Nonlinear Function Approximation with Neural Nets[1]

[1] Sections 4.2: Yang&Ying

Nonlinear function approximation using neural networks

Nonlinear Function Approximation

Consider nonlinear function approximation parameterized by w, i.e. $\hat{Q}(x, u; w)$ is an estimate of $Q(x, u)$. (x, u) is the input of the function and w are the weights.

TD Learning

Similar to TD learning with linear function approximation, we want to find w to minimize

$$\frac{1}{2}\mathbb{E}\left[\left(\hat{Q}(x, u; w) - r(x, u) - \alpha \max_{v} \hat{Q}(x', v; w)\right)^2\right].$$

Nonlinear function approximation using neural networks

TD Learning

Given an experience (x_k, u_k, r_k, x_{k+1}), we

$$\frac{1}{2}\delta_k^2(w) = \frac{1}{2}\left(\hat{Q}(x_k, u_k; w) - r_k - \alpha \max_v \hat{Q}(x_{k+1}, v; w)\right)^2.$$

The (improper) gradient is

$$\begin{aligned}\frac{1}{2}\tilde{\nabla}\delta_k^2(w) &= \left(\hat{Q}(x_k, u_k, w) - r_k - \alpha \max_v \hat{Q}(x_{k+1}, v; w)\right)\nabla \hat{Q}(x_k, u_k, w) \\ &= \delta_k(w)\nabla \hat{Q}(x_k, u_k, w).\end{aligned}$$

Nonlinear function approximation using neural networks

- r_k is from the sample
- $\max_v \hat{Q}(x_{k+1}, v; w)$ and $\hat{Q}(x_k, u_k; w)$ can be computed from the neural network.
- Question: how to compute

$$\nabla_w \hat{Q}(x_k, u_k; w)?$$

Nonlinear function approximation using neural networks

Preliminaries:

- A neuron is a nonlinear function which takes $x \in \mathbb{R}$ as input and produce $\sigma(x) \in \mathbb{R}$.

$$x \to \sigma(x)$$

Examples:

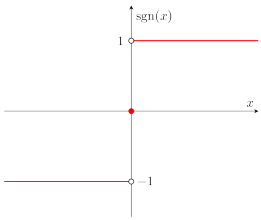


Figure: Sign function

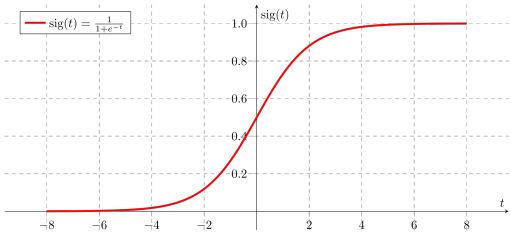


Figure: Sigmoid function

Nonlinear function approximation using neural networks

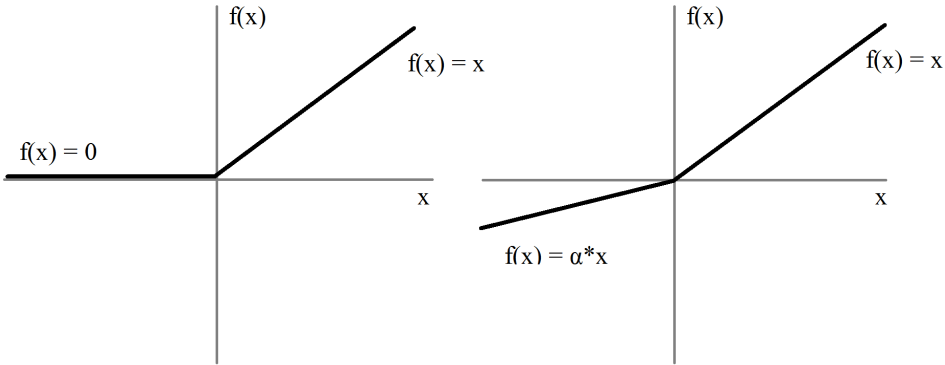
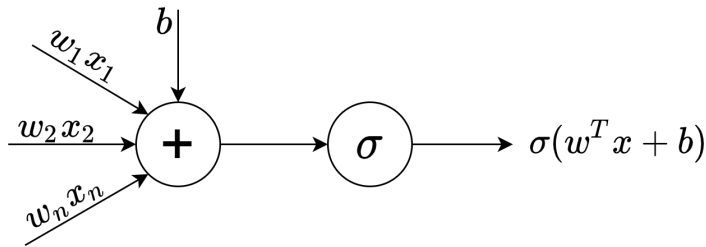


