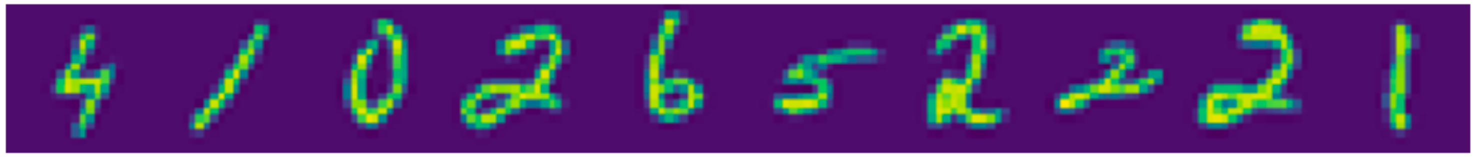
Práctica 3:
regresión logística multiclase y redes neuronales

```python
from scipy.io import loadmat

data = loadmat('ex3data1.mat')
# se pueden consultar las claves con data.keys()
y = data['y']
X = data['X']
# almacena los datos leídos en X, y
```

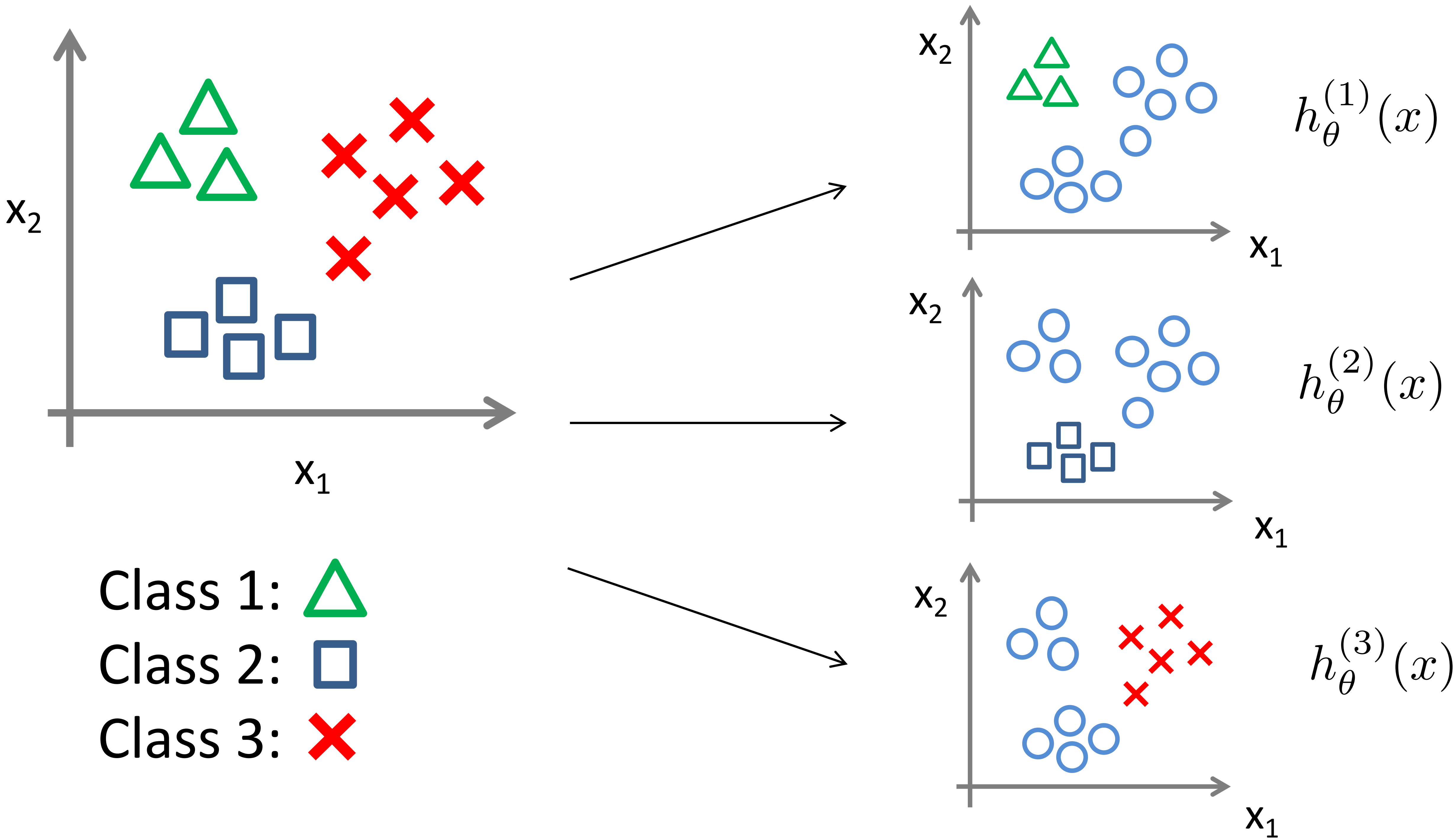
```python
y = np.ravel(y)  #para convertir la y en un array 1-D
```

```python
# Selecciona aleatoriamente 10 ejemplos y los pinta
sample = np.random.choice(X.shape[0], 10)
plt.imshow(X[sample, :].reshape(-1, 20).T)
plt.axis('off')
```



