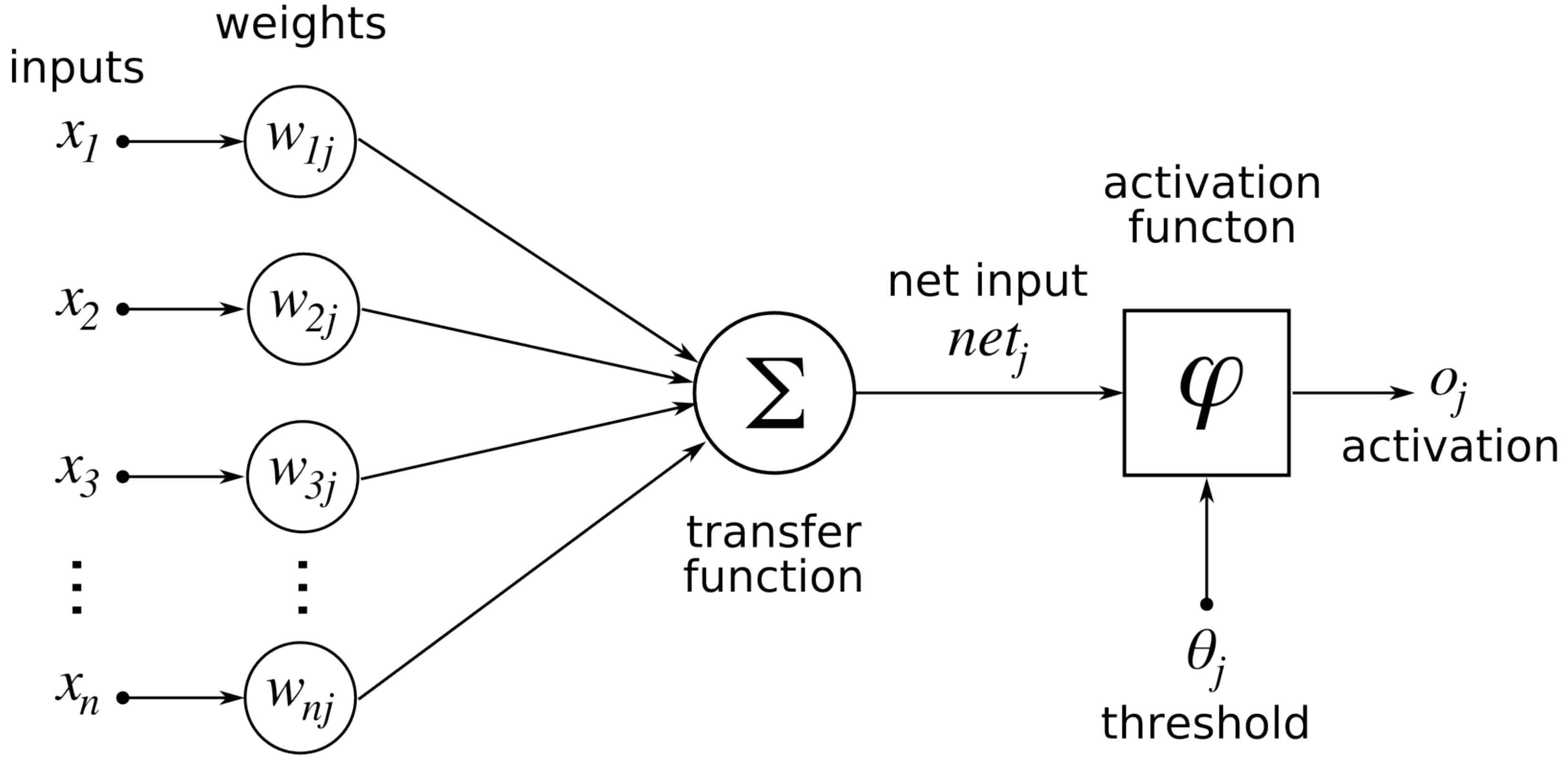
Lab 10. Artificial Neural Networks

Intro to Machine Learning
Fall 2018, Innopolis University

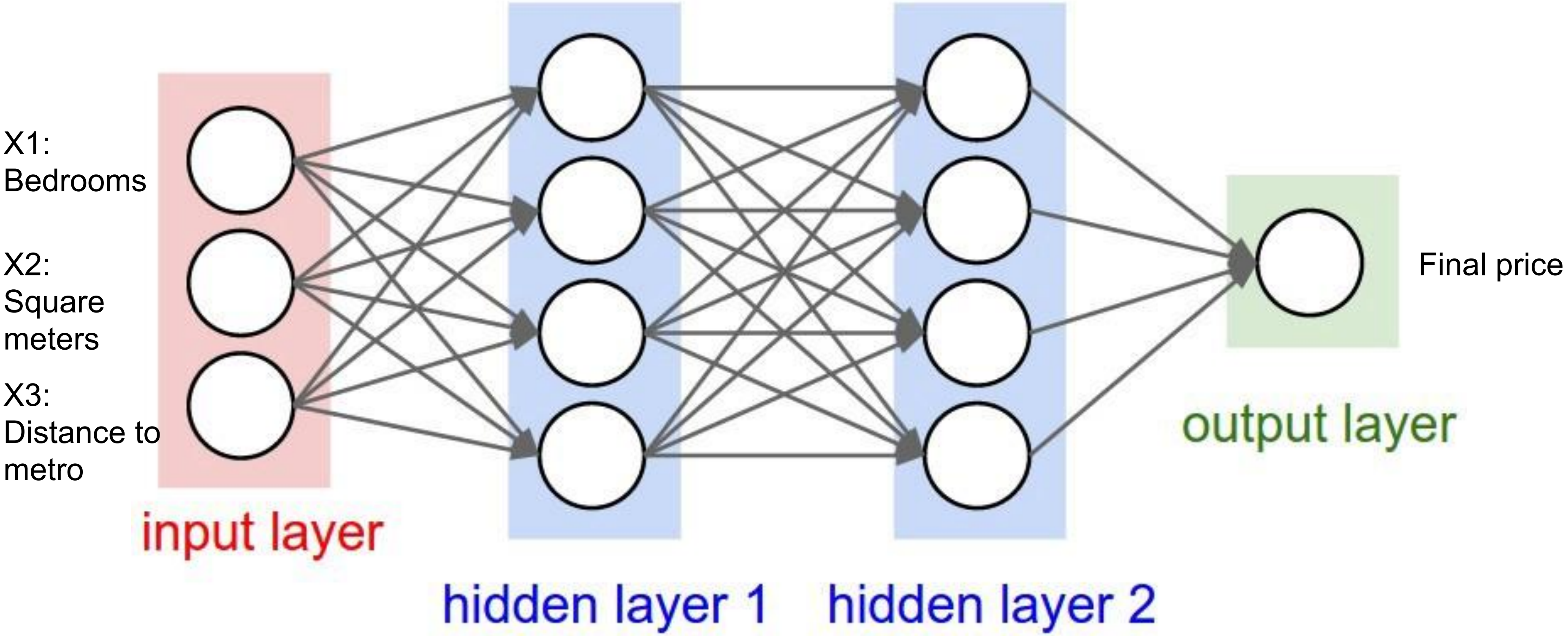
Plan

- NN layers
- Forward propagation (matrix form)
- Multiclass classification
- Activation functions
- Back propagation equations
- Homework

