Nonlinear Function Approximation: Neural Networks

Qiang Liu
UT Austin

Approaches to Nonlinear Regression:

- Fixed basis function:

$$f(x;\ w) = \sum_i w_i \phi_i(x).$$

e.g., polynomial regression: $f(x;\ w) = w_0 + w_1 x + w_2 x^2 + \cdots$.

- Adaptive basis functions:
  - Kernel method:

$$f(x;\ w) = \sum_{i \in data} w_i k(x, x_i).$$

  - Neural networks:

$$f(x; w, v) = \sum_i w_i \phi(x; v_i).$$

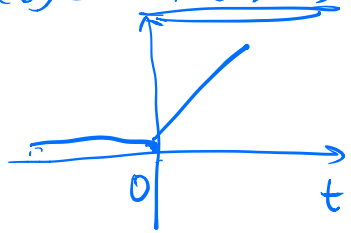
$\phi_i(x) = \phi(x, v_i)$

$$\min_{w,v} E_D\left[(y - f(x; w, v))^2\right]$$

$$\phi(x) = \sigma\left(\sum_{\ell=1}^{d} v_\ell x_\ell + v_0\right) \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

$\sigma(\cdot)$: activation function $\qquad v = \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix}$

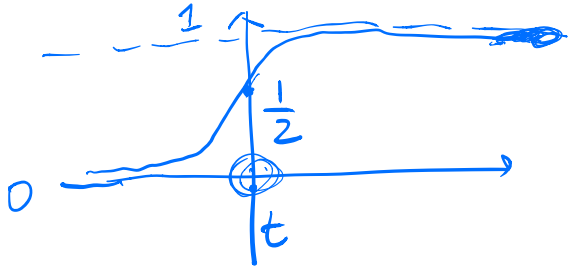
$\sigma(t) = \max(0, t)$: Rectified Linear Unit (ReLU)



$$\sigma(t) = \begin{cases} 0 & t \le 0 \\ t & t > 0 \end{cases}$$

$$\sigma(t) = \frac{\exp(t)}{1+\exp(t)} = \frac{1}{1+\exp(-t)}$$

Sigmoid function.

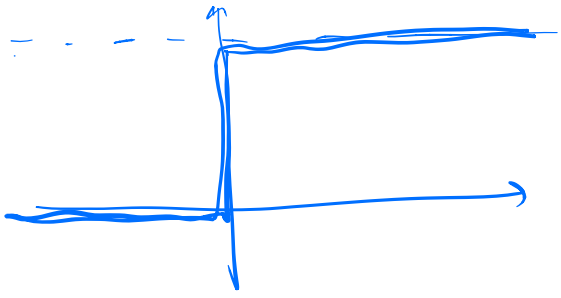


$\phi(x)$: Neuron.

$\sigma(\cdot)$: activation

v : weights

$$\sigma(t) = \mathbb{I}(x>0) = \begin{cases} 1 & x>0 \\ 0 & x \le 0 \end{cases}$$



$$f(x; w, V) = \sum_{i=1}^{m} w_i \sigma\left(\sum_{\ell=1}^{d} v_{i\ell} x_\ell + v_{i0}\right)$$

$$v_i = [v_{i0}, v_{i1} \cdots v_{id}]$$

$$V = [v_1 \cdots v_m]$$

* Learning Neural Nets is Non-convex opt.

$$f(x, V) = \sigma(x - v_1) + \sigma(x - v_2)$$

$$L(V) = E_D\left[\left(y - \sigma(x-v_1) - \sigma(x-v_2)\right)^2\right]$$

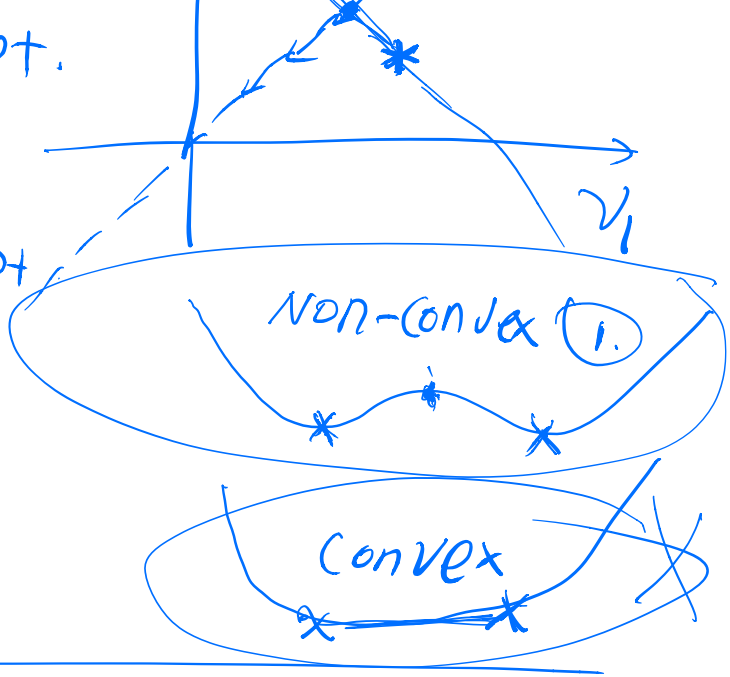
Assume $[v_1^*, v_2^*]$ is opt.

