Day-29, Oct-31, 2024.

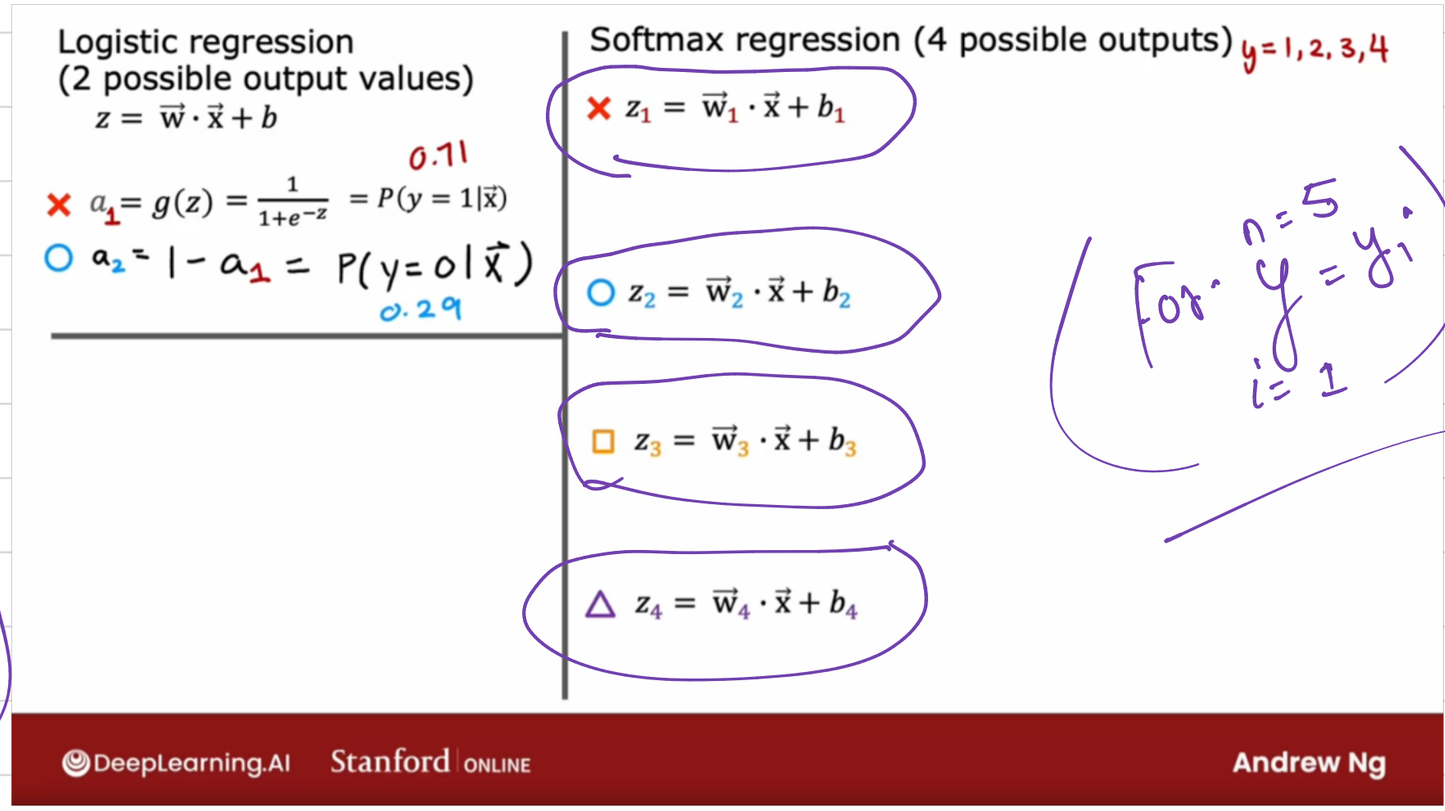
Softmax Activation functions.