Artificial neuron (Perceptron) structure

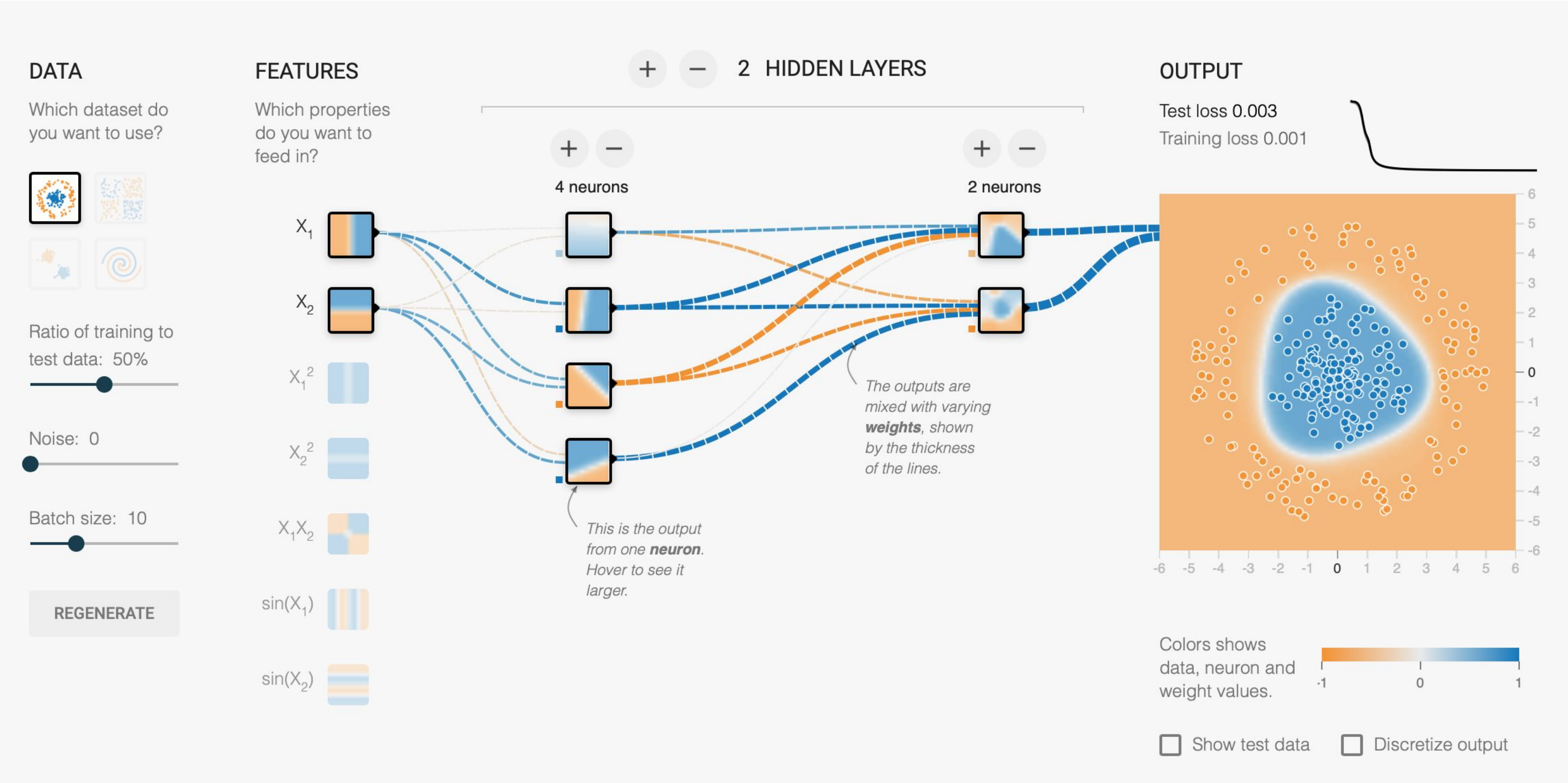


MLP (Multilayer Perceptron)

