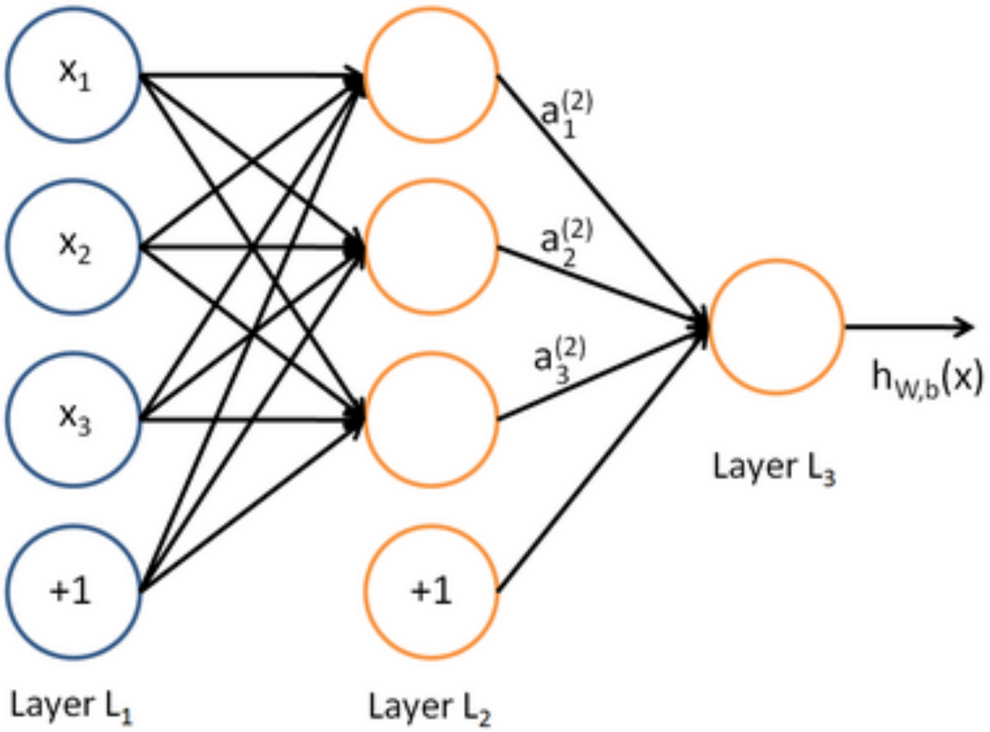
Neural network



$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

Forward propagation

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

$$z^{(2)} = W^{(1)}x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)})$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$
$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

```
W1 = theta[0:hidden_size * visible_size].reshape(hidden_size, visible_size)
W2 = theta[hidden_size * visible_size:2 * hidden_size * visible_size].reshape(visible_size, hidden_size)
b1 = theta[2 * hidden_size * visible_size:2 * hidden_size * visible_size + hidden_size]
b2 = theta[2 * hidden_size * visible_size + hidden_size:]
```

```
z2 = W1.dot(data) + np.tile(b1, (m, 1)).transpose()
a2 = sigmoid(z2)
z3 = W2.dot(a2) + np.tile(b2, (m, 1)).transpose()
h = sigmoid(z3)
```

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

Cost function

$$J(W,b;x,y) = \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2$$

Cost function with regularization

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\| h_{W,b}(x^{(i)}) - y^{(i)} \right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \end{aligned}$$

```
cost = np.sum((h - data) ** 2) / (2 * m) + \
       (lambda_ / 2) * (np.sum(W1 ** 2) + np.sum(W2 ** 2))
```

Cost function with sparsity

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} \left[a_j^{(2)} (x^{(i)}) \right]$$

```
rho_hat = np.sum(a2, axis=1) / m
rho = np.tile(sparsity_param, hidden_size)
```

