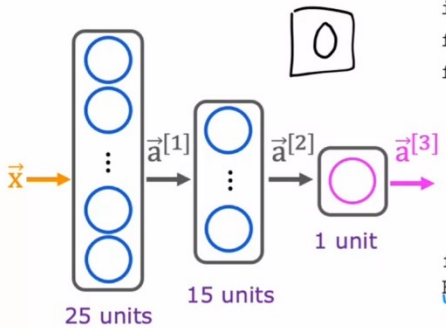
Subject: / /

Train a Nueral Network:

Train a Neural Network in TensorFlow




```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
   model = Sequential([
     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)
```

① ② ③

epochs: number of steps in gradient descent

Given set of (x,y) examples

How to build and train this in code?

Training Details:

① Specify how to compute output given input x and parametres w,b

$f_{\vec{w},b}(\vec{x}) = ?$

② Specify loss and Cost

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

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

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

For a nueral network,

① model = Sequential ....

② model.compile (loss = BinaryCrossentropy())

③ model.fit (X, Y, epochs = 100)

Subject: / /

Loss and Cost function:

Classification:

$$L(f(\vec{x}), y) = -y \log(f(\vec{x})) - (1-y)\log(1-f(\vec{x}))$$ → Binary cross enthropy

```
model.compile(loss = BinaryCrossenthropy())
```

Regression:

```
model.compile(loss = MeanSquaredError
```

$$J(W,B) = \frac{1}{m}\sum_{i=0}^{m} L(f(\vec{x}^{(i)}), y^{(i)})$$

$W^{[3]}, W^{[2]}, W^{[3]}$ $\quad$ $b^{[3]}, b^{[2]}, b^{[3]}$

3. Gradient descent:

Repeat {

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

$$b_j^{[l]} = b_j^{[l]} - \alpha \frac{\partial}{\partial b_j} J(\vec{w}, b)$$

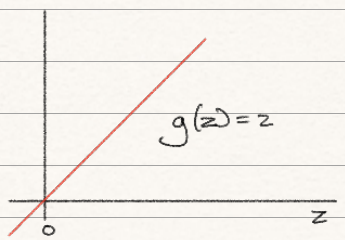
Backpropagation to compute the derivatives

}

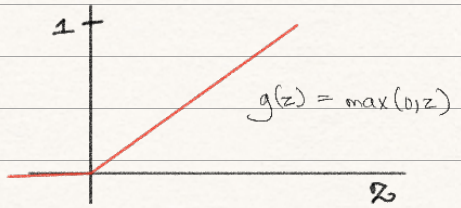
Alternatives to Sigmoid activation:

Subject: / /

linear activation function



ReLU - Rectified linear unit



How to choose?

- Depends on target
- For Binary classification, use sigmoid
- For Regression, use Linear activation function if y is +/-.
  , use ReLU if y is 0/+.
- For hidden layers, use ReLU

Multi-class classification:

↳ More than one class/category

Soft-max:

↳ Generalization of logistic Regression

X $z_1 = \vec{w}_1 \cdot \vec{x} + b_1$

O $z_2 = \vec{w}_2 \cdot \vec{x} + b_2$

□ $z_3 = \vec{w}_3 \cdot \vec{x} + b_3$

Δ $z_4 = \vec{w}_4 \cdot \vec{x} + b_4$

$$p(y=1|\vec{x}) = a_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}+e^{z_4}}$$

$$\vdots$$

$$P(y=4|\vec{x}) = a_4 = \frac{e^{z_4}}{e^{z_1}+e^{z_2}+e^{z_3}+e^{z_4}}$$

Subject: / /

$y = 1, 2, 3, \cdots N$

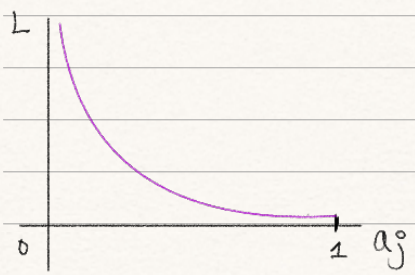
$z_j = \vec{w_j} \cdot \vec{x} + b_j \qquad j = 1, \cdots, N$

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} = P(y=j \mid \vec{x})$$

$a_1 + a_2 + \cdots + a_N = 1$

Cost:

$$\text{loss}(a_1, \cdots, a_N, y) = \begin{cases} -\log a_1 & \text{if } y=1 \\ -\log a_2 & \text{if } y=2 \\ \vdots & \vdots \\ -\log a_3 & \text{if } y=N \end{cases}$$



- Loss is more if a_j is small
- Motivates model to make a_j as close as possible to 1.

