Introduction to Neural Networks

Objectives

- relate neural networks to linear classifiers
- define structure of multilayer neural network
- overview procedure for training neural networks

The "neuron" generalizes a linear classifier 2

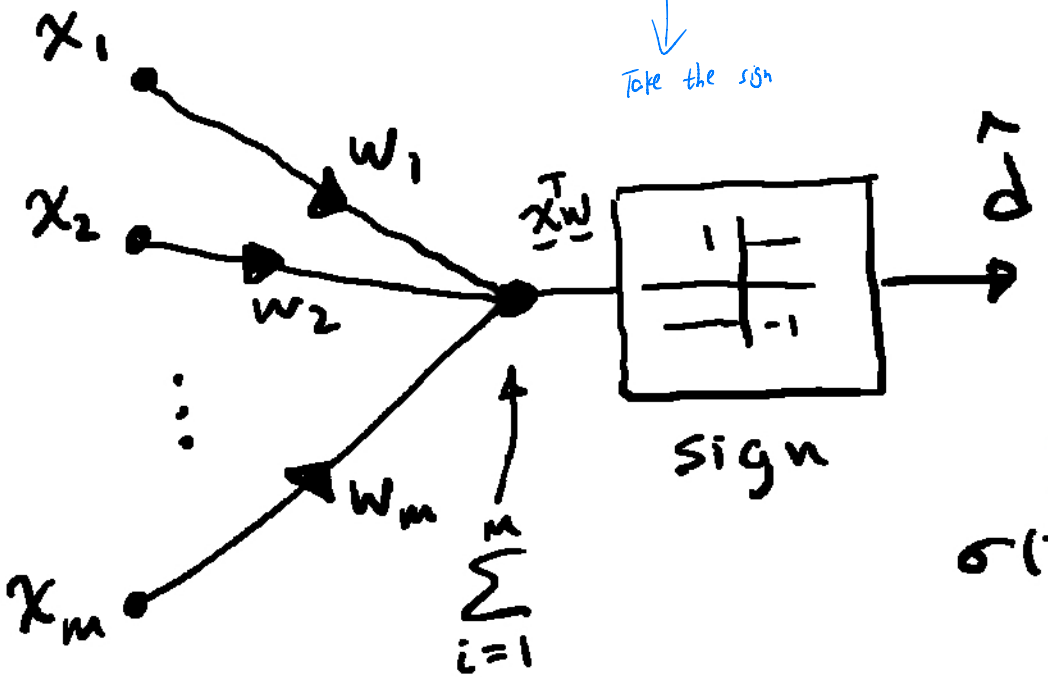
Feature: $\underline{x}^T = [x_1 \; x_2 \; \cdots \; x_m]$ (m features)

Weights: $\underline{w}^T = [w_1 \; w_2 \cdots \; w_m]$ (m corresponding weights)

