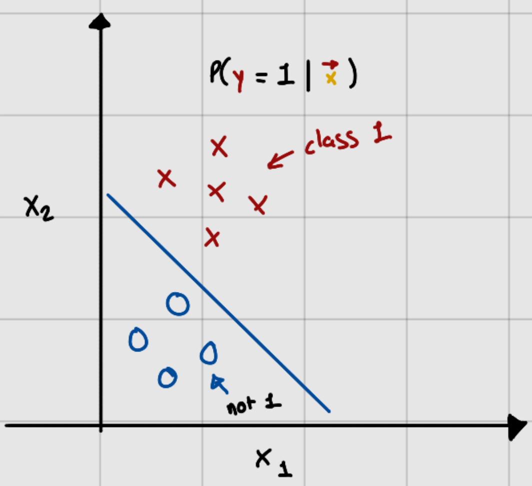
Multiclass

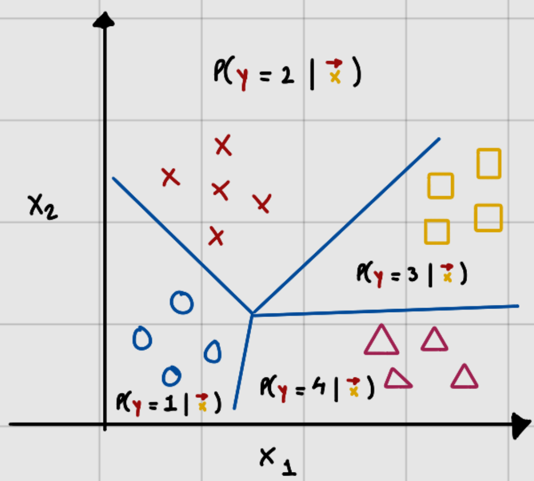
0 1 2 3 4 5 6 7 8 9

$y = 0$ 1 2 3 4 5 6 7 8 9

$\vec{x}$ 7 $y = 7$

multiclass classification problem:
target y can take on more than two possible values





Softmax

Logistic regression
(2 possible output values)

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

× $a_1 = g(z) = \dfrac{1}{1 + e^{-z}} = P(y = 1 \mid \vec{x})$ 0.71

○ $a_2 = 1 - a_1 = P(y = 0 \mid \vec{x})$ 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 $w_1, w_2, \ldots, w_n$
$b_1, b_2, \ldots, b_n$

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

note: $a_1 + a_2 + \ldots + a_n = 1$

Softmax regression (4 possible outputs)

$y = 1, 2, 3, 4$

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

$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y = 1 \mid \vec{x})$$ 0.30

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

$$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y = 2 \mid \vec{x})$$ 0.20

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

$$a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y = 3 \mid \vec{x})$$ 0.15

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

$$a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y = 4 \mid \vec{x})$$ 0.35

Cost

Logistic regression

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

$a_1 = g(z) = \dfrac{1}{1 - e^{-z}} = P(y=1 \mid \vec{x})$

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

$\text{loss} = -\underbrace{y \log a_1}_{\text{if } y=1} - (1-y)\underbrace{\log(a_2)}_{\text{if } y=0}$

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

Softmax regression

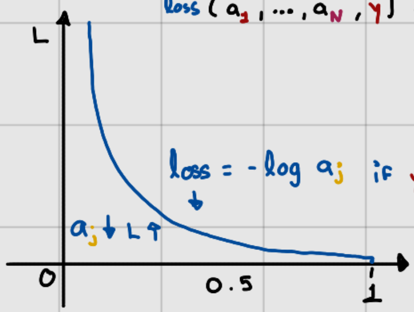
$a_1 = \dfrac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=1 \mid \vec{x})$

$\vdots$

$a_N = \dfrac{e^{z_N}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=N \mid \vec{x})$

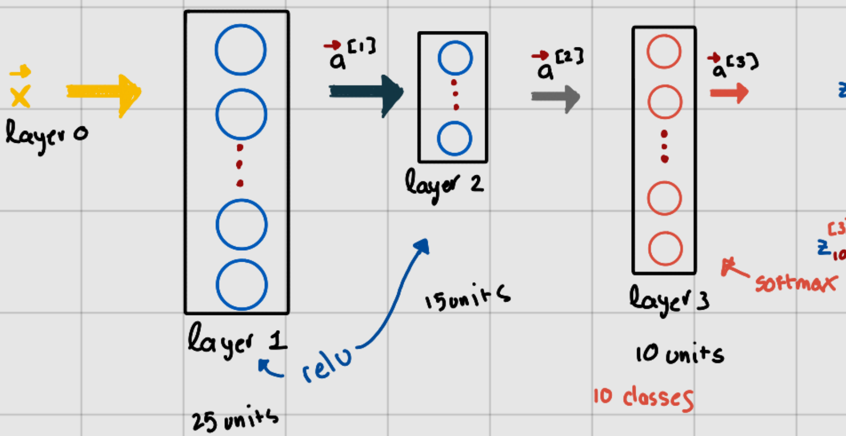
Cross entropy loss

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



$\text{loss} = -\log a_j$ if $y=j$

Neural Network with Softmax output



$z_1^{[3]} = \vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]}$ $\qquad a_1 = \dfrac{e^{z_1^{[3]}}}{e^{z_1^{[3]}} - \cdots + e^{z_{10}^{[3]}}} = P(y=1 \mid \vec{x})$

$\vdots$

$z_{10}^{[3]} = \vec{w}_{10}^{[3]} \cdot \vec{a}^{[2]} + b_{10}^{[3]}$ $\qquad a_{10} = \dfrac{e^{z_{10}^{[3]}}}{e^{z_1^{[3]}} - \cdots + e^{z_{10}^{[3]}}} = P(y=10 \mid \vec{x})$

logistic regression

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

Softmax

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

logistic regression

(more numerically accurate)

- model

```
model = Sequential([
    Dense(units = 25, activation = 'relu'),   # layer 1
    Dense(units = 15, activation = 'relu'),   # layer 2
    Dense(units = 25, activation = 'linear')  # layer 3
])
```

- loss

```
from tensorflow.keras.losses import BinaryCrossentropy
model.compile( ..., BinaryCrossentropy(from_logits = True))
```

- fit

```
model.fit(X, y, epoch = 100)
logit = model(X)
```

- predict

```
                          z
f_x = tf.nn.sigmoid(logit)
```