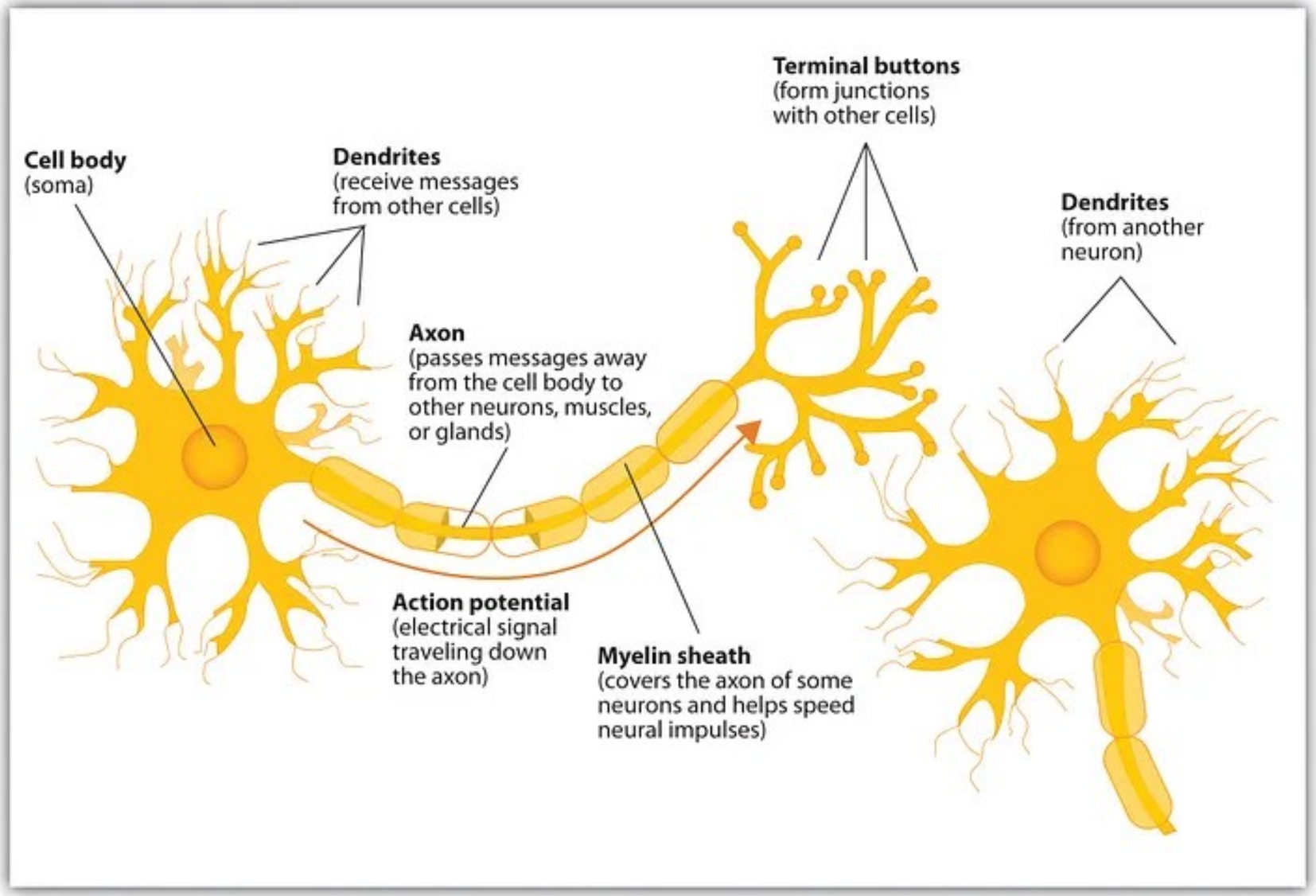
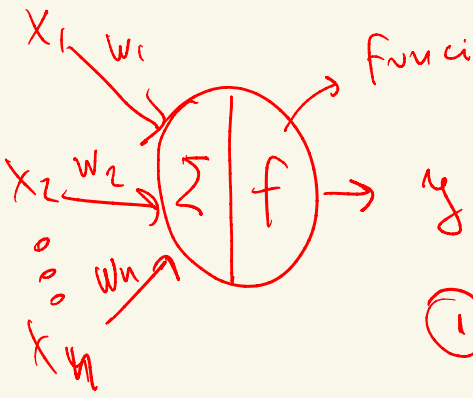
Redes neuronales artificiales

Dr. Alejandro Veloz

Comunicación en biología





función de activación

Una neurona

1943

① $D = \{(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)\}$

clase

imágenes

$\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$ entrada

$y \in \mathbb{R}$ salida

$$J = \frac{1}{N \cdot 2} \sum_{i=1}^{N} e_i^2$$

② Criterio

$$N(x_i) = \hat{y}_i \rightarrow y_i$$

$$J = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \qquad e_i = (y_i - \hat{y}_i)$$

$D = \{(\underline{x}_1, y_1), \dots, (\underline{x}_N, y_N)\}$

$$\frac{1}{2N}\sum_{i=1}^{N}\left(y_i - f(\underbrace{\underline{w}\,\underline{x}_i}_{\hat{y}_i})\right)^2$$

$$[w_1 \dots w_n]\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$w_1 x_1 + \dots + w_n x_n$$

$$\underset{N\times 1}{\underline{Y}} - \underset{N\times n}{\underline{X}}\,\underline{w}$$

$$\left\| \underline{Y} - \underline{X}\,\underline{w} \right\|_2$$

$$V \equiv \underline{Y} - \underline{X}\,\underline{w}$$

$$\hat{w} = \underset{w}{\operatorname{argmin}}\ \underbrace{l_2^2(V)}_{J}$$

función de costo

$\underline{X} \in \mathbb{R}^n$ Imagen como vector

$$\underline{X} = \begin{bmatrix} x_1, y_1 \\ x_2, y_2 \\ \vdots \\ x_N, y_N \end{bmatrix}$$

$$l_2(V) = \sqrt{\sum_{i=1}^{N} V_i}$$

$$= \sqrt{V'V}$$

$$l_2^2(V) = V'V$$

norma euclidiana

1) J convexa

2) J derivable

$$l_2^2(\underline{V}) = (Y - Xw)'(Y - Xw)$$

$N\times 1$ $\quad$ $N\times n$ $n\times 1$ $\quad$ $N\times 1$

$N\times 1$

$N\times 1$

$$J = \ell_2^2\left(y - f(xw)\right)$$ (función activación.)

$$= \left(y - f(xw)\right)^T \underbrace{\left(y - f(xw)\right)}_{\hat{y}}$$

$$\hat{w} = \underset{w}{\text{Argmin}}\ J$$

$$= \underset{w}{\text{Argmin}} \frac{1}{N_2} \| y - f(xw) \|_2^2$$

$f(x) = x$

$f(xw) = xw$

$$\underbrace{\frac{\partial J}{\partial w}}_{\text{Gradiente de } J \text{ wrt } w} = 0 \Rightarrow \frac{1}{N_2} \frac{\partial \| y - xw \|_2^2}{\partial w} = 0$$

$$\underbrace{\frac{2}{2N} x^T (y - xw)}_{\text{qué tamaño tiene}} \cdot = 0$$

$M \times 1$

$$x^T(y - xw) = 0$$

$$x^T y - x^T x w = 0$$

$$x^T y = x^T x w$$

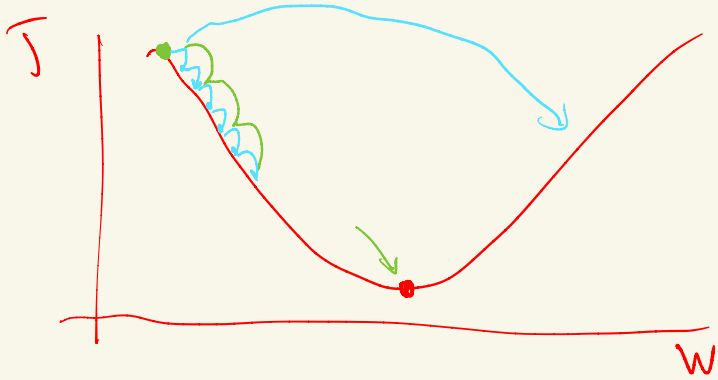
$$w = (x^T x)^{-1} x^T y$$

$$J = \frac{1}{2N} \| y - f(xw) \|_2^2$$

$$\frac{\partial J}{\partial w} = \frac{1}{N} (y - f(xw)) \frac{\partial f}{\partial w} \cdot \boxed{\frac{\partial xw}{\partial w}}$$

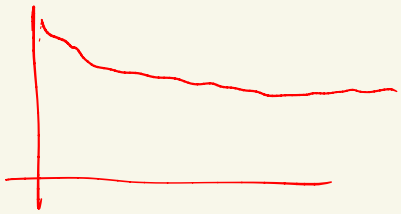
$$\frac{\partial J}{\partial w} = \frac{1}{N} X' (y - f(xw)) \frac{\partial f}{\partial w}$$ (Gradiente de J en función de w)

Descenso por el Gradiente.