Regresión logística

One-vs-all (one-vs-rest):



$$h_\theta^{(i)}(x) = P(y = i | x; \theta) \qquad (i = 1, 2, 3)$$

```python
def oneVsAll(X, y, n_labels, reg):
    """
    oneVsAll entrena varios clasificadores por regresión logística con término
    de regularización 'reg' y devuelve el resultado en una matriz, donde
    la fila i-ésima corresponde al clasificador de la etiqueta i-ésima
    """
```

Recuerda que el argumento y es un vector con etiquetas de 1 a 10, donde el dígito "0" se ha hecho corresponder con la etiqueta 10. Por otra parte, cuando entrenes al clasificador para la clase $k \in \{1, \ldots, K\}$, tendrás que obtener un vector m-dimensional de etiquetas y donde $y_j \in 0, 1$ indica si el ejemplo de entrenamiento j-ésimo pertenece a la clase k $(y_j = 1)$ o a otra clase $(y_j = 0)$. Para ello, te será útil saber que en Python `True * 1` es igual a 1 y `False * 1` es igual a 0.

```python
y = np.array([10, 10, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9])
```

```python
y == 10
```

```
array([ True,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False])
```

```python
(y == 10) * 1
```

```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

one_hot y

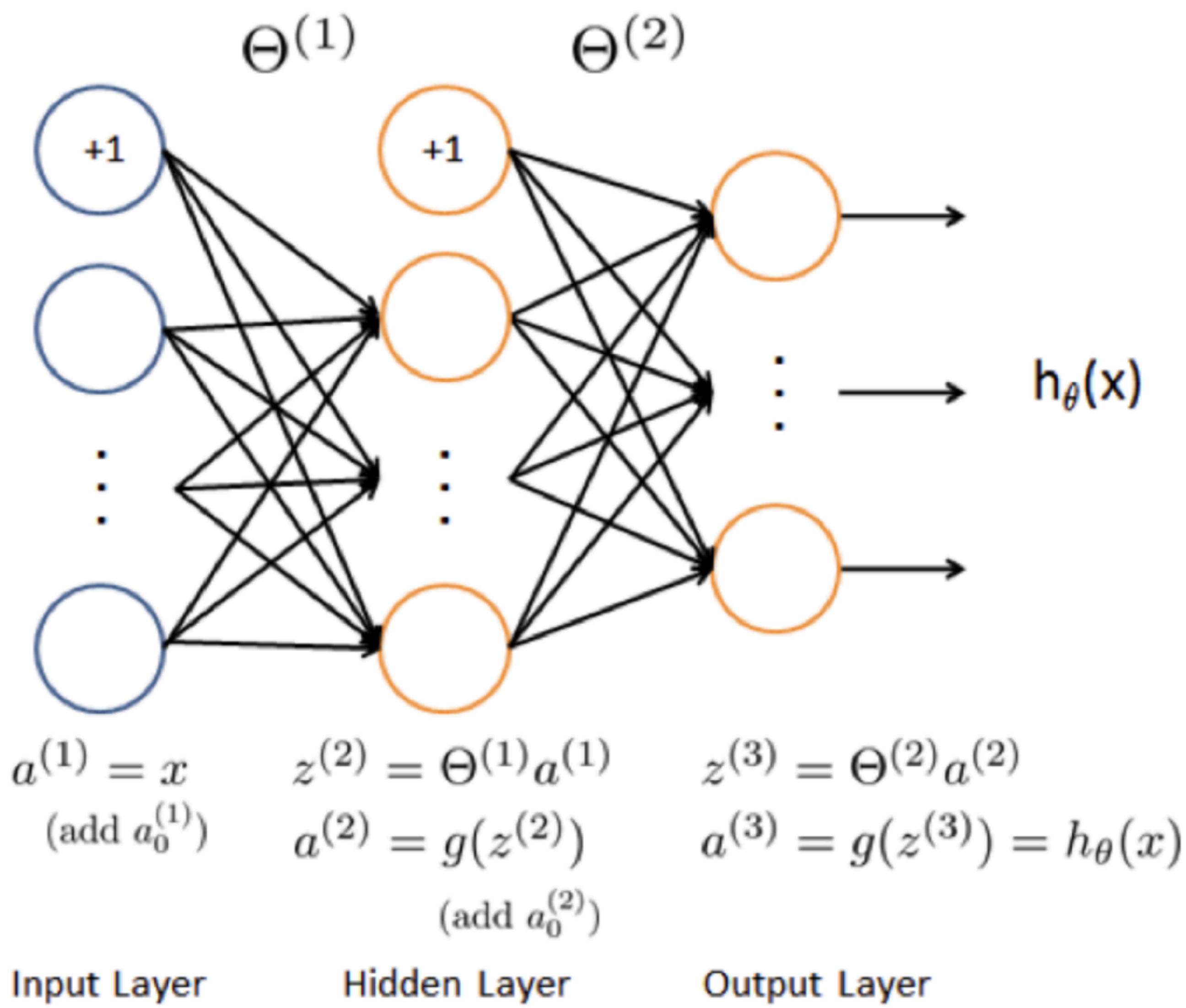
```
m = len(y)
num_labels = 10
```

$$\begin{array}{ccccccc} & 1 & & 2 & & 10 & y \\ y = & \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} & , & \begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix} & , \ldots \; o & \begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix} & \\ & 0 & & 1 & & 9 & y - 1 \end{array}$$