then $[V_2^*, V_1^*]$ is also Opt.

If $L(V)$ is convex

then $[\bar{V}^*, \bar{V}^*]$ is also opt

$$\bar{V}^* = \frac{V_1^* + V_2^*}{2}$$



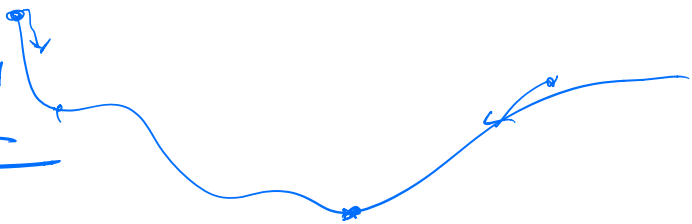
Gradient descent

$$\min_{W,V} L(W, V)$$

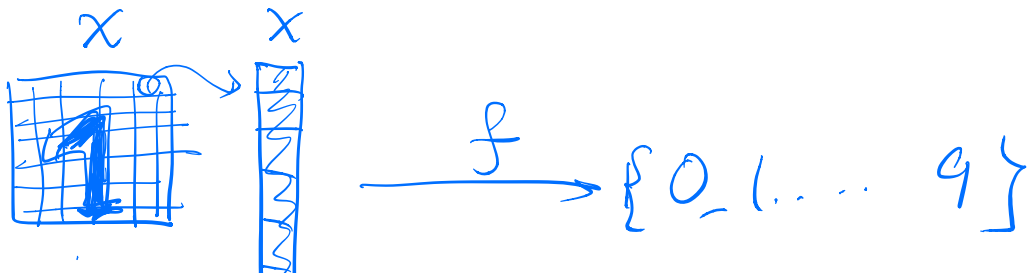
$$\begin{cases} W \leftarrow W - \epsilon \nabla_W L(W, V) \\ V \leftarrow V - \epsilon \nabla_V L(W, V) \end{cases}$$

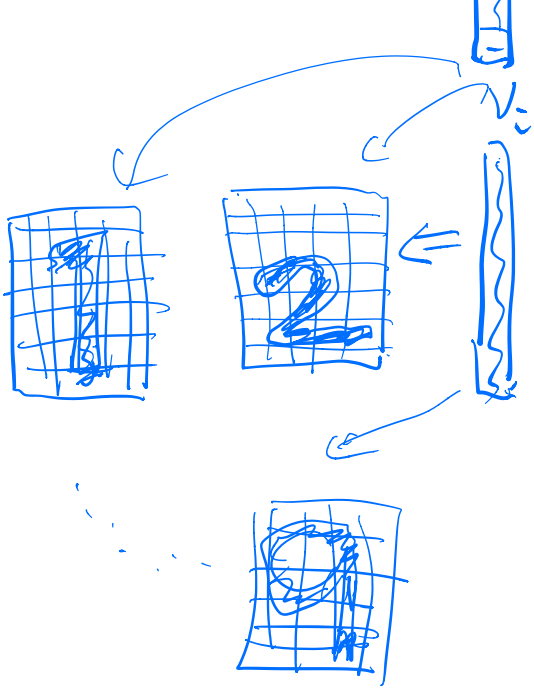
Initialize V

to be "small Random Values"