Learning Rate.

$$w^t = w^{t-1} - \eta \nabla_w J$$



$$w^t = w^{t-1} - \eta \nabla_w J = w^{t-1} - \eta\, x^T (y - f(xw)) f'(xw)$$

⇒ Actualizar para todos los datos de una sola vez.

Descenso de gradiente batch.

$$w^t = w^{t-1} - \eta\, x^T (y - f(xw)) f'(xw)$$

una iteración de esto se llama época (epoch)

⇒ Actualizar para un dato:

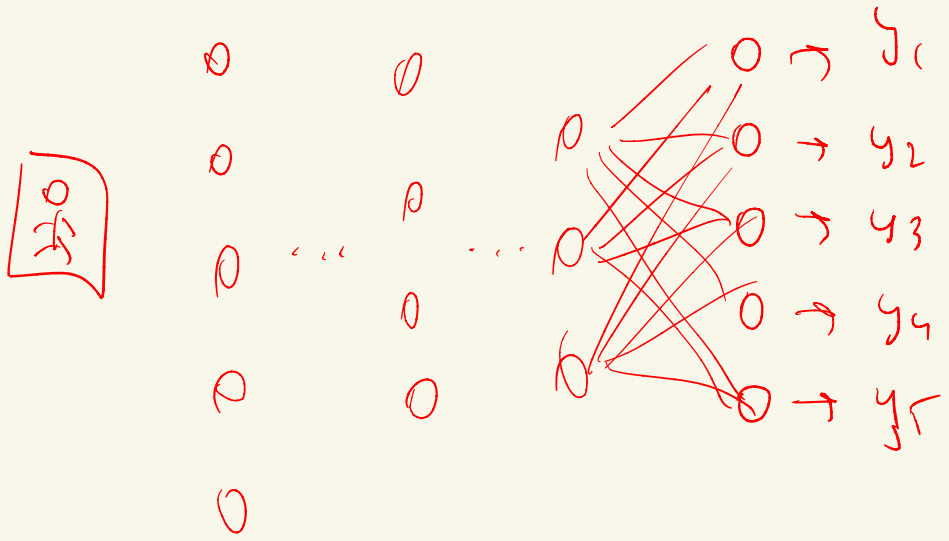
$$w^t = w^{t-1} - \eta \underbrace{x_i^T (y_i - f(x_i w)) f'(x_i w)}_{\nabla_w J_i}$$

for $t = 1 \dots T$ — nº de épocas

for $i = 1 \dots N$

$$w^t = w^{t-1} - \eta \nabla_w T_i$$

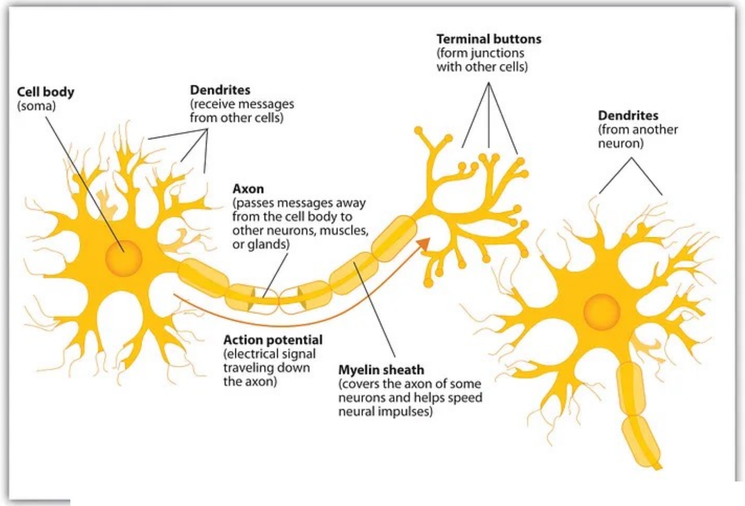
batch size



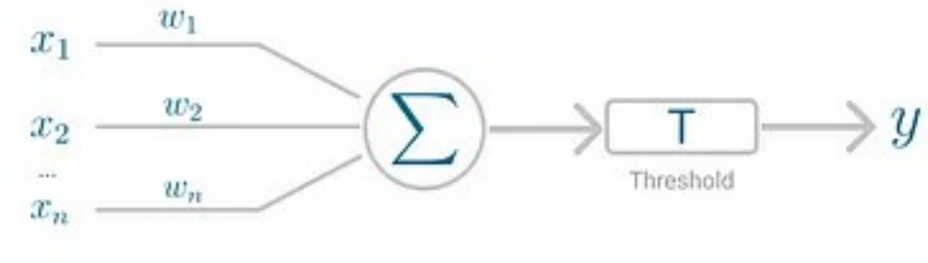
$$\frac{e^{z}}{\sum_{j} e^{z_j}} = p(c_j)$$

$$\sum_{j} y_j = 1$$

El perceptrón (1940)



$$\underbrace{f(x,w)}_{\text{output}} = \underbrace{x_1}_{\text{inputs}} w_1 + \cdots + x_n \underbrace{w_n}_{\text{weights}}$$

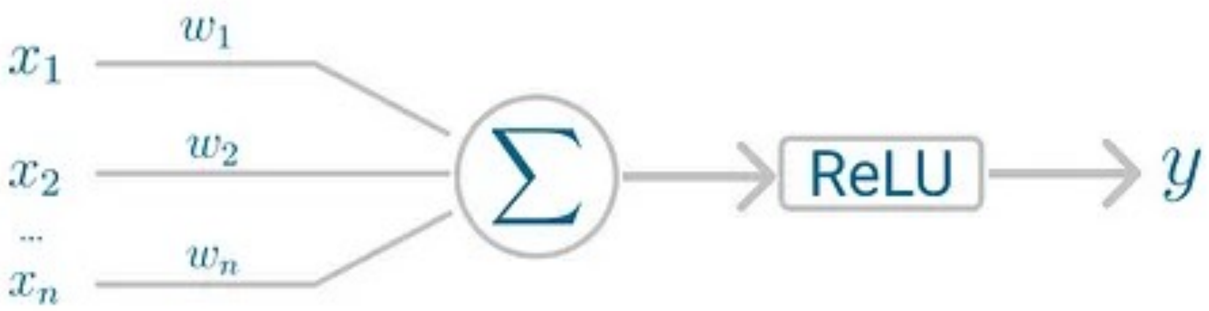


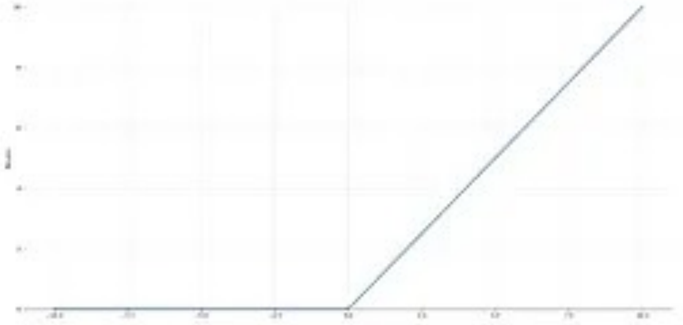
$$y = \begin{cases} 1, \text{if } \overbrace{\sum_i w_i x_i}^{\text{weighted sum}} - \underbrace{T}_{\text{threshold}} > 0 \\ 0, \text{otherwise} \end{cases}$$

$$\underbrace{D(w,c)}_{\text{distance}} = -\sum_{i \in \mathrm{M}} \overbrace{y_i}^{\text{output}} \left(x_i w_i + c\right)$$