we have only 2 possible Output values in Logistic Regression

$$z = \vec{w} \cdot \vec{x} + b$$

$$a_1 = g(z) \Rightarrow \frac{1}{1+e^{-z}} = p(y=1|\vec{x})$$

Suppose $y=1$ given then if a_1 is 0.7 then $a_2 = 1 - a_1 \Rightarrow p(y=0|\vec{x})$ (0.3)



$a_1 = 0.7$
$a_2 = 0.3$
$P \rightarrow 1.0$

So for $y = 1, 2, 3, 4$

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

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

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

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

So, a_1, a_2, a_3 & a_4 are

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

$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \Rightarrow p(y=2|\vec{x})$ ... $a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} \Rightarrow p(y=4|\vec{x})$

Logistic regression
(2 possible output values)

$z = \vec{w} \cdot \vec{x} + b$

✗ $a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$ (0.71)

○ $a_2 = 1 - a_1 = P(y=0|\vec{x})$ (0.29)

Softmax regression (4 possible outputs) $y = 1, 2, 3, 4$

✗ $z_1 = \vec{w}_1 \cdot \vec{x} + b_1$ $\quad a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=1|\vec{x})$

○ $z_2 = \vec{w}_2 \cdot \vec{x} + b_2$ $\quad a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=2|\vec{x})$

□ $z_3 = \vec{w}_3 \cdot \vec{x} + b_3$ $\quad a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=3|\vec{x})$

△ $z_4 = \vec{w}_4 \cdot \vec{x} + b_4$ $\quad a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=4|\vec{x})$

DeepLearning.AI Stanford ONLINE Andrew Ng

Logistic regression (2 possible output values)

$z = \vec{w} \cdot \vec{x} + b$

✗ $a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$ 0.71

○ $a_2 = 1 - a_1 = P(y=0|\vec{x})$ 0.29

Softmax regression (4 possible outputs) $y = 1, 2, 3, 4$

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

$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$ ✗ ○ □ △

$= P(y=1|\vec{x})$ 0.30

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

$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$

$= P(y=2|\vec{x})$ 0.20

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

$a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$

$= P(y=3|\vec{x})$ 0.15

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

$a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$

$= P(y=4|\vec{x})$ 0.35

Logistic regression (2 possible output values)

$z = \vec{w} \cdot \vec{x} + b$

✗ $a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$ 0.71

○ $a_2 = 1 - a_1 = P(y=0|\vec{x})$ 0.29

Softmax regression (N possible outputs) $y = 1, 2, 3, \ldots, N$

$z_j = \vec{w}_j \cdot \vec{x} + b_j \quad j = 1, \ldots, N$

parameters $w_1, w_2, \ldots, w_N$ $b_1, b_2, \ldots, b_N$

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

Softmax regression (4 possible outputs) $y = 1, 2, 3, 4$

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

$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$ ✗ ○ □ △

$= P(y=1|\vec{x})$ 0.30

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

$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$

$= P(y=2|\vec{x})$ 0.20

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

$a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$

$= P(y=3|\vec{x})$ 0.15

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

$a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$

$= P(y=4|\vec{x})$ 0.35

DeepLearning.AI Stanford | ONLINE Andrew Ng

'Σ' of a_n must be equal to 1 because probability is always '1'.

Notation for Activation Function (SoftMax)

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

Recall.

$a_1 = g(z) \Rightarrow \frac{1}{1+e^{-z}}$

$\Rightarrow P(y=1|\vec{x})$

$a_2 = 1 - a_1 \Rightarrow P(y=0|\vec{x})$

$[loss = -y \log a_1 - (1-y) \log(1-a_1)]$

if $y=1$ — if $y=0$.