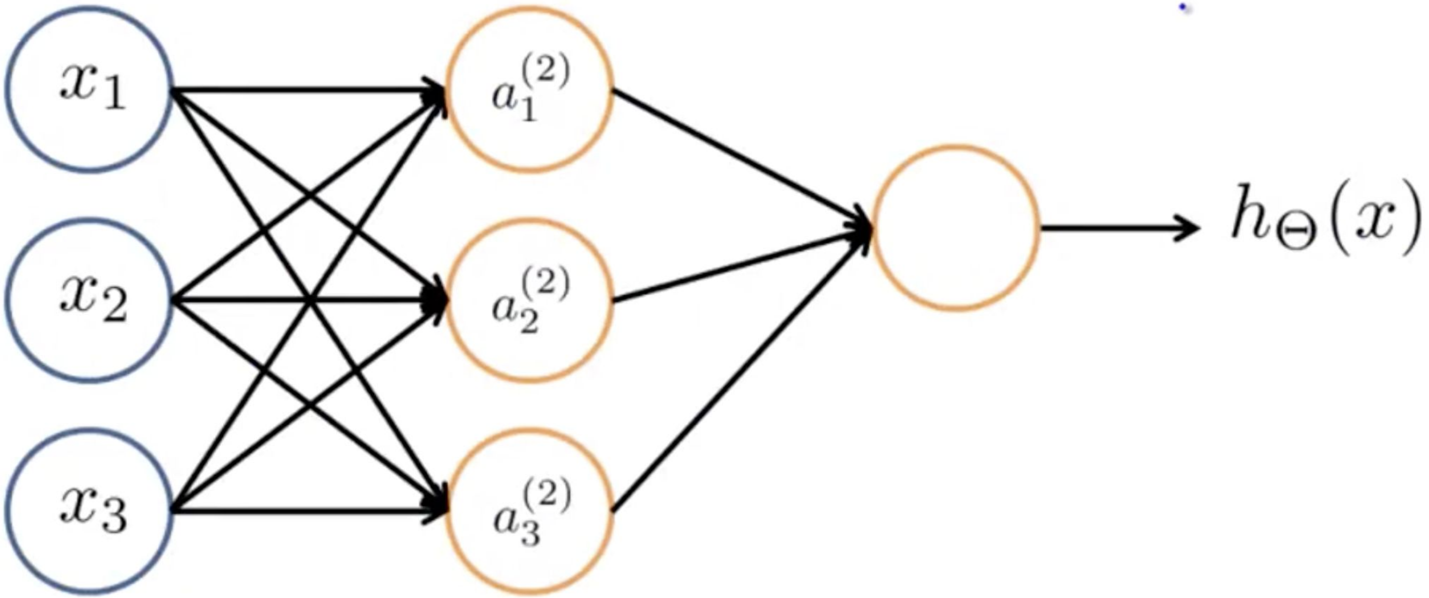


Demonstration

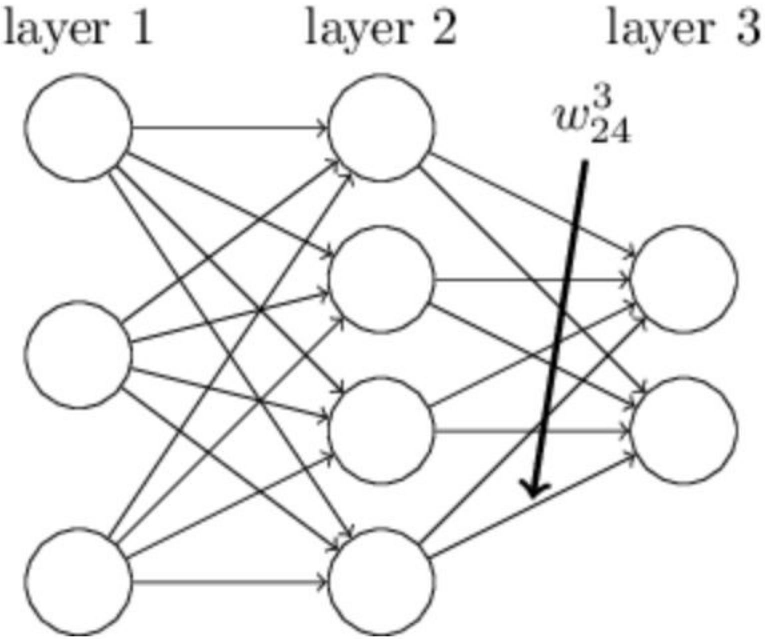
Online demonstration



Forward propagation. 1 layer example

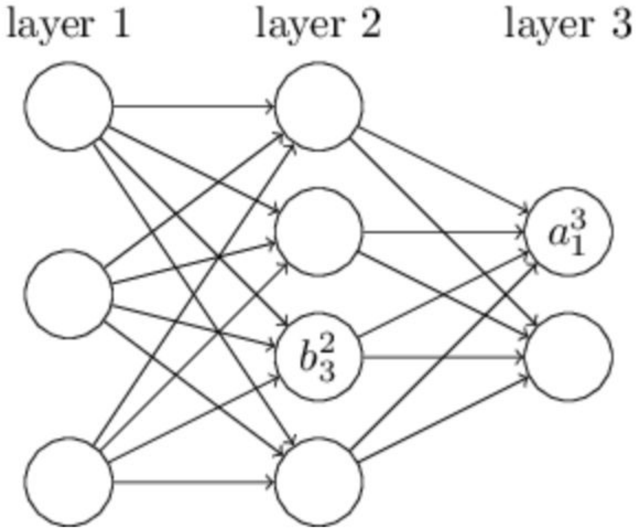


Notation. Weights



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

Notation. Activation. Bias

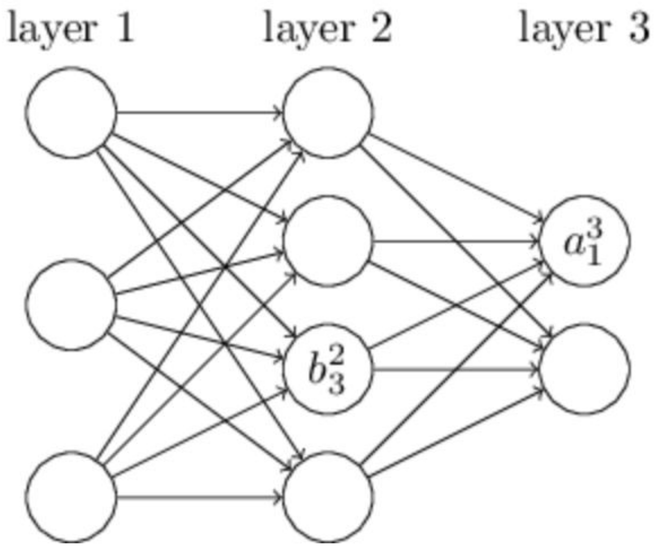


b_j^l for the bias of the j^{th} neuron in the l^{th} layer

a_j^l for the activation of the j^{th} neuron in the l^{th} layer

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

Forward propagation. 1 layer example