```
y = (y - 1)
y_onehot = np.zeros((m, num_labels))  # 5000 x 10

for i in range(m):
    y_onehot[i][y[i]] = 1
```

Redes neuronales



La red neuronal está entrenada de forma que la primera neurona de salida se activa cuando reconoce un 1, la segunda cuando reconoce un 2, y así sucesivamente hasta la décima que se activa cuando reconoce un 0

```
weights = loadmat('ex3weights.mat')
theta1, theta2 = weights['Theta1'], weights['Theta2']
# Theta1 es de dimensión 25 x 401
# Theta2 es de dimensión 10 x 26
```

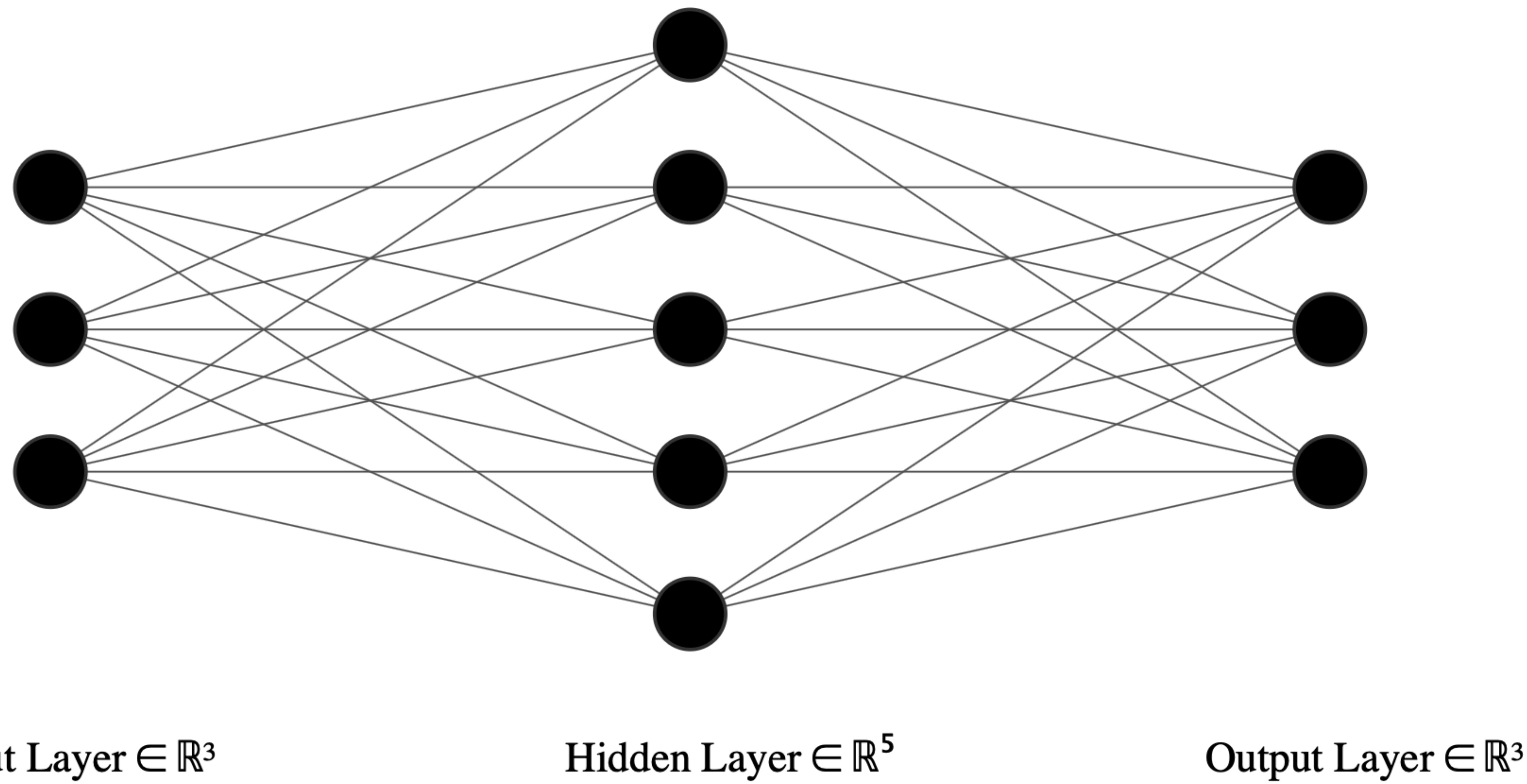
```
input_layer_size = 3
hidden_layer_size = 5
num_labels = 3
m = 5
```

$X : 5 \times 4$

$y : 5 \times 3$

$\Theta^{(1)} : 5 \times 4$

$\Theta^{(2)} : 3 \times 6$



http://alexlenail.me/NN-SVG/