Digit classification

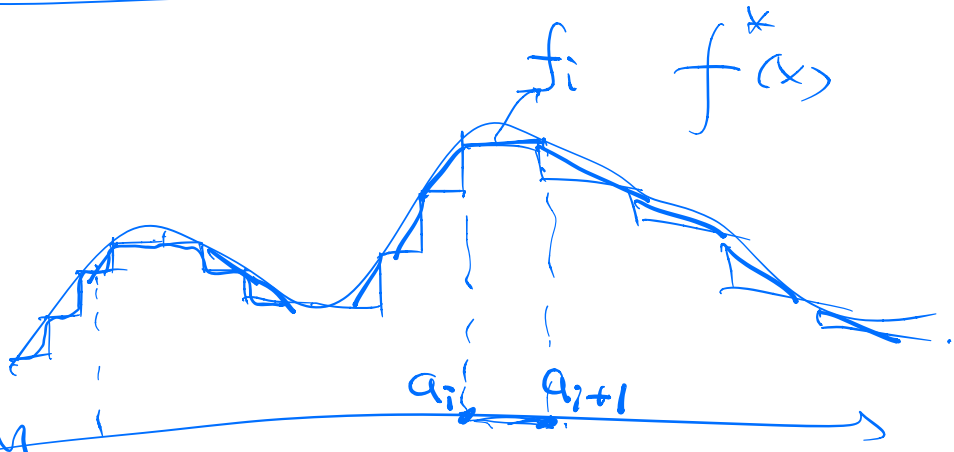




$$\sum_{i=1}^{m} w_i \sigma\left(v_i^T x - v_0\right)$$

$$\sum_{i=1}^{m} w_i \, \mathbb{I}\left(v_i^T x \geq v_0\right)$$

If m is large enough, $f(x, \underline{w}, \underline{v})$ can approximate any continous function



$$\sum_{i=1}^{M} f_i \, \mathbb{I}\left(a_i \leq x \leq a_{i+1}\right)$$

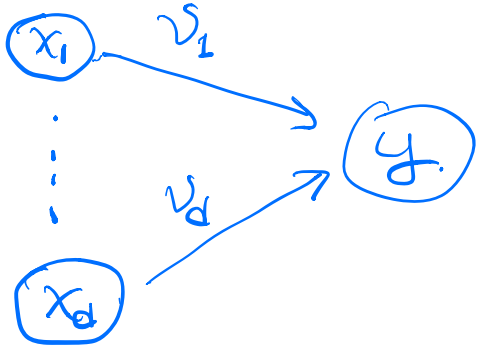
$$\sum_{i=1}^{M} f_i \left(\mathbb{I}(x > a_i) - \mathbb{I}(x > a_{i+1})\right)$$

$$= \sum_{i=1}^{M} f_i \left(\mathbb{I}(x \geq a_i) - \mathbb{I}(x \geq a_{i+1}) \right)$$

$$= \boxed{\sum_{i=1}^{M} (f_i - f_{i-1}) \, \mathbb{I}(x \geq a_i)}$$

Graphical Representation of NN

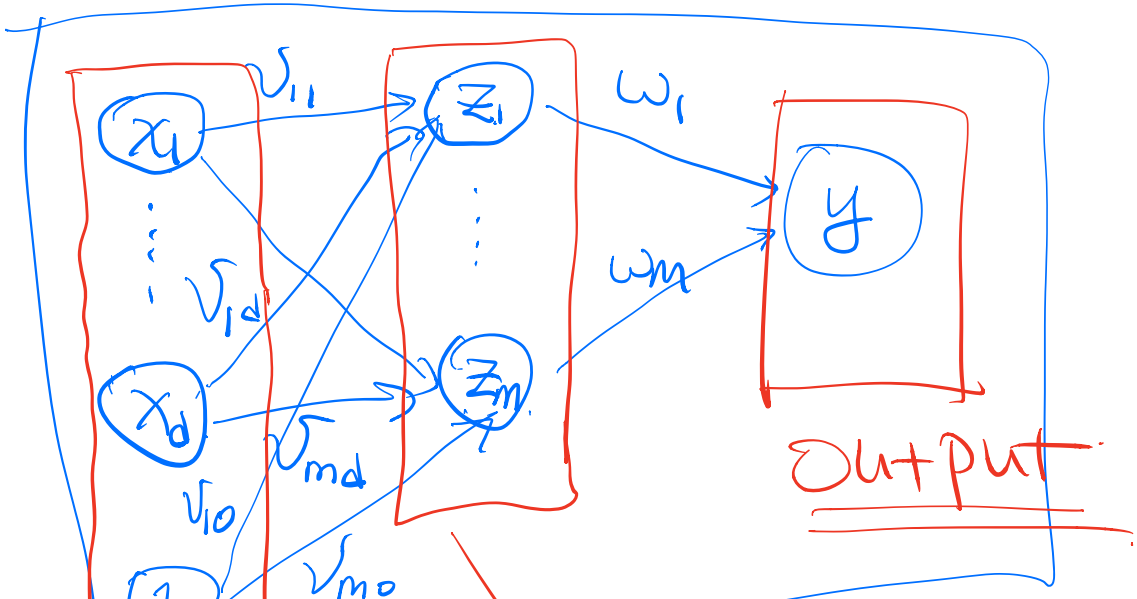
$$y = \sum_{\ell=1}^{d} v_\ell x_\ell$$



$$y = \sum_{i=1}^{M} w_i \, \sigma\left(\sum_{\ell=1}^{d} v_{i\ell} x_\ell + v_{i0} \right)$$