Figure: ReLU (Rectifier Linear Unit) and Leaky ReLU

A Single Neuron View

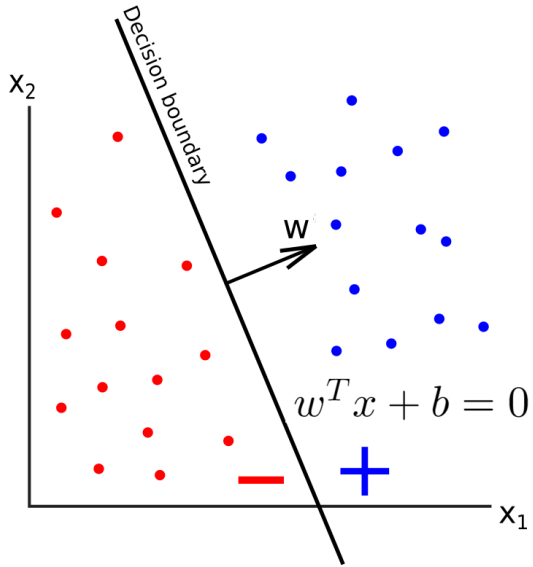
- Vector input: $\sigma(w^T x + b)$



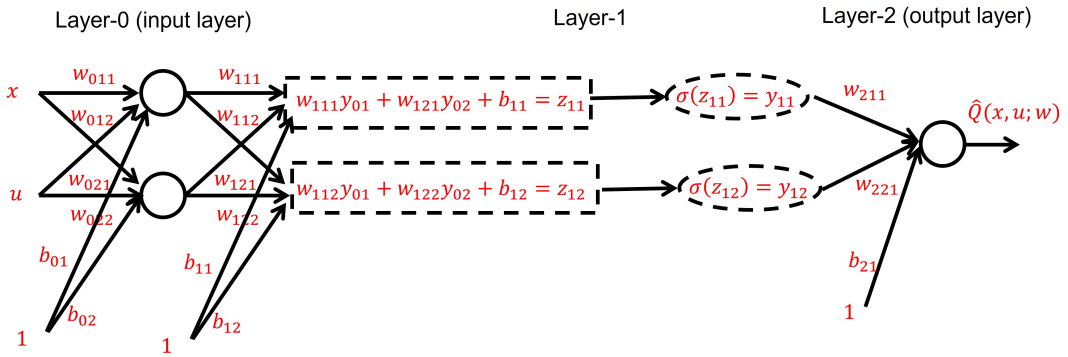
where b is the bias.

- $w^T x + b = 0$ is a hyperplane, which divides the input space into two parts.