$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

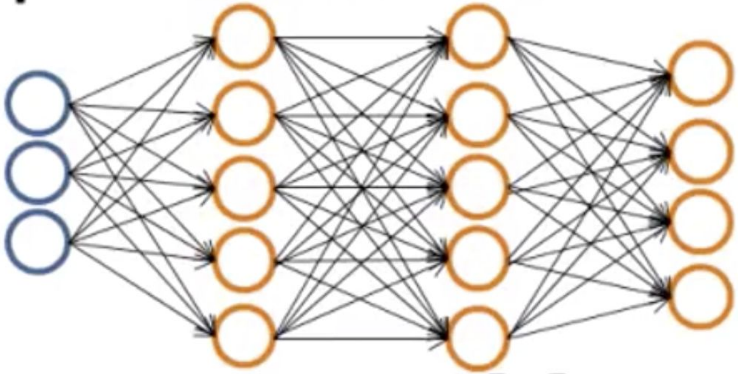
$$a_1^2 = sigmoid(w_{11}^2 a_1^1 + w_{12}^2 a_2^1 + w_{13}^2 a_3^1 + b_1^2)$$

$$a_2^2 = sigmoid(w_{21}^2 a_1^1 + w_{22}^2 a_2^1 + w_{23}^2 a_3^1 + b_2^2)$$

$$a_3^2 = sigmoid(w_{31}^2 a_1^1 + w_{32}^2 a_2^1 + w_{33}^2 a_3^1 + b_3^2)$$

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

Multiple output units: One-vs-all.



$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

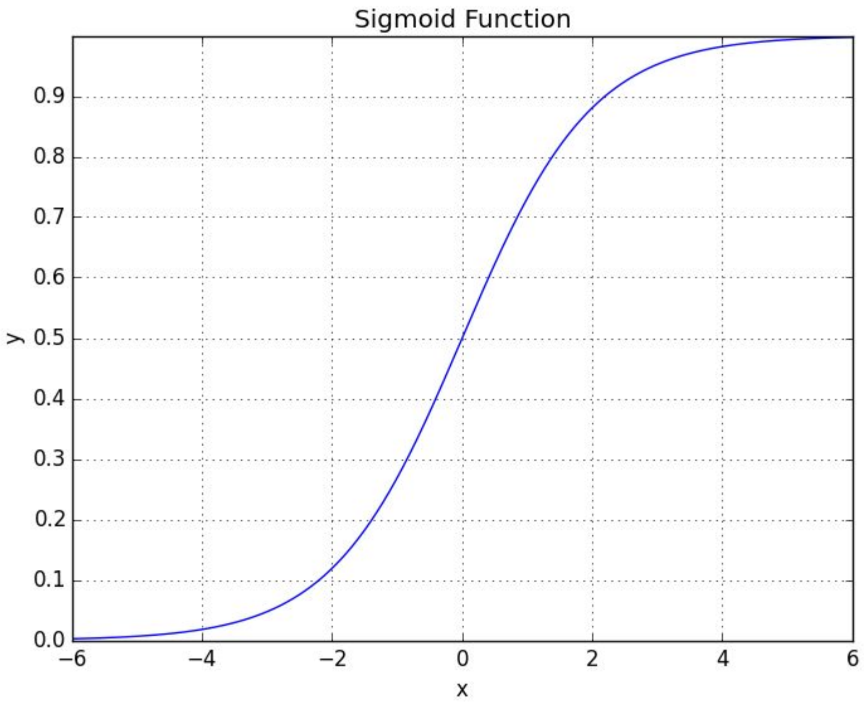
$y^{(i)}$ one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