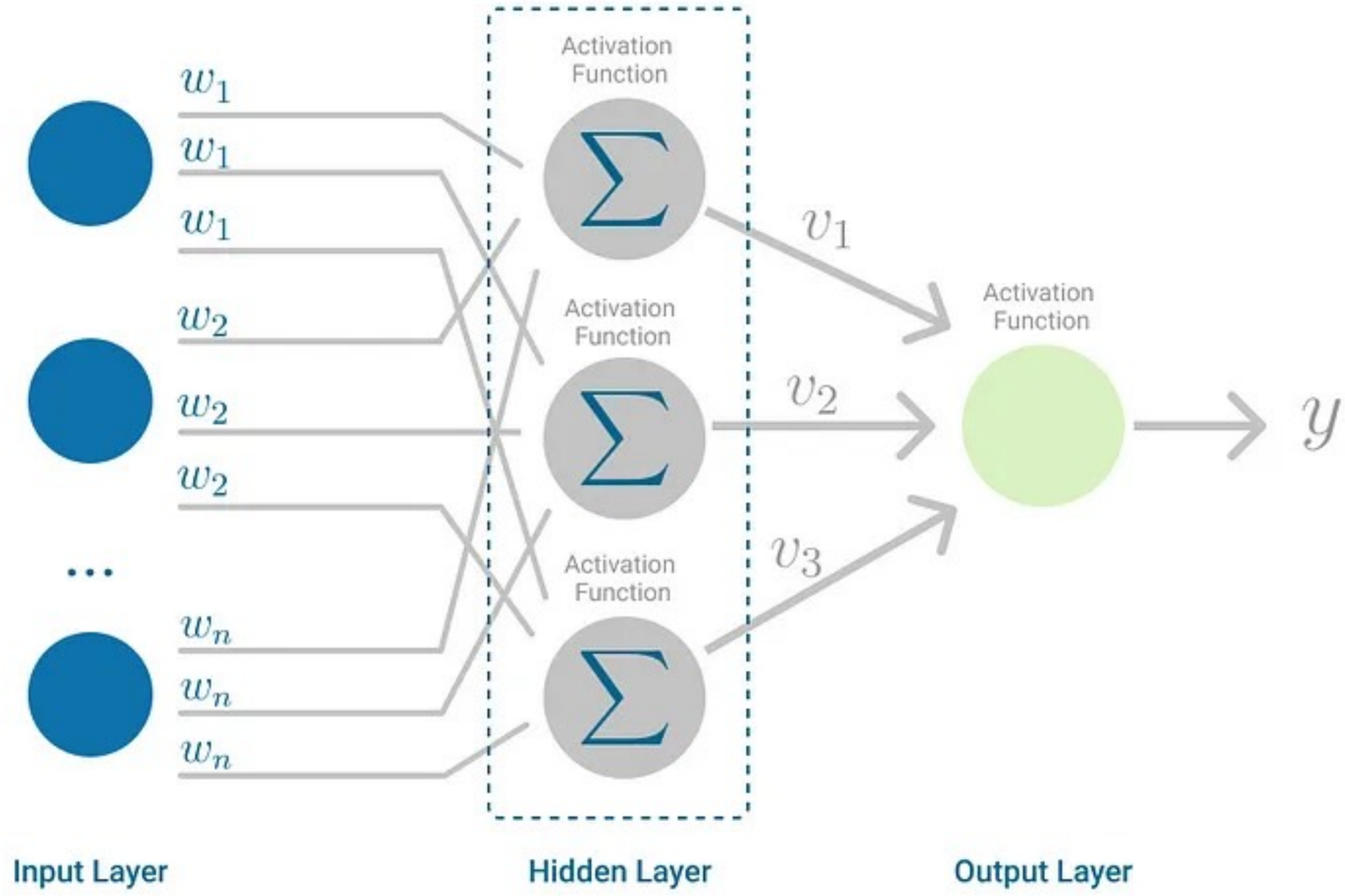
misclassified observations

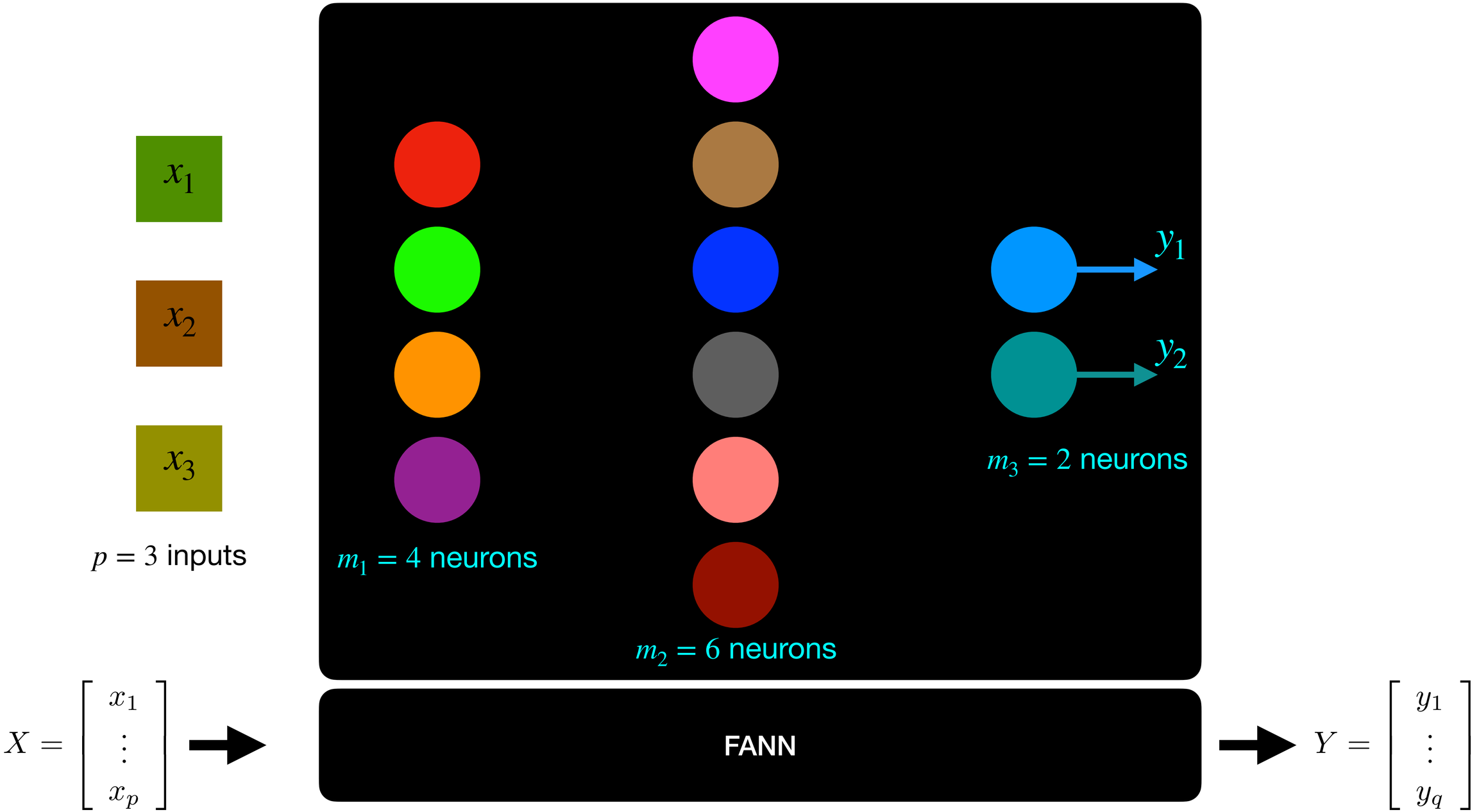
El perceptrón (1940)