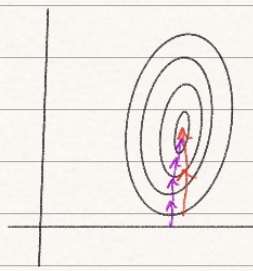
loss = SparseCategoricalCrossentropy()

Subject: / /

Multiclass is classifying single piece, y is scalar
Multi-label is classifying many pieces, y is vector

$$J(w,b) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} 1\{y^{(i)}==j\}\log\frac{e^{z_j^{(i)}}}{\sum_{k=1}^{N} e^{z_k^{(i)}}}\right.$$

Advanced propogation:



α is big } done automatically
α is small

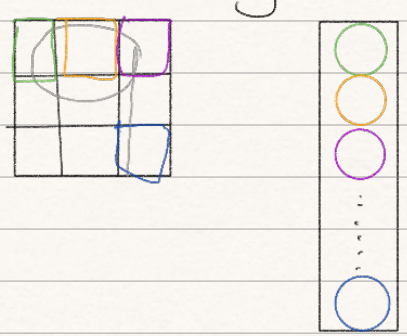
as gradient descent is in same direction, why not make α bigger? Adam does this.

Adam:

- Adaptive Moment estamation
- α is diff for each parametre

Additional layer types:

Convolutional layer



Subject: / /

- Helps avoid overfitting

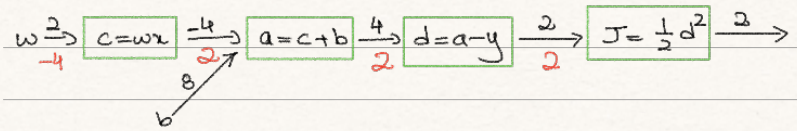
Back propagation: ← (Right — left)

Computation graph:

$\vec{x} \rightarrow \boxed{} \xrightarrow{a}$ $\quad w=2 \quad b=8 \quad x=-2 \quad y=2$

$J(w,b) = \frac{1}{2}(a-y)^2$

→ Forward
← back prop



$\frac{\partial J}{\partial d} = 2$

$d \uparrow 0.001 \quad J \uparrow 0.002$

$\frac{\partial J}{\partial a} = 2$

$a \uparrow 0.001 \quad d \uparrow 0.001 \quad J \uparrow 0.002$

$a = 4.001 \quad d = 2.001$

$\frac{\partial J}{\partial a} = \frac{\partial d}{\partial a} \times \frac{\partial J}{\partial d}$

$= 1 \times 2$

$\frac{\partial J}{\partial c} = 2$

$c \uparrow 0.001 \quad a \uparrow 0.001$

$J \uparrow 0.002$

$\frac{\partial J}{\partial c} = \frac{\partial a}{\partial c} \times \frac{\partial J}{\partial a}$

$= 1 \times 2$

$\frac{\partial J}{\partial b} = 2$

$b \uparrow 0.001 \quad a \uparrow 0.001$

$J \uparrow 0.002$

$\frac{\partial J}{\partial b} = \frac{\partial a}{\partial b} \times \frac{\partial J}{\partial a}$

$= 1 \times 2$

$\frac{\partial J}{\partial w} = -4$

$w \uparrow 0.001 \quad c = -4.002$

$w = 2.001 \quad c \downarrow 2 \times 0.001$

$c \uparrow -2 \times 0.001$

$J \uparrow -4 \times 0.001$

$\frac{\partial J}{\partial w} = \frac{\partial c}{\partial w} \times \frac{\partial J}{\partial c}$

$= -2 \times 2$

Subject: / /

$w \xrightarrow{3} a = 2 + 3w \xrightarrow{11} J = a^2 \xrightarrow{121} J$

$\frac{\partial w}{\partial J} = 66$ $\qquad$ $\frac{\partial J}{\partial a} = 22$

$a \uparrow 0.001$ $\qquad$ $J \uparrow 0.022 \Rightarrow 22 \times 0.002$

$w \uparrow 0.001$ $\qquad$ $a \uparrow 0.003$ $\qquad$ $a = 11.001$ $\qquad$ $J = 121.022$

3×0.001