 = ∂C_∂Z2 @ ∂Z2_dW2.T
∂C_∂b2 = np.sum(∂C_W2, axis)

∂Z2_∂A1 = W2
∂C_∂A1 = W2.T @ ∂C_∂Z2
```

Step 10: back prop 1 :-

```
∂A1_∂Z1 = A1 * (1 - A1)
∂C_∂Z1 = prop_∂C_A1 * ∂A1_∂Z1

∂Z1_∂W1 = A0
∂C_W1 = ∂C_∂Z1 @ ∂Z1_W1.T
∂Cb1 = np.sum
```

Step 11: Train model !

```
for e in epoch range():
    → ŷ, cache = feed_fwd(A0)
    → error
    → cost
    → back prop 3
    → back prop 2
    → back prop 1
Adjust weights
    W3 = W3 - (alpha * ∂C_W3)
    W2 = W2 - (alpha * ∂C_W2)
    W1 = W1 - (alpha * ∂C_W1)
    b3 = b3 - (alpha * ∂C_∂b3)
    b2 = b2 - (alpha * ∂C_∂b2)
    b1 = b1 - (alpha * ∂C_∂b1)

Call train()
```