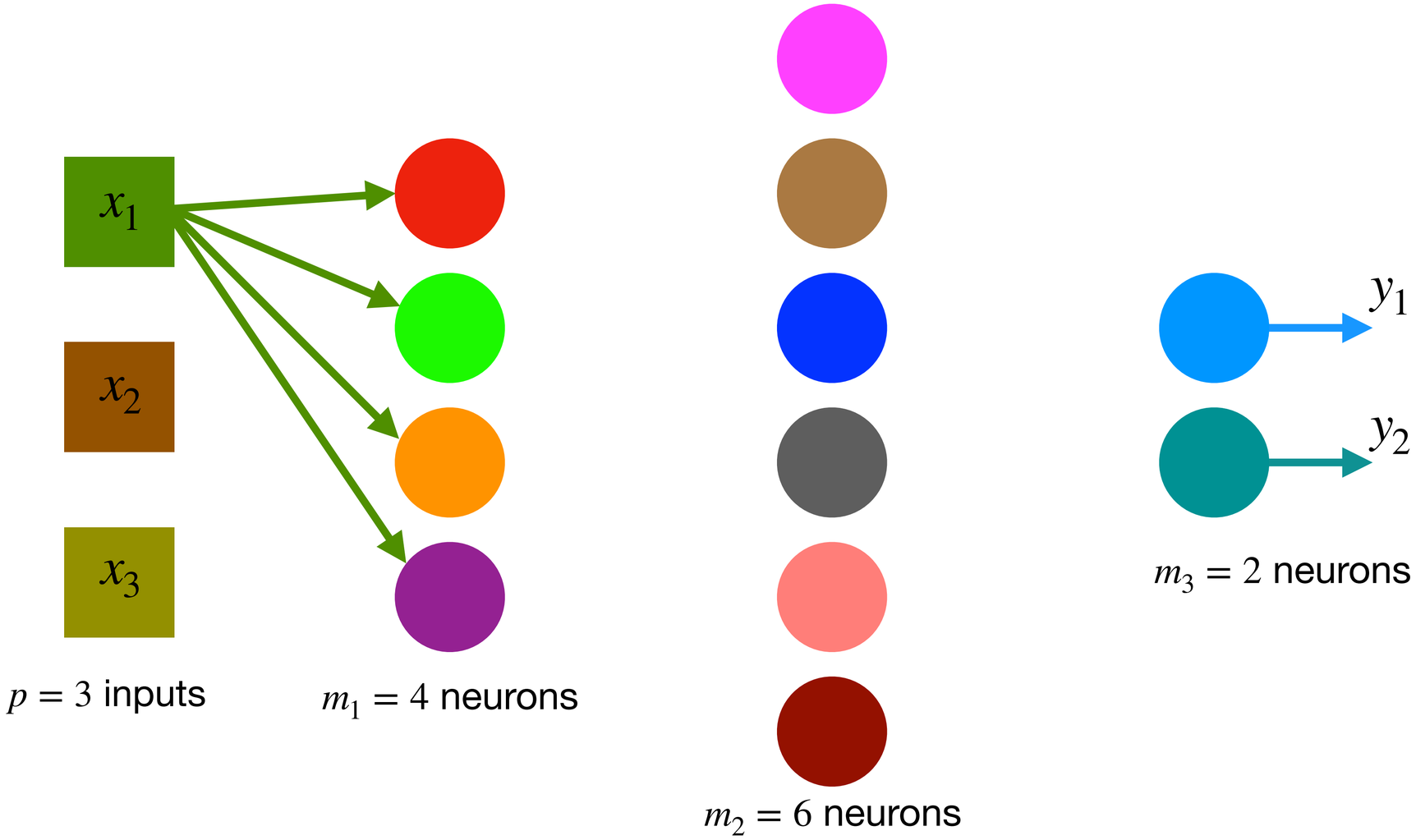


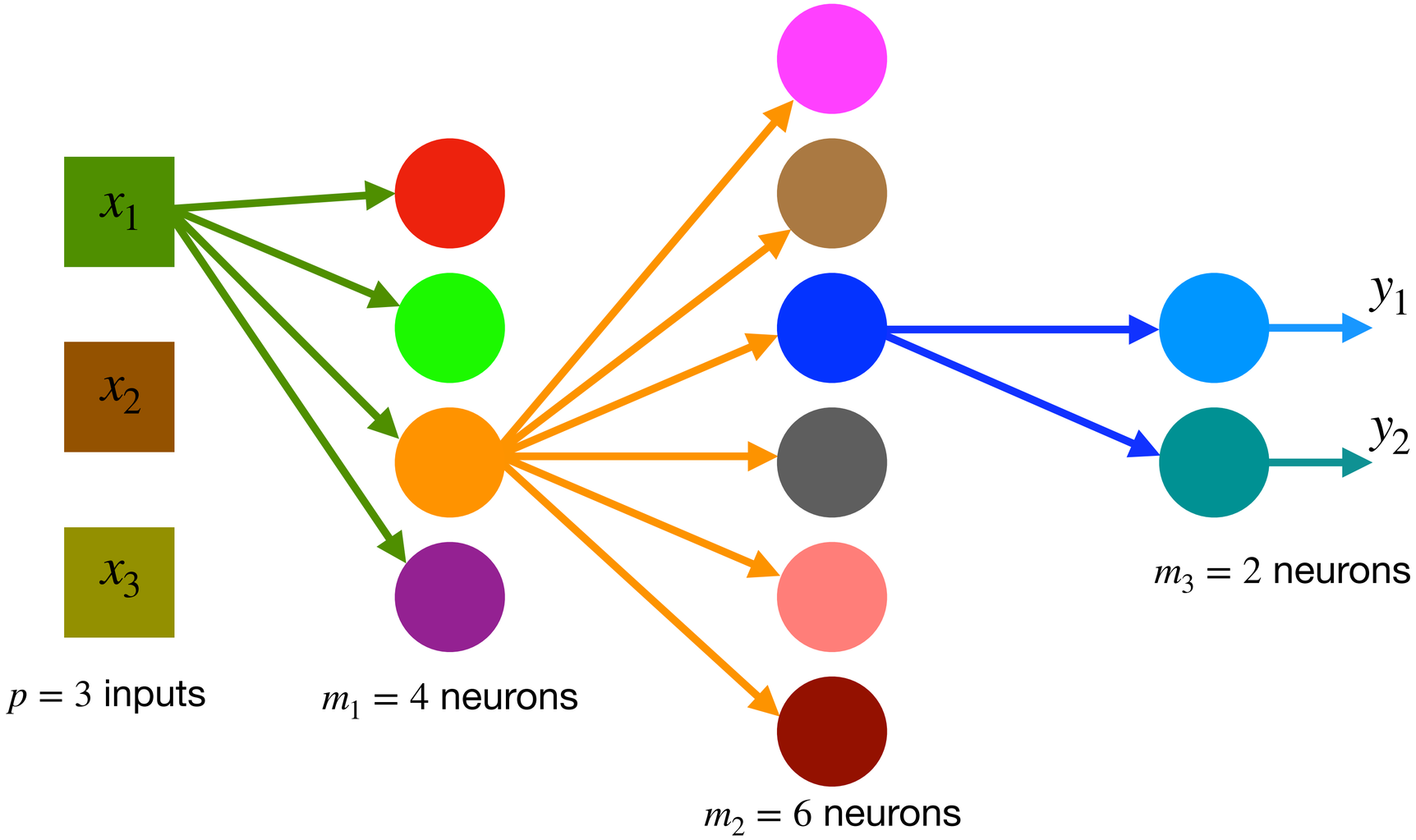


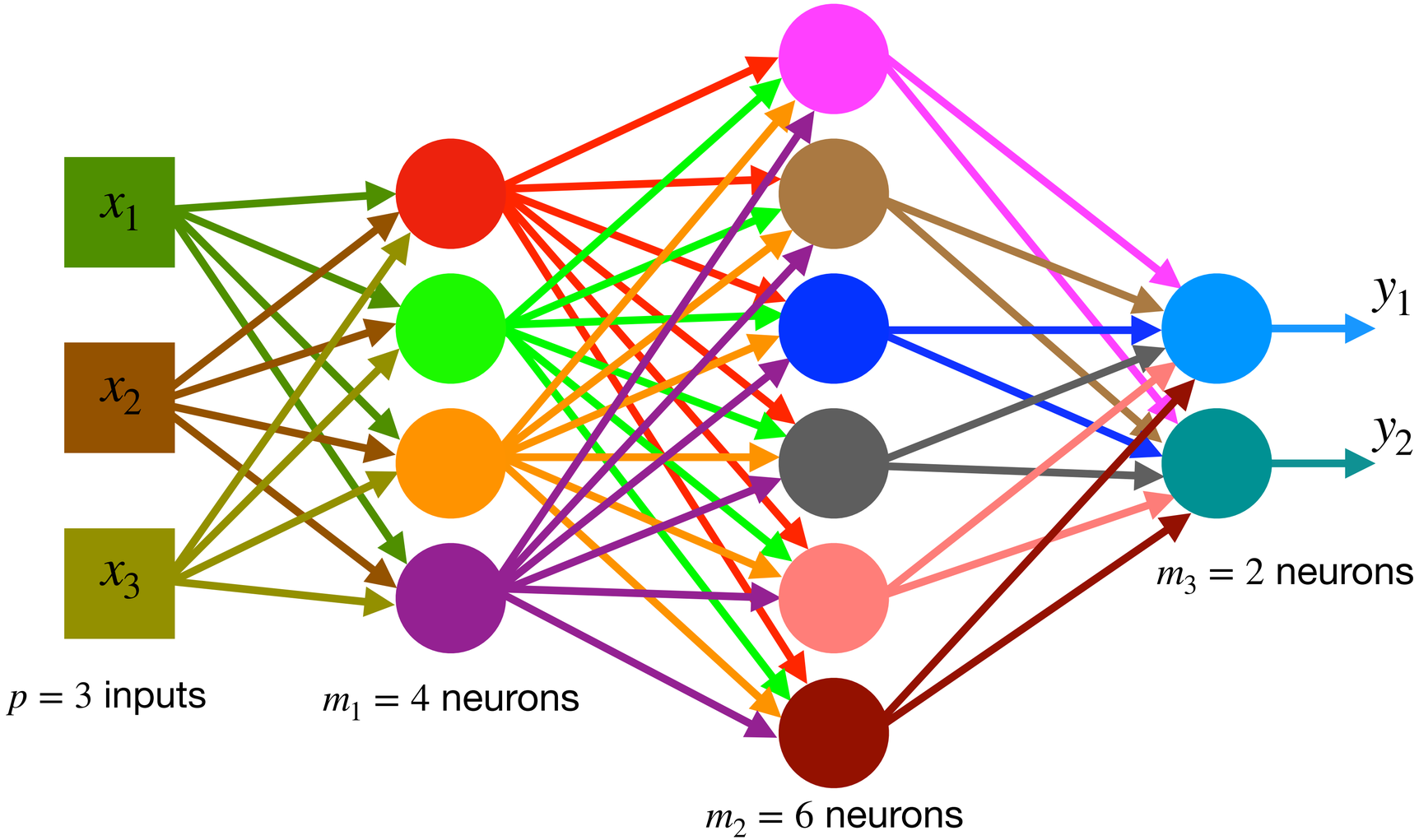
EL perceptrón multicapa

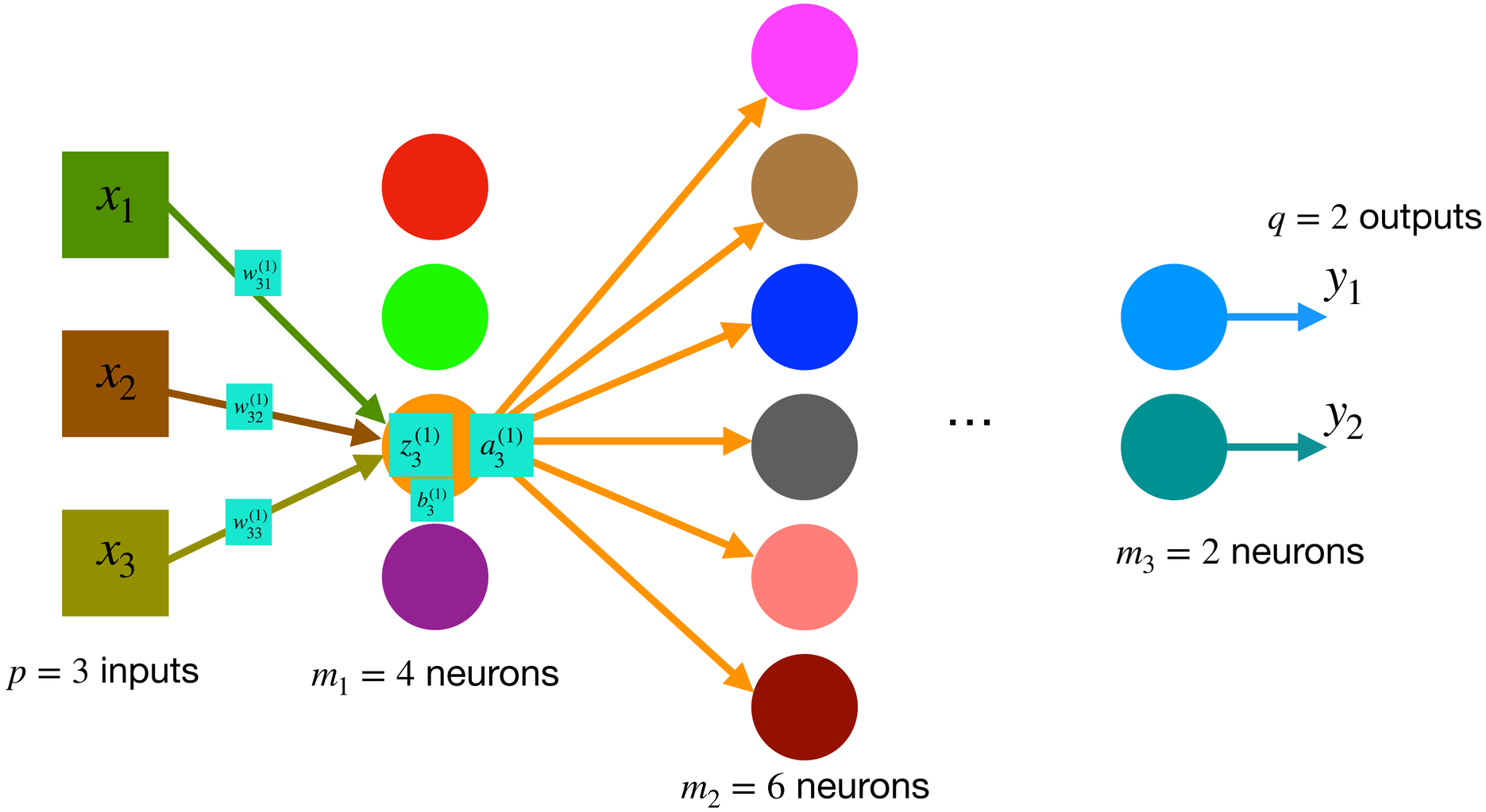


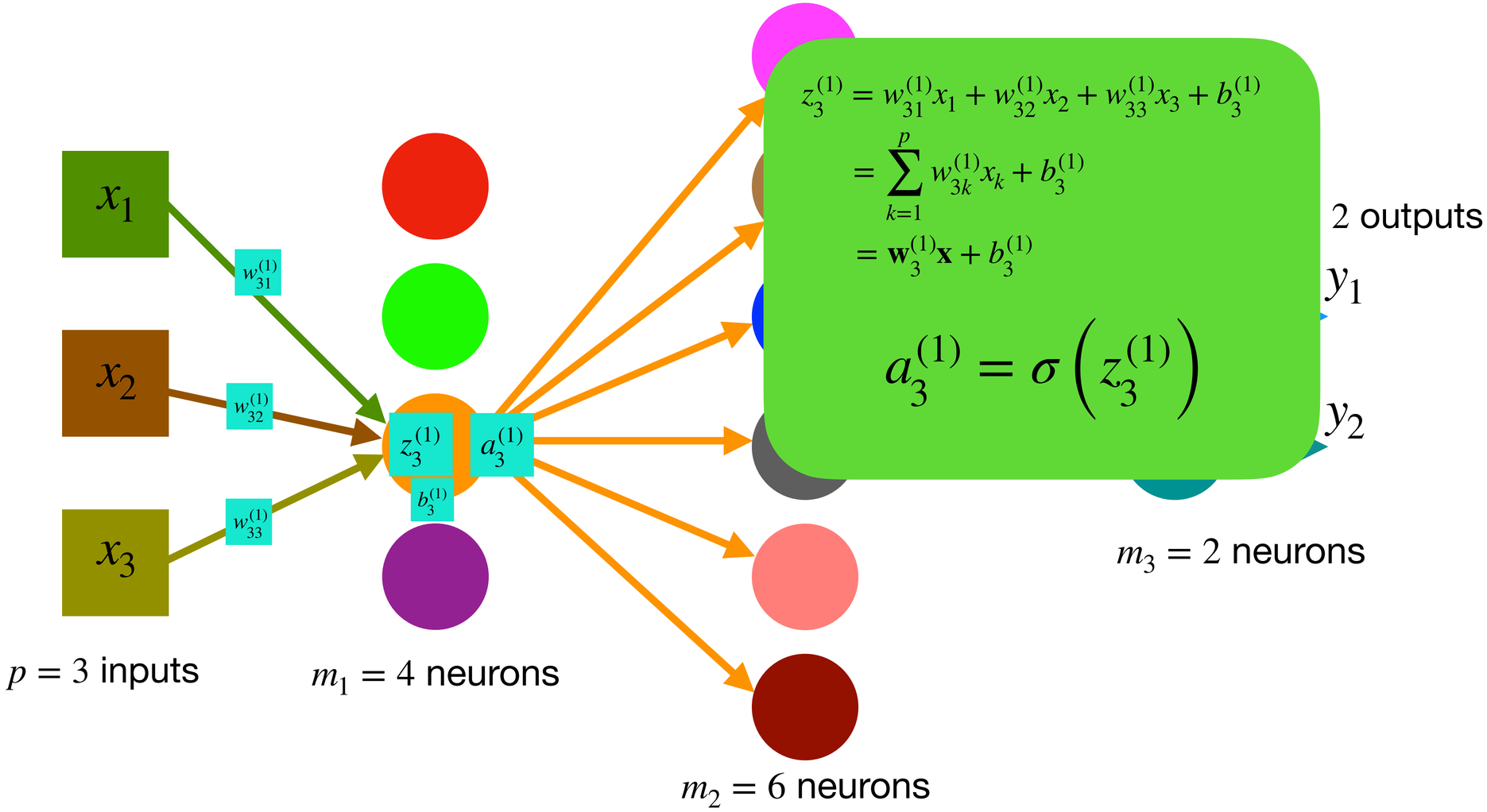

x_1
x_2
x_3
$p = 3$ inputs
$m_1 = 4$ neurons
$m_2 = 6$ neurons
$m_3 = 2$ neurons
y_1
y_2
$X = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$
FANN
$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_q \end{bmatrix}$