pedestrian car motorcycle truck

Activation function

- sigmoid
- ReLU
- leaky ReLU
- tanh

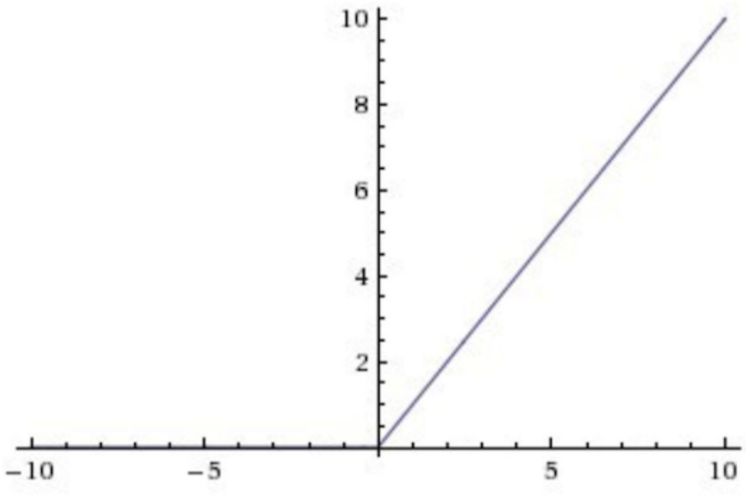
Sigmoid



$$f(x) = \frac{1}{1 + e^{-x}}$$

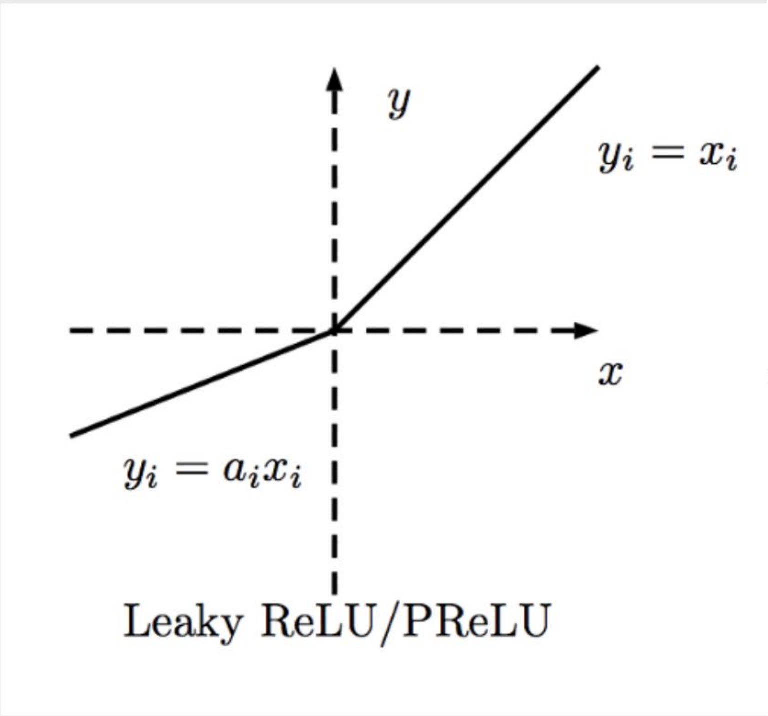
ReLU - Rectified Linear Unit

$$relu(z) = max(0, z)$$



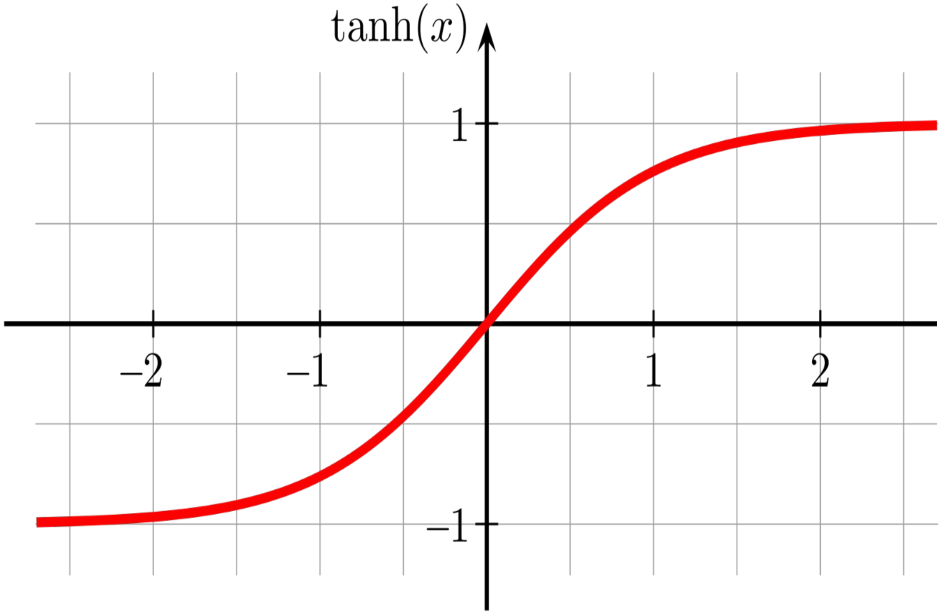
$$f(x) = \max(0, x)$$

Leaky ReLU



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

Tanh



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Function of many arguments

$$\frac{\partial}{\partial x} f(g_1(x), g_2(x), \ldots, g_k(x)) = \sum_{i=1}^{k} \frac{\partial f}{\partial g_i} \frac{\partial g_i}{\partial x}$$

Back propagation. Equation 1

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}$$

Back propagation. Equation 1

$$\boxed{\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L),}$$

Back propagation. Equation 2

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1},$$

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}$$

Back propagation. Equation 2

$$\boxed{\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l).}$$

Back propagation. Equation 3

$$\frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_k^l} \cdot \frac{\partial z_k^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} = \delta_j^l$$