$J(\vec{w}, b) =$ average loss

Cost

Logistic regression

$z = \vec{w} \cdot \vec{x} + b$

$a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$

$a_2 = 1 - a_1 = P(y=0|\vec{x})$

$loss = -y \log a_1 - (1-y) \log(1-a_1)$

if $y=1$ — if $y=0$

$J(\vec{w}, b) =$ average loss

Softmax regression

DeepLearning.AI Stanford ONLINE Andrew Ng

For SoftMax Regression? So, we figure out & observe the Crossentropy loss (predicted distribution probability to the original).

Softmax Regression.

$$a_1 \Rightarrow \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_n}}$$

$$\Rightarrow p(y=1|\vec{x})$$

$$\vdots$$

$$a_N \Rightarrow \frac{e^{z_n}}{e^{z_1} + e^{z_2} + \cdots + e^{z_n}}$$

$$\Rightarrow p(y=N|\vec{x})$$

Cost

Logistic regression

$$z = \vec{w} \cdot \vec{x} + b$$

$$a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$$

$$a_2 = 1 - a_1 = P(y=0|\vec{x})$$

$$loss = -y \log a_1 - (1-y) \log(1-a_1)$$

if $y=1$; if $y=0$

$$J(\vec{w}, b) = \text{average loss}$$

Softmax regression

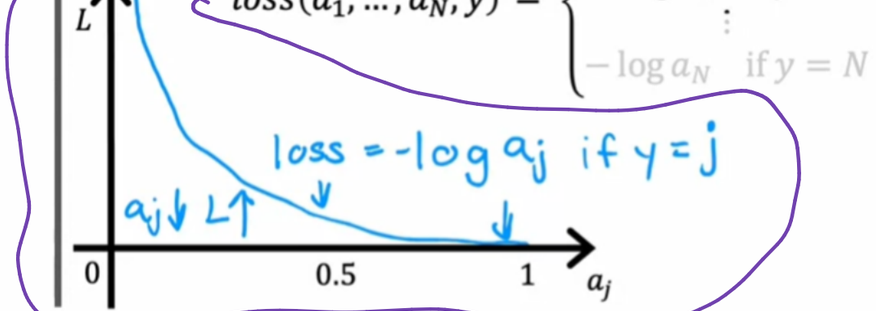
$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y=1|\vec{x})$$

$$\vdots$$

$$a_N = \frac{e^{z_N}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y=N|\vec{x})$$

Crossentropy loss

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

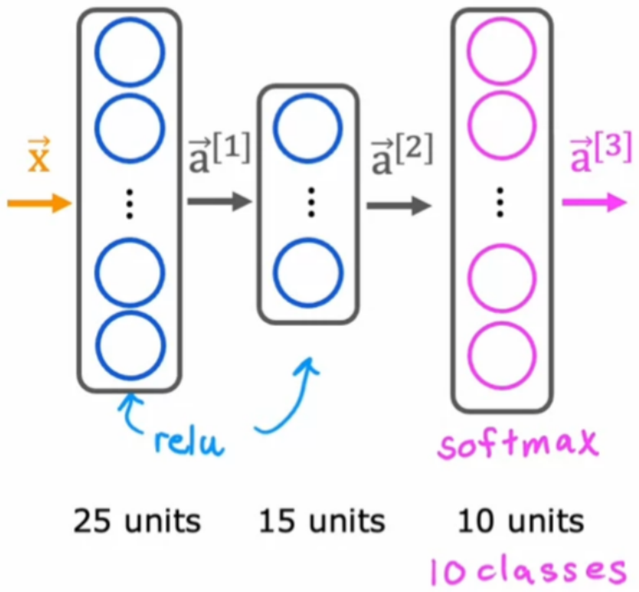


DeepLearning.AI Stanford ONLINE Andrew Ng

As you go near the '1' a_j it is $loss = -\log a_j$

but if go near the '0' it is $\log(1-a_j)$

ReLu used in 25 nodes and 15 nodes and they are hidden layers. Whereas Softmax is used in output layer having 10 nodes or the 10 classes →