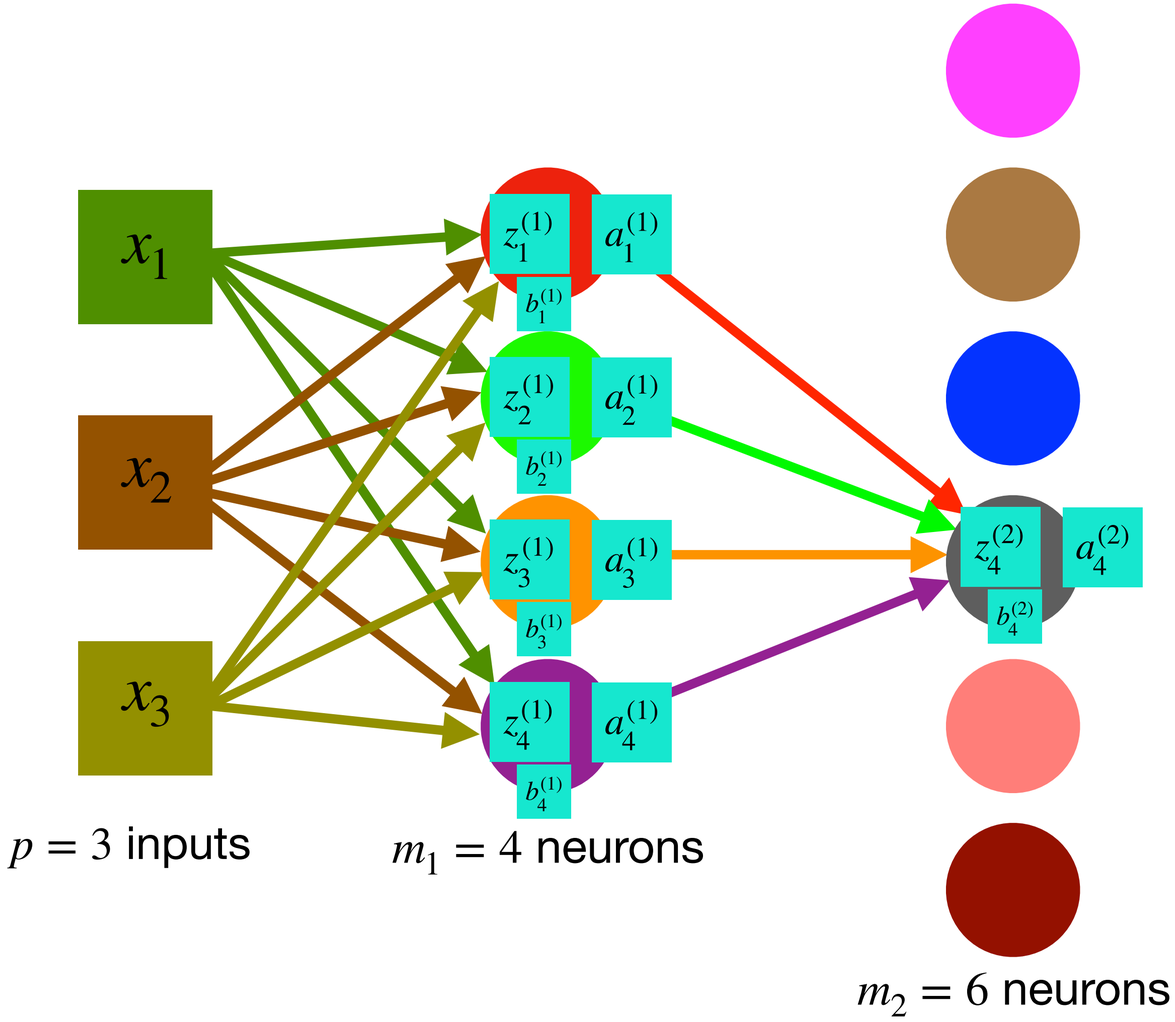

x_1
x_2
x_3
y_1
y_2
$p = 3$ inputs
$m_1 = 4$ neurons
$m_2 = 6$ neurons
$m_3 = 2$ neurons


x_1
x_2
x_3
y_1
y_2
$p = 3$ inputs
$m_1 = 4$ neurons
$m_2 = 6$ neurons
$m_3 = 2$ neurons


x_1
x_2
x_3
y_1
y_2
$p = 3$ inputs
$m_1 = 4$ neurons
$m_2 = 6$ neurons
$m_3 = 2$ neurons


x_1
x_2
x_3
$w_{31}^{(1)}$
$w_{32}^{(1)}$
$w_{33}^{(1)}$
$z_3^{(1)}$
$a_3^{(1)}$
$b_3^{(1)}$
$p = 3$ inputs
$m_1 = 4$ neurons
$m_2 = 6$ neurons
...
$q = 2$ outputs
y_1
y_2
$m_3 = 2$ neurons


x_1
x_2
x_3
$w_{31}^{(1)}$
$w_{32}^{(1)}$
$w_{33}^{(1)}$
$z_3^{(1)}$
$a_3^{(1)}$
$b_3^{(1)}$
$z_3^{(1)} = w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3^{(1)}$
$= \sum_{k=1}^{p} w_{3k}^{(1)}x_k + b_3^{(1)}$
$= \mathbf{w}_3^{(1)}\mathbf{x} + b_3^{(1)}$
$a_3^{(1)} = \sigma\left(z_3^{(1)}\right)$
2 outputs
y_1
y_2
$p = 3$ inputs
$m_1 = 4$ neurons
$m_2 = 6$ neurons
$m_3 = 2$ neurons

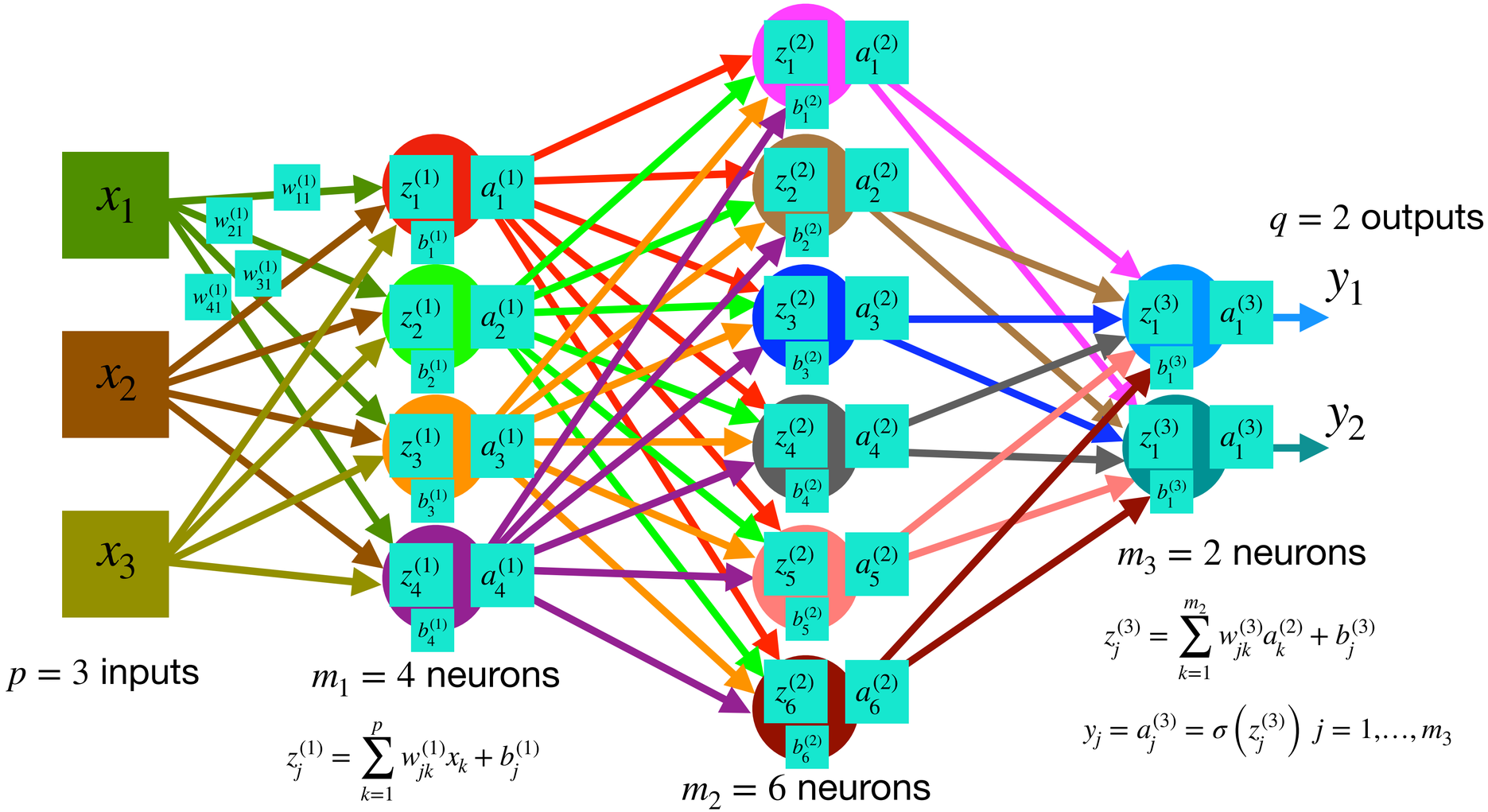


$$z_4^{(2)} = w_{41}^{(2)} a_1^{(1)} + w_{42}^{(2)} a_2^{(1)} + w_{43}^{(2)} a_3^{(1)} + w_{44}^{(2)} a_4^{(1)} + b_4^{(2)}$$