$$\triangleq z_i$$

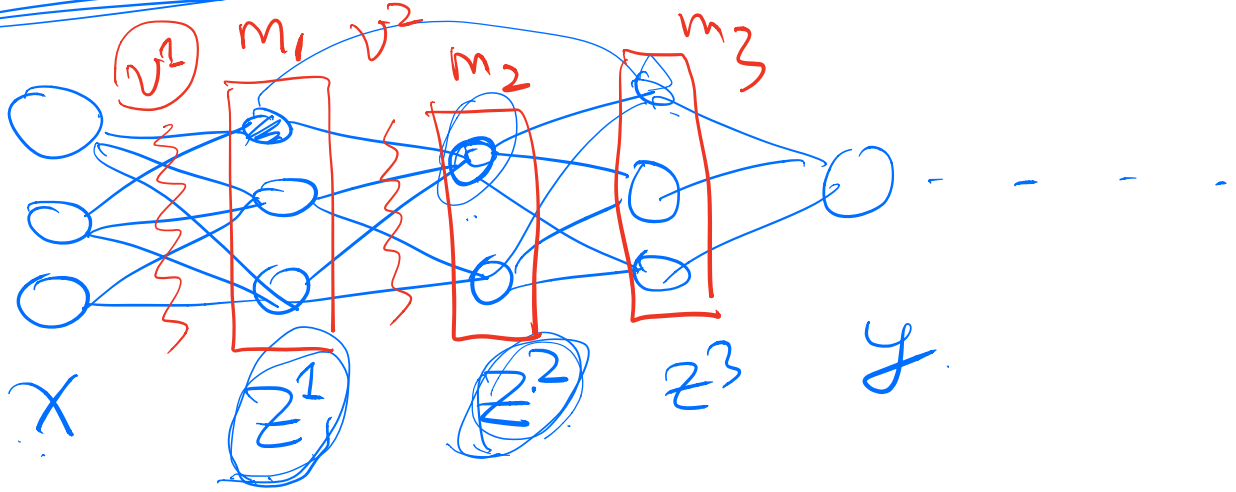
Neural Network



Input Layer

Hidden Layer.

Deep Neural Networks

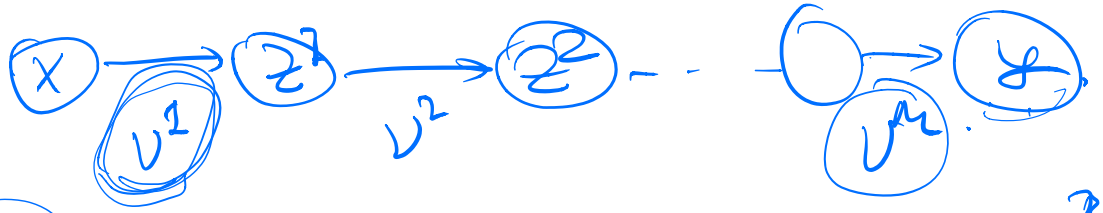


$$z^1 = \Phi(x, v^1) = \begin{bmatrix} \sigma((v_1^1)^T x) \\ \vdots \\ \sigma((v_{m_1}^1)^T x) \end{bmatrix}$$

$$z^2 = \Phi(z^1, v^2) = \begin{bmatrix} \sigma((v_1^2)^T z^1) \\ \vdots \end{bmatrix}$$

$$y = \sum_{i=1}^{m_3} w_i z_i^3$$

Back-propagation = Calculation Gradient of Deep Neural Nets



$$L(v) = E_D\left[(y - f(x, v^1 \dots v^m))^2\right]$$

$$\nabla_{v^1} L(v) = 2 E_D\left[(f(x,v) - y)\frac{df(x,v)}{dv^1}\right]$$

$$\frac{df(x,v)}{dv^1} = \frac{dy}{dv^1} = \left(\frac{dy}{dz^m}\right)\left(\frac{dz^m}{dz^{m-1}}\right)\cdots\left(\frac{dz^3}{dz^2}\right)\left(\frac{dz^2}{dz^1}\right)\left(\frac{dz^1}{dv^1}\right)$$