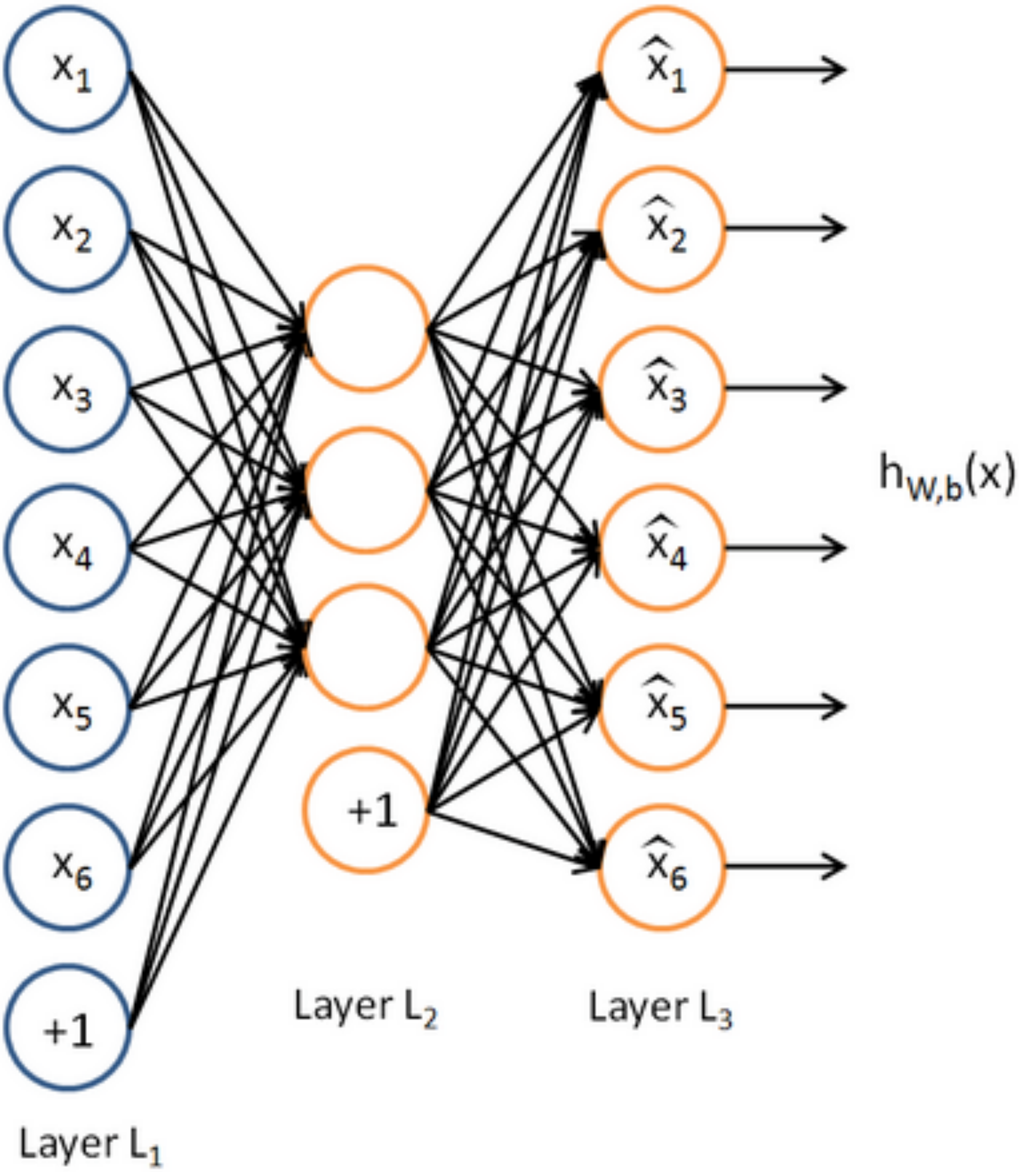
$$\sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}.$$

```
def KL_divergence(x, y):
    return x * np.log(x / y) + (1 - x) * np.log((1 - x) / (1 - y))
```

$$J_{\mathrm{sparse}}(W,b) = J(W,b) + \beta \sum_{j=1}^{s_2} \mathrm{KL}(\rho || \hat{\rho}_j)$$

```
cost = np.sum((h - data) ** 2) / (2 * m) + \
       (lambda_ / 2) * (np.sum(W1 ** 2) + np.sum(W2 ** 2)) + \
       beta * np.sum(KL_divergence(rho, rho_hat))
```

Sparse autoencoder with backpropagation



Output layer(layer3)

For each output unit i in layer n_l (the output layer), set

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \; \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

For the output layer (layer n_l), set

$$\delta^{(n_l)} = -(y - a^{(n_l)}) \bullet f'(z^{(n_l)})$$

```
delta3 = -(data - h) * sigmoid_prime(z3)
```

Hidden layer(layer2)

For $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$

For each node i in layer l, set

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

For $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$

Set

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)} \right) \bullet f'(z^{(l)})$$

```
delta2 = (W2.transpose().dot(delta3) ) * sigmoid_prime(z2)
```

Hidden layer with sparsity

$$\delta_i^{(2)} = \left(\left(\sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)} \right) + \beta \left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i} \right) \right) f'(z_i^{(2)})$$

```
sparsity_delta = np.tile(- rho / rho_hat + (1 - rho) / (1 - rho_hat), (m, 1)).transpose()
```

```
delta2 = (W2.transpose().dot(delta3) + beta * sparsity_delta) * sigmoid_prime(z2)
```

Compute the desired partial derivatives, which are given as:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)}.$$

Compute the desired partial derivatives:

$$\nabla_{W^{(l)}} J(W,b;x,y) = \delta^{(l+1)} (a^{(l)})^T,$$

$$\nabla_{b^{(l)}} J(W,b;x,y) = \delta^{(l+1)}.$$

Iteration step

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)}) \right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(i)},y^{(i)})$$

```
W1grad = delta2.dot(data.transpose()) / m + lambda_ * W1
W2grad = delta3.dot(a2.transpose()) / m + lambda_ * W2
b1grad = np.sum(delta2, axis=1) / m
b2grad = np.sum(delta3, axis=1) / m
```

Update weighting

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)$$

$$b_{i}^{(l)} = b_{i}^{(l)} - \alpha \frac{\partial}{\partial b_{i}^{(l)}} J(W,b)$$

```
J = lambda x: sparse_autoencoder.sparse_autoencoder_cost(x, visible_size, hidden_size,
                                                         lambda_, sparsity_param,
                                                         beta, patches)
options_ = {'maxiter': 400, 'disp': True}
result = scipy.optimize.minimize(J, theta, method='L-BFGS-B', jac=True, options=options_)
```

Set $\Delta W^{(l)} := 0$, $\Delta b^{(l)} := 0$ (matrix/vector of zeros) for all l.

For $i = 1$ to m,

a. Use backpropagation to compute $\nabla_{W^{(l)}} J(W, b; x, y)$ and $\nabla_{b^{(l)}} J(W, b; x, y)$.
b. Set $\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$.
c. Set $\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$.

Update the parameters:

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$