$$= \sum_{k=1}^{m_1} w_{4k}^{(2)} a_k^{(1)} + b_4^{(2)}$$

$$= \mathbf{w}_4^{(2)} \mathbf{x} + b_4^{(2)}$$

$$a_4^{(2)} = \sigma\left(z_4^{(2)}\right)$$



$$z_j^{(1)} = \sum_{k=1}^{p} w_{jk}^{(1)} x_k + b_j^{(1)}$$

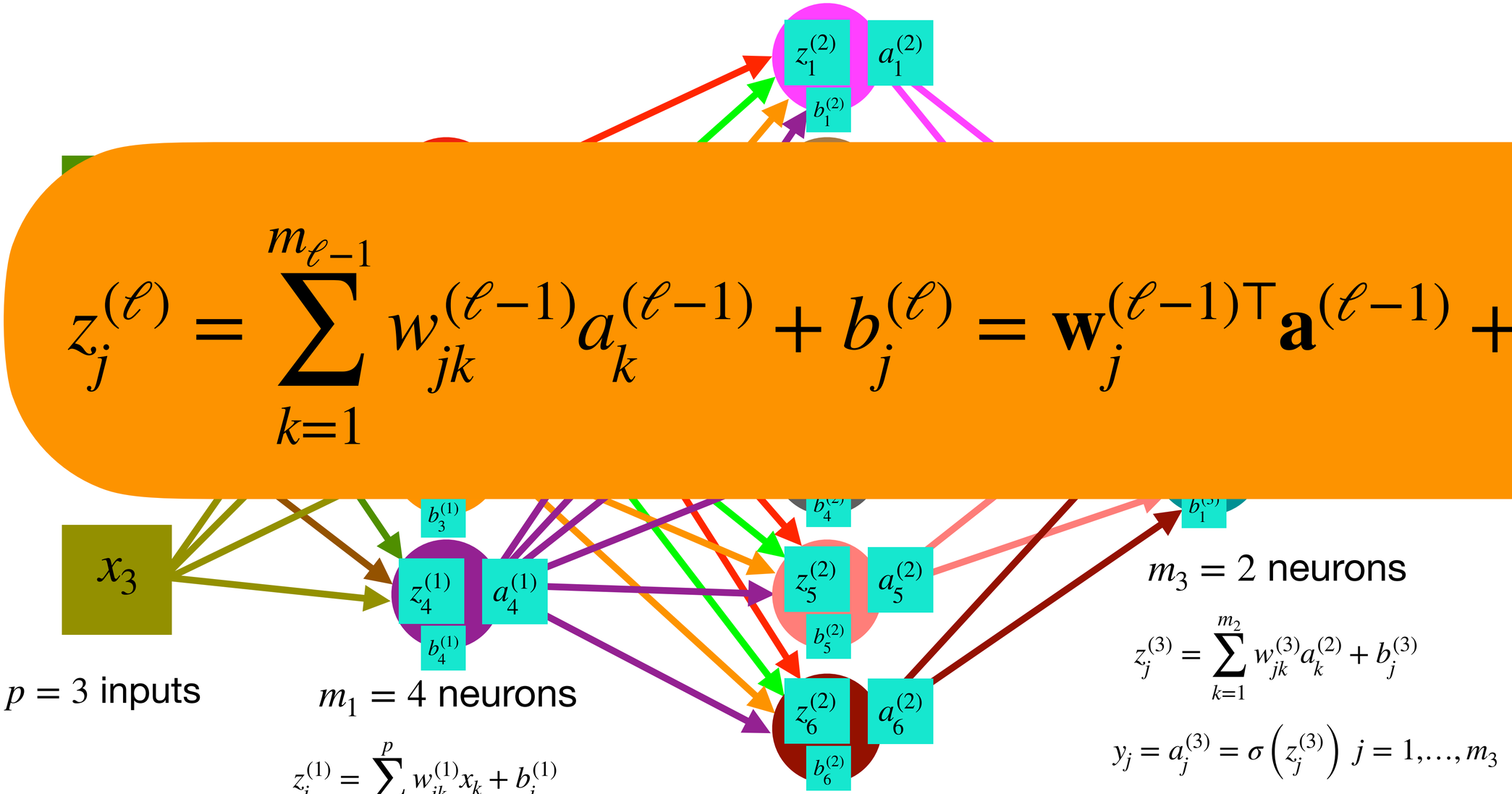
$$a_j^{(1)} = \sigma\left(z_j^{(1)}\right) \quad j = 1,\ldots,m_1$$

$$z_j^{(2)} = \sum_{k=1}^{m_1} w_{jk}^{(2)} a_k^{(1)} + b_j^{(2)}$$

$$a_j^{(2)} = \sigma\left(z_j^{(2)}\right) \quad j = 1,\ldots,m_2$$

$$z_j^{(3)} = \sum_{k=1}^{m_2} w_{jk}^{(3)} a_k^{(2)} + b_j^{(3)}$$

$$y_j = a_j^{(3)} = \sigma\left(z_j^{(3)}\right) \quad j = 1,\ldots,m_3$$



$$z_j^{(\ell)} = \sum_{k=1}^{m_{\ell-1}} w_{jk}^{(\ell-1)} a_k^{(\ell-1)} + b_j^{(\ell)} = \mathbf{w}_j^{(\ell-1)\top} \mathbf{a}^{(\ell-1)} + b_4^{(2)}$$