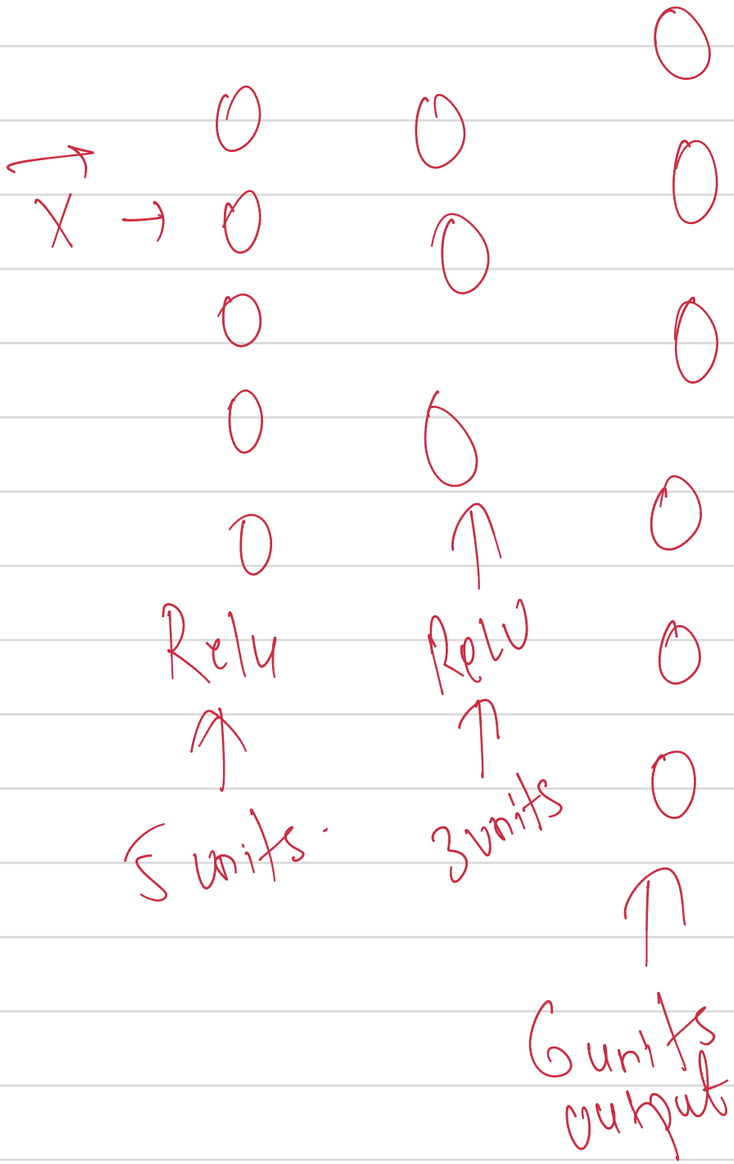
For $z_n = 5$

Output class 5

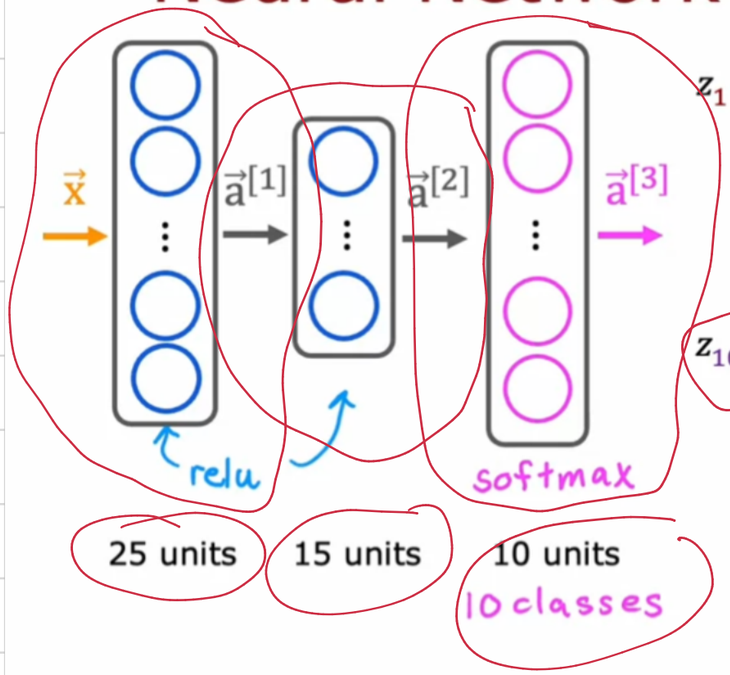
$$z_1 = \vec{w}_1 \cdot \vec{a}^{[2]} + b_1 \Rightarrow a_1 \Rightarrow \frac{e^{z_1}}{e^{z_1} + \dots + e^{z_5}} \Rightarrow p(y=1|\vec{x})$$