Back propagation. Equation 4. Let's prove it

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j.$$

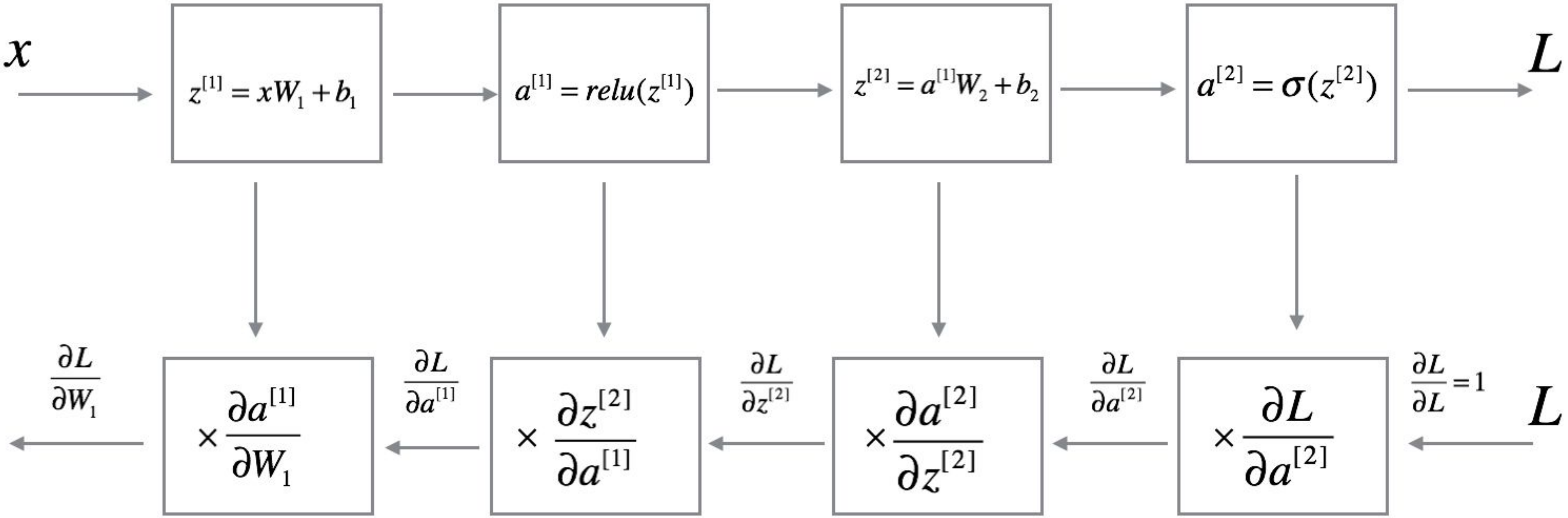
Recap

$$\boxed{\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L),}$$

$$\boxed{\frac{\partial C}{\partial b_j^l} = \delta_j^l}$$

$$\boxed{\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l).}$$

$$\boxed{\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l.}$$



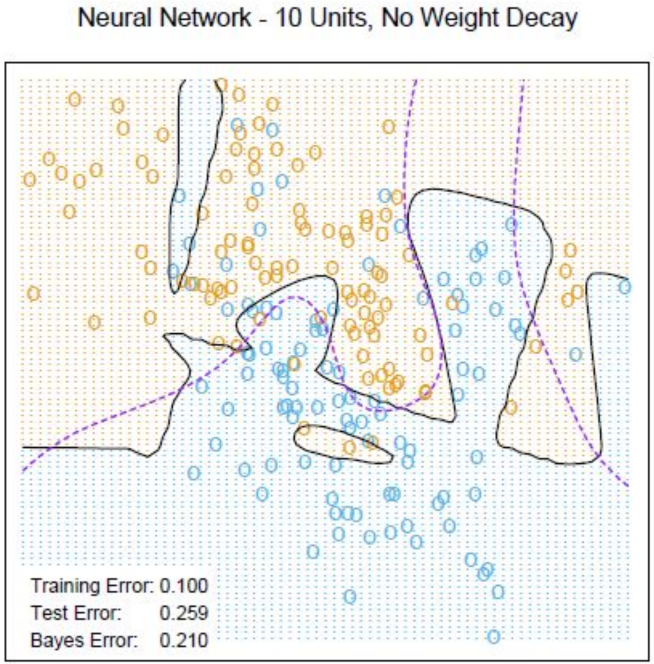
$$\boxed{\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial W_1}}$$

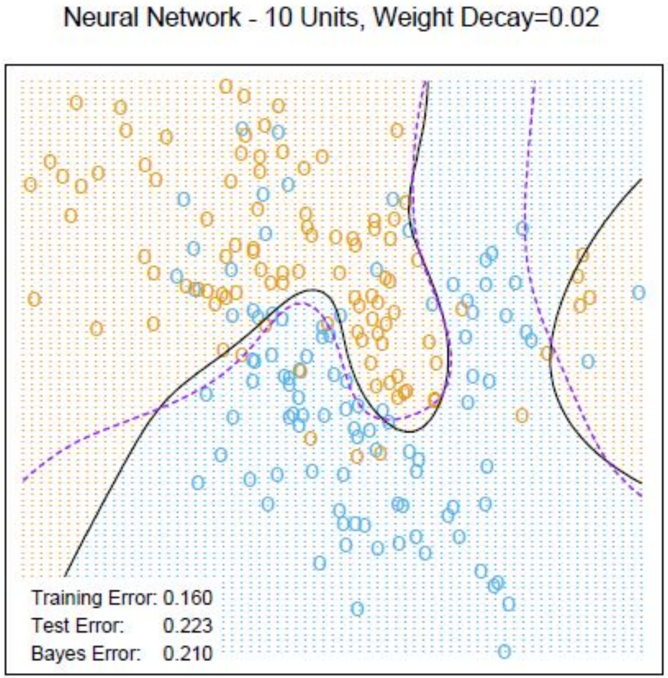
Issues in Training Neural Networks

- **Initialization of parameters** (starting weights near zero)
  Use of exact zero weights leads to zero derivatives and perfect symmetry, and the algorithm never moves. Starting instead with large weights often leads to poor solutions.

Issues in Training Neural Networks

- **Overfitting** (regularization, weight decay)





Issues in Training Neural Networks

- **Scaling the input** (mean=0, std=1)
  At the outset it is best to standardize all inputs to have mean zero and standard deviation one. This ensures all inputs are treated equally in the regularization process, and allows one to choose a meaningful range for the random starting weights.

Issues in Training Neural Networks

- **Number of hidden layers and units** (higher better)
  With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data; with too many hidden units, the extra weights can be shrunk toward zero if appropriate regularization is used.

Issues in Training Neural Networks

- **Multiple minima** (random start)
  One must at least try a number of random starting configurations, and choose the solution giving lowest (penalized) error. Probably a better approach is to use the average predictions over the collection of networks as the final prediction (Ripley, 1996)

Homework

Please read the code (hw10.py) and do the Following.

A. We used a sigmoid activation function in our hidden layer. Please write 2 more activation functions:
   a. ReLU
   b. Tanh

   Try to change activations in our hidden layer. Remember that changing the activation function also means changing the backpropagation derivative.

B. Add one more hidden layer to this network. Experiment with the layers sizes. Remember that after adding hidden layer you need to adjust the forward propagation and the backpropagation code as well.