Linear classifier

$$\hat{d} = \text{sign}(\underline{x}^T \underline{w})$$

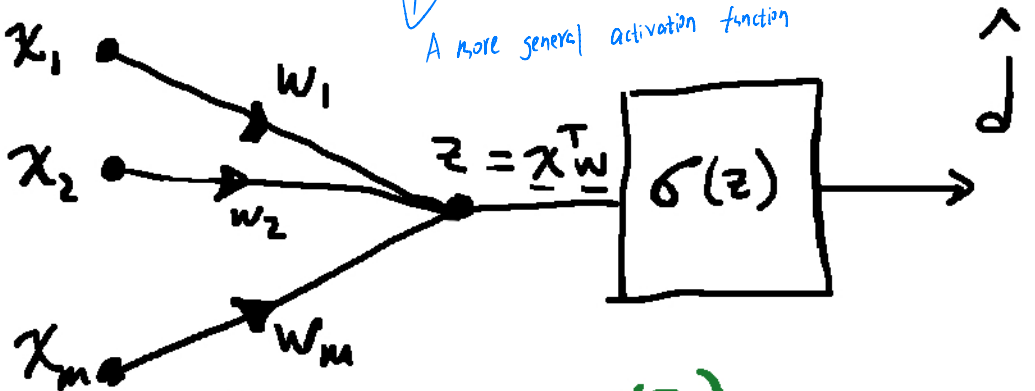
Take the sign



Neuron

$$\hat{d} = \sigma(\underline{x}^T \underline{w})$$

σ: A more general activation function



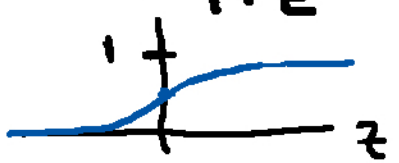
Common $\sigma(z)$

ReLU

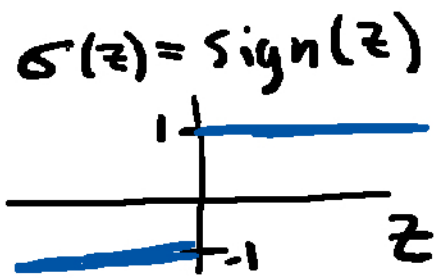
$\sigma(z) = \max\{0, z\}$

Logistic

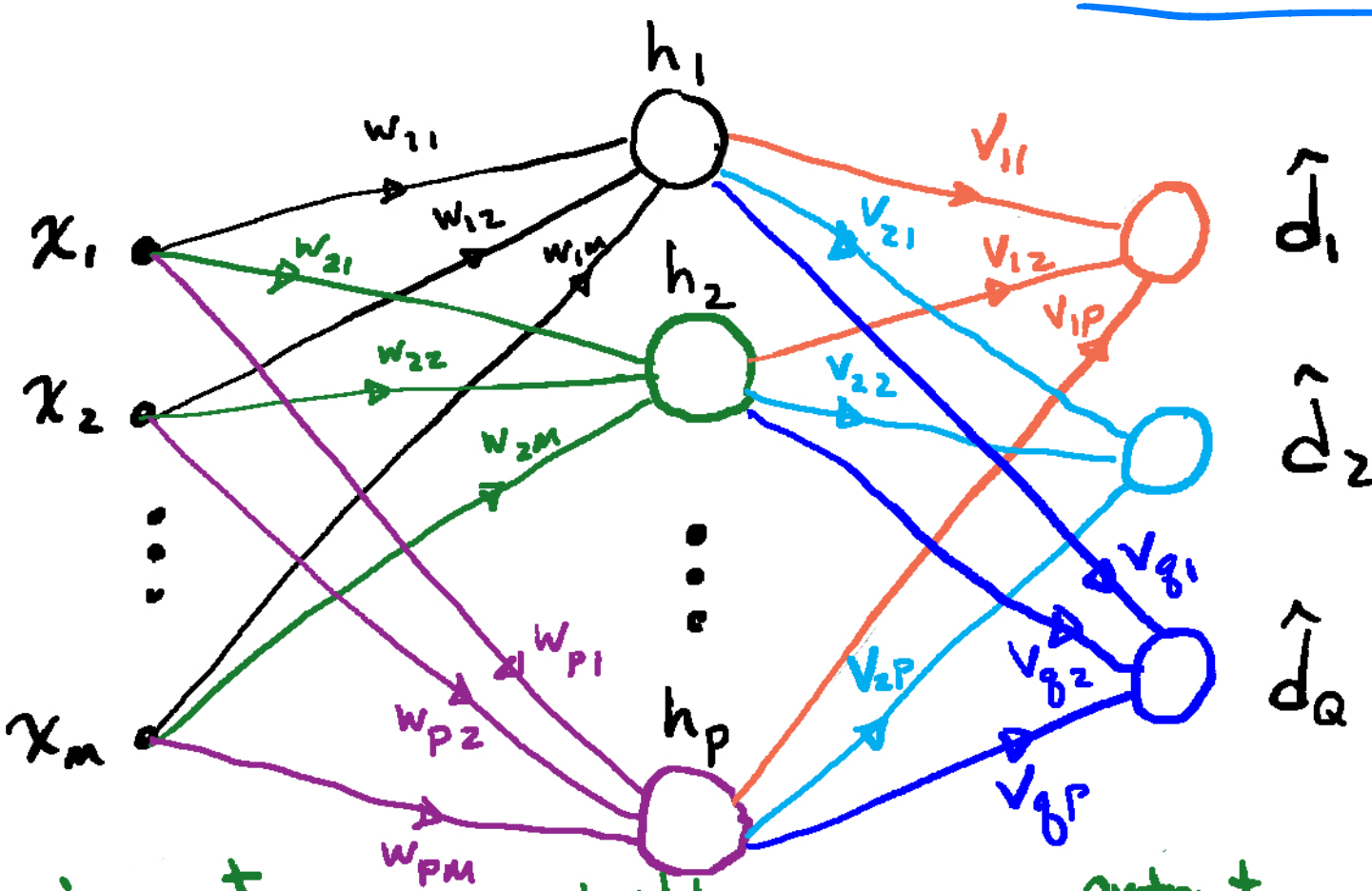
$\sigma(z) = \frac{1}{1+e^{-z}}$



$\sigma(z) = \text{sign}(z)$



A neural network is a network of neurons



input layer

hidden layer

output layer

shallow → deep

multiple outputs → solve multiple problems

$$h_l = \sigma\left(\sum_{j=1}^{m} w_{lj} x_j\right)$$

$$\hat{d}_m = \sigma\left(\sum_{n=1}^{p} v_{mn} h_n\right)$$

$$= \sigma\left(\sum_{n=1}^{p} v_{m,n}\, \sigma\left(\sum_{j=1}^{m} w_{nj} x_j\right)\right)$$

"deep" learning → many hidden layers

Two issues must be addressed to use NNs

1) Network structure: number of layers, number of hidden nodes in each layer

Open question. Universal approximation theorem (1991): Three layer network can approximate any function arbitrarily well given enough hidden nodes and the right weights

2) Choosing the weights

Stochastic gradient descent and backpropagation

Nonconvex → local minima

Backpropagation updates each layer in sequence

– work back from deep (output) to shallow (input)

$$\min_{w_{lj}, v_{lj}, \ldots} \sum_{i=1}^{N} \sum_{q=1}^{Q} \frac{1}{2} \left(\hat{d}_{i,q} - d_{i,q} \right)^2$$

Objective; weights; # of samples; # of outputs; generated label; true label

N training samples, Q outputs

1) Initial guess on w_{lj}, v_{lj} (etc)

2) Randomly choose training sample i

3) Calculate $h_{i,p}, \hat{d}_{i,q}$

4) Gradient descent update v_{lj}, then w_{lj} (deep to shallow)

Chain rule is key for deriving gradients