$$\vdots$$

$$z_5 \quad \vec{w}_5 \cdot \vec{a}^{[2]} + b_5 \Rightarrow a_5 \Rightarrow \frac{e^{z_5}}{e^{z_1} + \dots + e^{z_5}} \Rightarrow p(y=5|\vec{x})$$



Neural Network with Softmax output



$$z_1 = \vec{w}_1 \cdot \vec{a}^{[2]} + b_1 \qquad a_1 = \frac{e^{z_1}}{e^{z_1} + \cdots + e^{z_{10}}} = P(y = 1|\vec{x})$$

$$\vdots$$

$$z_{10} = \vec{w}_{10} \cdot \vec{a}^{[2]} + b_{10} \qquad a_{10} = \frac{e^{z_{10}}}{e^{z_1} + \cdots + e^{z_{10}}} = P(y = 10|\vec{x})$$

DeepLearning.AI Stanford ONLINE Andrew Ng

So, we add Superscript to denote the Layer of Neural Network whereas Subscript tells the index in the particular layer itself.

MNIST with softmax

① specify the model

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

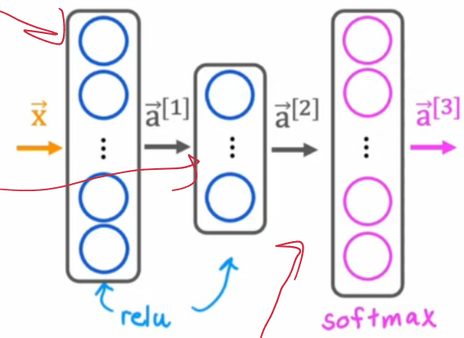
```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=10, activation='softmax')
                  ])
```

specify loss and cost

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

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

Neural Network with Softmax output



$z_1^{[3]} = \vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]}$ $\quad a_1^{[3]} = \frac{e^{z_1^{[3]}}}{e^{z_1^{[3]}} + \cdots + e^{z_{10}^{[3]}}}$

$= P(y = 1|\vec{x})$

$z_{10}^{[3]} = \vec{w}_{10}^{[3]} \cdot \vec{a}^{[2]} + b_{10}^{[3]}$ $\quad a_{10}^{[3]} = \frac{e^{z_{10}^{[3]}}}{e^{z_1^{[3]}} + \cdots + e^{z_{10}^{[3]}}}$

$= P(y = 10|\vec{x})$

logistic regression

$a_1^{[3]} = g(z_1^{[3]})$ $\quad a_2^{[3]} = g(z_2^{[3]})$

softmax

$\vec{a}^{[3]} = (a_1^{[3]}, \ldots a_{10}^{[3]}) = g(z_1^{[3]}, \ldots, z_{10}^{[3]})$

DeepLearning.AI Stanford | ONLINE Andrew Ng

Loss Function:

Sparse Categorical Cross Entropy.