$p = 3$ inputs

$m_1 = 4$ neurons

$$z_j^{(1)} = \sum_{k=1}^{p} w_{jk}^{(1)} x_k + b_j^{(1)}$$

$$a_j^{(1)} = \sigma\left(z_j^{(1)}\right) \; j = 1, \ldots, m_1$$

$m_2 = 6$ neurons

$$z_j^{(2)} = \sum_{k=1}^{m_1} w_{jk}^{(2)} a_k^{(1)} + b_j^{(2)}$$

$$a_j^{(2)} = \sigma\left(z_j^{(2)}\right) \; j = 1, \ldots, m_2$$

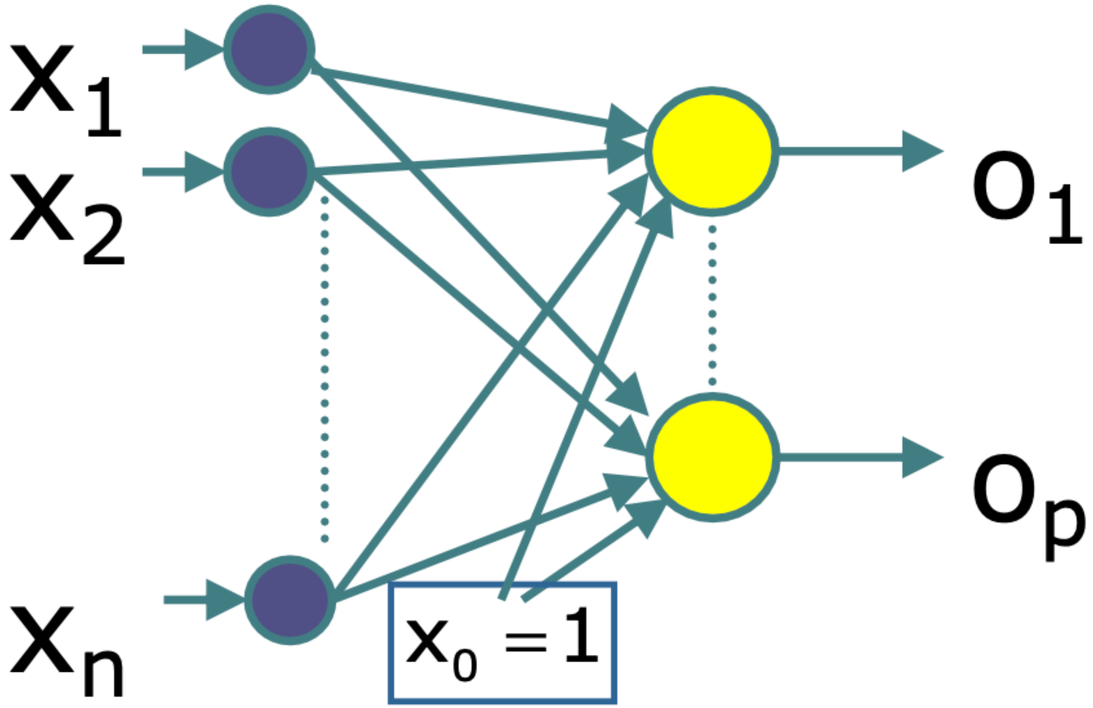
$m_3 = 2$ neurons

$$z_j^{(3)} = \sum_{k=1}^{m_2} w_{jk}^{(3)} a_k^{(2)} + b_j^{(3)}$$

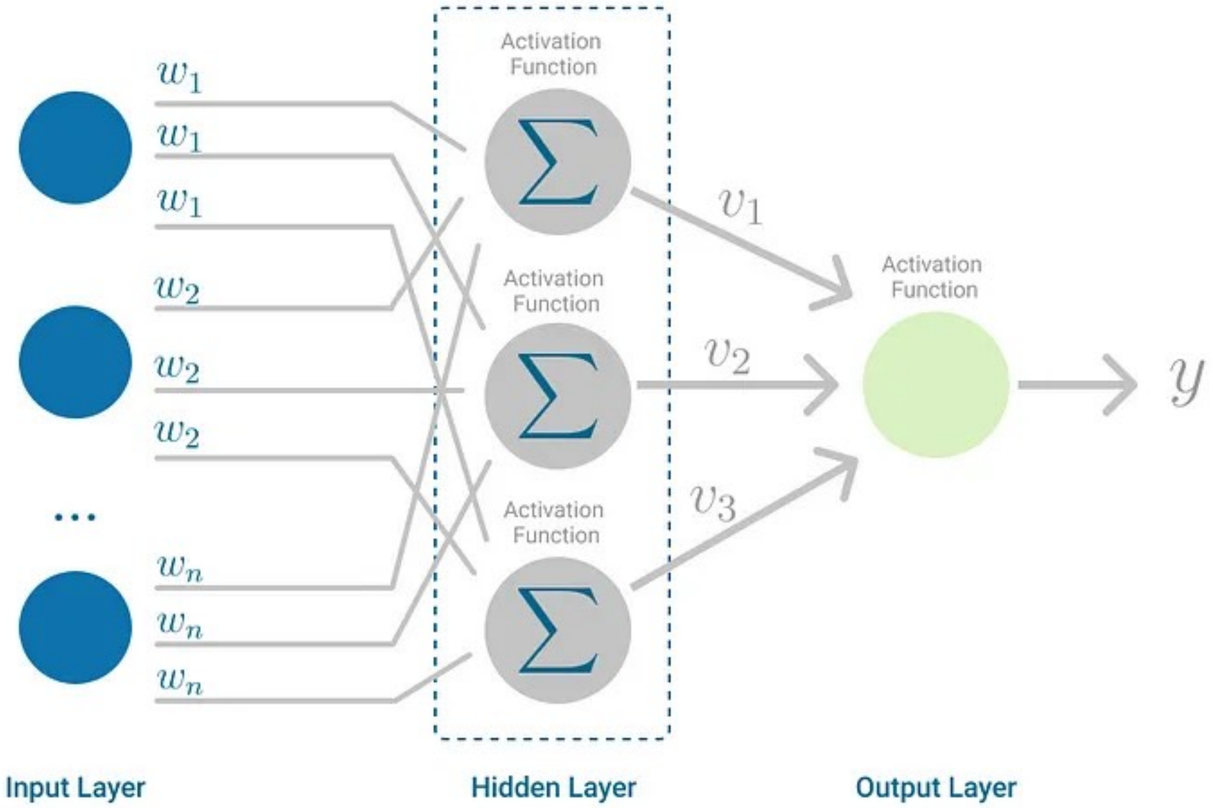
$$y_j = a_j^{(3)} = \sigma\left(z_j^{(3)}\right) \; j = 1, \ldots, m_3$$

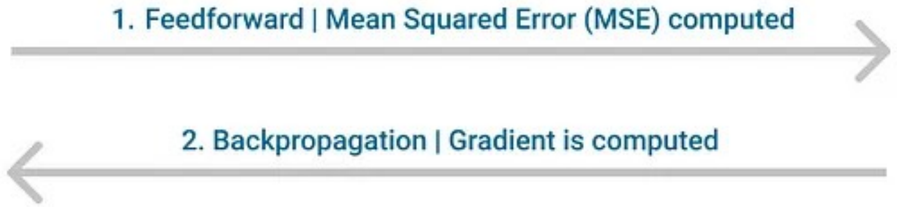
Entrenamiento

$$o_i = f\left(\sum_{k=0}^{n} w_{ik}x_k\right) \quad i = 1,\ldots,p$$



Retroprogación (backprograpation)





Algoritmo 1 Forward Propagation

Requerir: Profundidad de la red, l
Requerir: $\boldsymbol{W}^{(i)}, i \in \{1, \ldots, l\}$, pesos de la red
Requerir: $\boldsymbol{b}^{(i)}, i \in \{1, \ldots, l\}$, parámetros bias de la red
Requerir: $\boldsymbol{x}$, el input
Requerir: $\boldsymbol{y}$, el output target

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \ldots, l$:
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

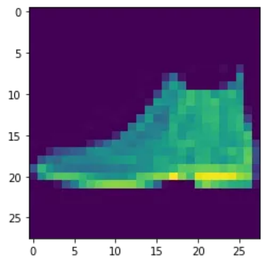
Algoritmo 2 Back-Propagation

Luego de completar forward propagation:

$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
for $k = l, l-1, \ldots, 1$:
$\nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$
$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g}$
$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g} \boldsymbol{h}^{(k-1)T}$
$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)T} \boldsymbol{g}$

Keras

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
tf.__version__ # '2.9.1'
keras.__version__ # '2.9.0'

fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
print(X_train_full.shape)
print(X_train_full.dtype)

(60000, 28, 28)
uint8

X_valid, X_train = X_train_full[:5000]/255.0, X_train_full[5000:]/255.0
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress',
'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
class_names[y_train[0]]

img = X_test[0]
plt.imshow(img)
```




```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28,28]),
    keras.layers.Dense(300, activation='relu'),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 300)               235500

 dense_1 (Dense)             (None, 100)               30100

 dense_2 (Dense)             (None, 10)                1010

=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

Keras

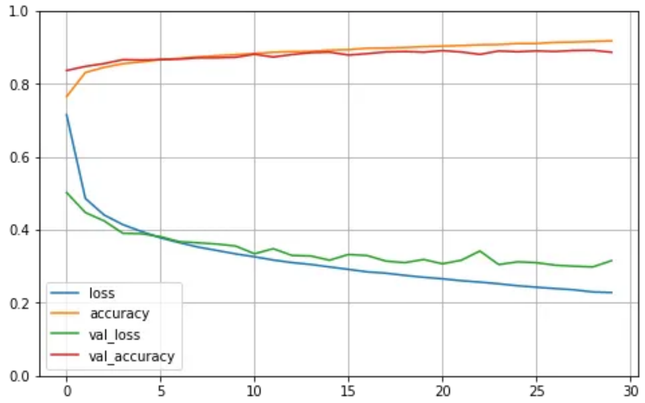
```
model.layers
[<keras.layers.reshaping.flatten.Flatten at 0x7fc17f658910>,
 <keras.layers.core.dense.Dense at 0x7fc17f4a9760>,
 <keras.layers.core.dense.Dense at 0x7fc17f579b20>,
 <keras.layers.core.dense.Dense at 0x7fc17f579bb0>]

hidden1 = model.layers[1]
hidden1.name
layer1 = model.get_layer(hidden1.name)
weights, biases = hidden1.get_weights()

weights
array([[ 0.05980267, -0.00851811, -0.03223576, ...,  0.0417341 ,
        -0.05151674,  0.02141821],
       [ 0.02560429,  0.02692608,  0.01197942, ...,  0.00847472,
        -0.01258808,  0.00496984],
       [ 0.0110984 ,  0.03018346, -0.06010713, ..., -0.06778863,
         0.01074214, -0.06266507],
       ...,
       [-0.04093829, -0.05239757,  0.05745775, ...,  0.04037939,
        -0.04425038,  0.0710177 ],
       [ 0.07078798,  0.02598803, -0.025334  , ..., -0.0633328 ,
        -0.04895741, -0.05786116],
       [-0.03853485, -0.00021387, -0.03003352, ...,  0.06768821,
         0.01444261,  0.05353366]], dtype=float32)
```

```
biases.shape
(300,)

biases
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                  ...
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer = 'sgd',
              metrics=['accuracy'])
```

Keras

```
history = model.fit(X_train, y_train, epochs=30, validation_data=(X_valid, y_valid))

Epoch 1/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.7158 - accuracy: 0.7652 - val_loss: 0.5024 - val_accuracy: 0.8368
Epoch 2/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.4862 - accuracy: 0.8314 - val_loss: 0.4475 - val_accuracy: 0.8480
Epoch 3/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.4408 - accuracy: 0.8455 - val_loss: 0.4246 - val_accuracy: 0.8558
Epoch 4/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.4142 - accuracy: 0.8555 - val_loss: 0.3906 - val_accuracy: 0.8666

...

Epoch 29/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.2299 - accuracy: 0.9164 - val_loss: 0.2983 - val_accuracy: 0.8922
Epoch 30/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.2281 - accuracy: 0.9180 - val_loss: 0.3155 - val_accuracy: 0.8868
```



Keras

```
X_new = X_test[:5]
y_preds = model.predict(X_new)
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```