```
input_layer_size = 3
hidden_layer_size = 5
num_labels = 3
m = 5
```

$X : 5 \times 4$

$y : 5 \times 3$

$\Theta^{(1)} : 5 \times 4$

$\Theta^{(2)} : 3 \times 6$

$$X = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} \\ 1 & x_1^{(4)} & x_2^{(4)} & x_3^{(4)} \\ 1 & x_1^{(5)} & x_2^{(5)} & x_3^{(5)} \end{pmatrix} \quad y = \begin{pmatrix} y_1^{(1)} & y_2^{(1)} & y_3^{(1)} \\ y_1^{(2)} & y_2^{(2)} & y_3^{(2)} \\ y_1^{(3)} & y_2^{(3)} & y_3^{(3)} \\ y_1^{(4)} & y_2^{(4)} & y_3^{(4)} \\ y_1^{(5)} & y_2^{(5)} & y_3^{(5)} \end{pmatrix}$$

$$\Theta^{(1)} = \begin{pmatrix} \Theta_{1,0}^{(1)} & \Theta_{1,1}^{(1)} & \Theta_{1,2}^{(1)} & \Theta_{1,3}^{(1)} \\ \Theta_{2,0}^{(1)} & \Theta_{2,1}^{(1)} & \Theta_{2,2}^{(1)} & \Theta_{2,3}^{(1)} \\ \Theta_{3,0}^{(1)} & \Theta_{3,1}^{(1)} & \Theta_{3,2}^{(1)} & \Theta_{3,3}^{(1)} \\ \Theta_{4,0}^{(1)} & \Theta_{4,1}^{(1)} & \Theta_{4,2}^{(1)} & \Theta_{4,3}^{(1)} \\ \Theta_{5,0}^{(1)} & \Theta_{5,1}^{(1)} & \Theta_{5,2}^{(1)} & \Theta_{5,3}^{(1)} \end{pmatrix} \Theta^{(2)} = \begin{pmatrix} \Theta_{1,0}^{(2)} & \Theta_{1,1}^{(2)} & \Theta_{1,2}^{(2)} & \Theta_{1,3}^{(2)} & \Theta_{1,4}^{(2)} & \Theta_{1,5}^{(2)} \\ \Theta_{2,0}^{(2)} & \Theta_{2,1}^{(2)} & \Theta_{2,2}^{(2)} & \Theta_{2,3}^{(2)} & \Theta_{2,4}^{(2)} & \Theta_{2,5}^{(2)} \\ \Theta_{3,0}^{(2)} & \Theta_{3,1}^{(2)} & \Theta_{3,2}^{(2)} & \Theta_{3,3}^{(2)} & \Theta_{3,4}^{(2)} & \Theta_{3,5}^{(2)} \end{pmatrix}$$

```
input_layer_size = 3
hidden_layer_size = 5
num_labels = 3
m = 5
```

$X : 5 \times 4$ $\quad \Theta^{(1)} : 5 \times 4$

$y : 5 \times 3$ $\quad \Theta^{(2)} : 3 \times 6$

forward:

$$a^{(1)} = x^{(1)}$$

$$z^{(2)} = \Theta^{(1)} . a^{(1)}$$

```
z2 = np.matmul(X, theta_1.T)
```

$$a^{(2)} = g(z^{(2)})$$

$$\text{add } a_0^{(2)} \rightarrow a^{(2)}$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$a^{(3)} = g(z^{(3)})$$