$-\log(\hat{y})$ used in classification, dealing with multi-class classification Problem

Input → 1

Category [0, 10] → Sparse only selects or predict category 1.

takes [0 to 10]

&

only Sparse one value it Can be only one of the category for a given digit

MNIST with softmax

1. specify the model

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

2. specify loss and cost

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

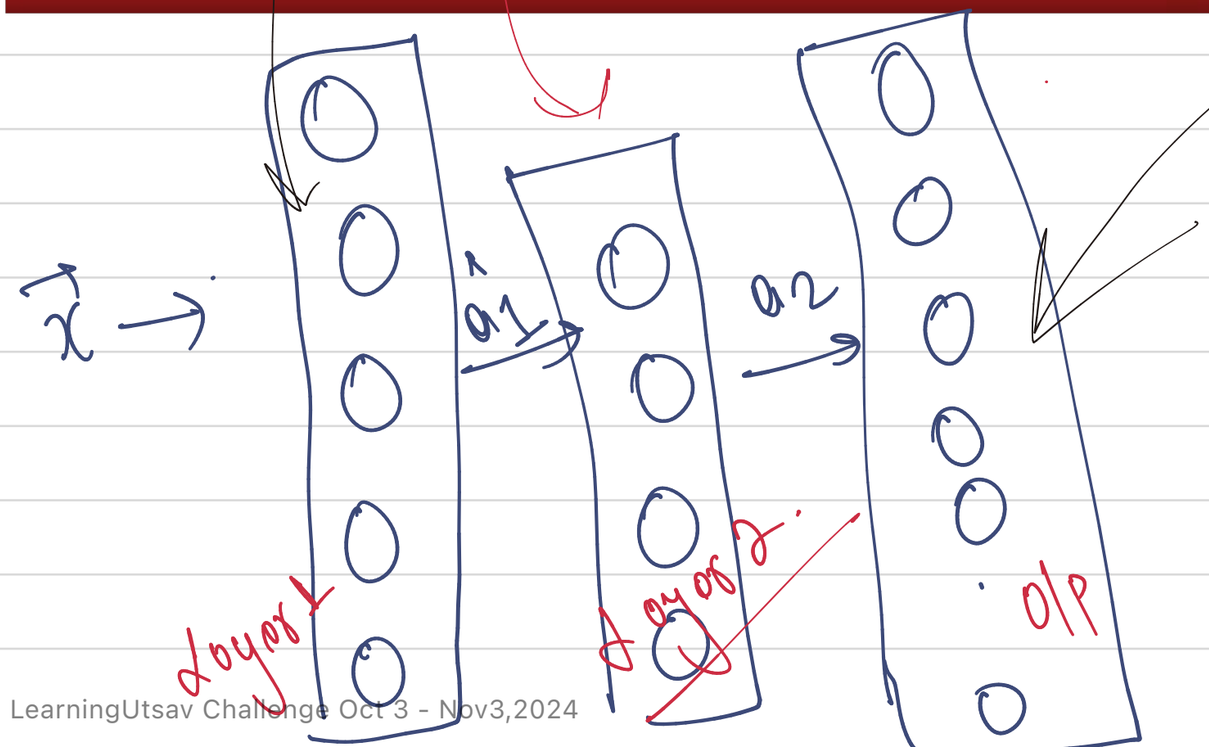
3. Train on data to minimize $J(\vec{w}, b)$

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu'),
  Dense(units=15, activation='relu'),
  Dense(units=10, activation='softmax')
                  ])
from tensorflow.keras.losses import
  SparseCategoricalCrossentropy
model.compile(loss= SparseCategoricalCrossentropy() )
model.fit(X,Y,epochs=100)
```

Note: better (recommended) version later.

Don't use the version shown here!

DeepLearning.AI Stanford ONLINE Andrew Ng



#. So, the main idea from the slides is that we use softmax for multi-class supported by SparseCategoricalCrossentropy (loss function).