Nonlinear function approximation using neural networks



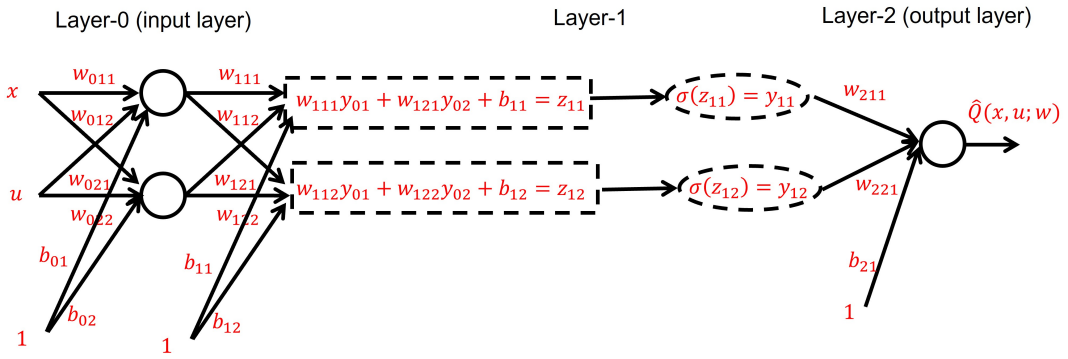
Multilayer Feedforward Neural Nets



Fixing (x, u), $\hat{Q}(x, u; w)$ is a function of w. How to compute

$$\frac{\partial \hat{Q}(x, u; w)}{\partial w_{lij}}?$$

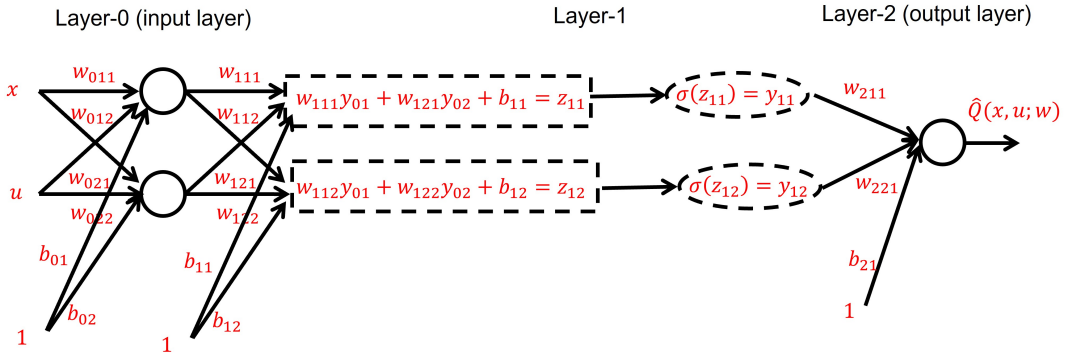
Multilayer Feedforward Neural Nets



$$\frac{\partial \hat{Q}(x,u;w)}{\partial w_{211}} = \frac{\partial \sigma(z_{21})}{\partial z_{21}} \frac{\partial z_{21}}{\partial w_{211}} = \sigma'(z_{21}) y_{11}.$$

where $z_{21} = w_{211}y_{11} + w_{221}y_{12} + b_{21}$. Note that z_{21} and y_{11} are computed using the nn. $\sigma'(\cdot)$ can be computed locally at the given neuron.

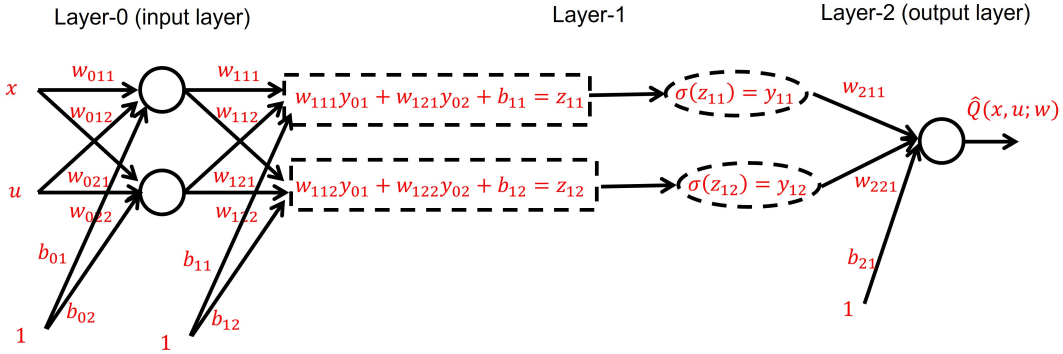
Multilayer Feedforward Neural Nets



$$\frac{\partial \hat{Q}(x,u;w)}{\partial w_{111}} = \frac{\partial \sigma(z_{21})}{\partial z_{21}} \frac{\partial z_{21}}{\partial y_{11}} \frac{\partial y_{11}}{\partial z_{11}} \frac{\partial z_{11}}{\partial w_{111}} = \sigma'(z_{21}) w_{211} \sigma'(z_{11}) y_{01}.$$

Note that z_{11} and y_{01} are computed using the nn (forward propagation). $\sigma'(z_{21})$ and w_{211} can be obtained from layer-2 (backward propagation).

Deep Q-Learning



Deep Q-Learning

Given experience (x, u, r, x'), where x' is the next state,

$$w_{111} \leftarrow w_{111} - \beta\delta(w)\frac{\partial \hat{Q}(x, u; w)}{\partial w_{111}}$$