object $\xrightarrow[\text{representation}]{\text{data}}$ input space $\xrightarrow[\text{engineering}]{\text{feature}}$ feature space

Deep learning:

$\rightarrow$ 1st layer $\rightarrow$ 2nd layer $\rightarrow \cdots \rightarrow$ nth layer

abstraction level $\uparrow$

Supervised

Shallow

Deep

unsupervised

$\varphi(x)$ or $K(x,x)$ given

SVM

logistic regression

perceptron

deep NN

CNN

RNN (recurrent)

denoising autoencoder

restricted Bolzman machine

sparse coding

variational encoder

GAN (generative adversial network)

transformer

diffusion models

Types of neurons

Linear neurons

(通常用在 output layer of regression)

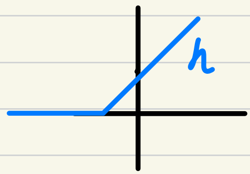
$$h = b + \sum_i w_i x_i$$

rectified linear neurons (Relu):

(a "threshold")

$$z = b + \sum_i w_i x_i$$

$$h = \max(z, 0)$$

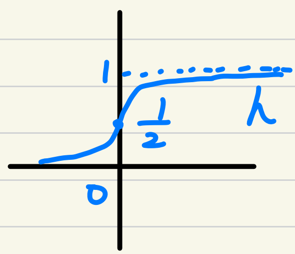


sigmoid (logistic) neurons

(have nice derivatives; 通常用在 hidden layers, 增加非线性)

$$z = b + \sum_i w_i x_i$$

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

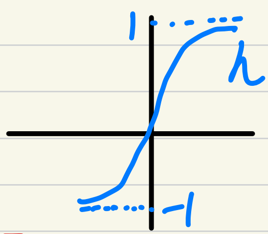


tanh neurons

(similar to σ, but larger derivative)

$$z = b + \sum_i w_i x_i$$

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



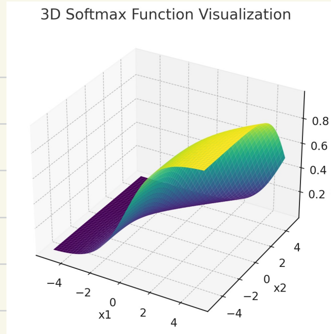
* $\sigma(z) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{z}{2}\right)$. 因而 tanh neuron 等价于 σ neuron (by shifting & scaling)

softmax neurons

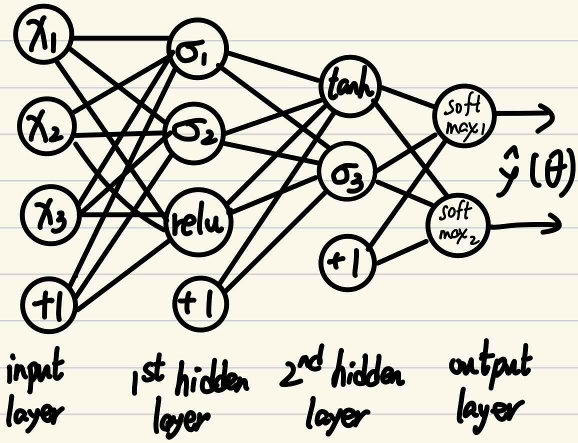
(通常用在 output layer of classification;)

$$z_j = b_j + \sum_i x_i w_i^{(j)}$$

$$h_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



ex of multilayer NN



train NN

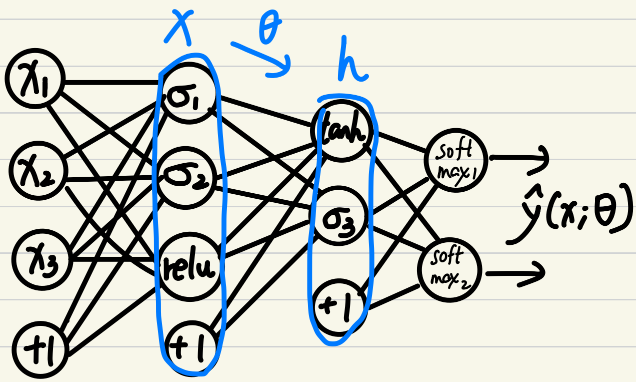
repeat till conv:

- (x,y) sample
- $\hat{y} \leftarrow f(x;\theta)$ forword propagation, 即 predict
- compute $L(y,\hat{y})$
- compute $\nabla_\theta L$ backword propagation, 即使用 chain rule 算 grad
- $\theta \leftarrow \theta - \alpha \nabla_\theta L$ (GD, can be stochastic & batch)

we denote $\begin{cases} x\in\mathbb{R}^D \\ h\in\mathbb{R}^N \\ \theta\in\mathbb{R}^M \end{cases}$ as $\begin{cases} \text{input} \\ \text{output} \\ \text{parameter} \end{cases}$ of a layer

$$\cdots \longrightarrow \underbrace{x}_{(n-1)^{th}\text{ layer}} \xrightarrow{\theta} \underbrace{h}_{n^{th}\text{ layer}} \longrightarrow \cdots \longrightarrow \hat{y}$$

ex



$$\cdots \longrightarrow \underbrace{x}_{(n-1)^{th}\text{ layer}} \xrightarrow{\theta} \underbrace{h}_{n^{th}\text{ layer}} \xrightarrow{\gamma} \underbrace{f}_{(n+1)^{th}\text{ layer}} \longrightarrow \cdots \longrightarrow \hat{y}$$

$$\frac{\partial L}{\partial h_j} = \sum_i \frac{\partial L}{\partial f_i}\frac{\partial f_i}{\partial h_j} \longrightarrow \cdots$$

By chain rule, $\frac{\partial L}{\partial \theta_k} = \sum_j \frac{\partial L}{\partial h_j}\frac{\partial h_j}{\partial \theta_k}$

可以通过 recursion 得到结果,

specially, $\frac{\partial L}{\partial \hat{y}_i}$ can be computed directly through the formula of $L(y,\hat{y})$, 作为 chain rule 的展开的最里层

vectorization form of chain rule:

$$\underbrace{\nabla_\theta L}_{\in \mathbb{R}^N} = \underbrace{\left(\frac{\partial h}{\partial \theta}\right)^T}_{\in \mathbb{R}^{N\times m}} \underbrace{\nabla_h L}_{\in \mathbb{R}^m}$$

where $\frac{\partial h}{\partial \theta}$ is the Jacobian matrix (即 derivative, if $\exists$)

*My remark:

正常而言，$\frac{\partial L}{\partial \theta}(\alpha) \in \mathbb{R}^{1\times M}$ 表示 derivative: $\left(\frac{\partial L}{\partial \theta_1}(\alpha), \dots, \frac{\partial L}{\partial \theta_m}(\alpha)\right)$

$\nabla_\theta L(\alpha) \in \mathbb{R}^M$ 表示 gradient: $\begin{pmatrix} \frac{\partial L}{\partial \theta_1}(\alpha) \\ \vdots \\ \frac{\partial L}{\partial \theta_m}(\alpha) \end{pmatrix}$

而 for $\theta \mapsto h$

$\mathbb{R}^M \mapsto \mathbb{R}^N$

Jacobian $J_\theta(\alpha) := \frac{\partial h}{\partial \theta}(\alpha) \in \mathbb{R}^{N\times m}$

with derivative $Dh(\alpha) = J_\theta(\alpha)$ if Dh exists at α

而 $\frac{\partial L}{\partial \theta}(\alpha) = \frac{\partial L}{\partial h}(h(\alpha))\frac{\partial h}{\partial \theta}(\alpha)$ (chain rule)

$$\nabla_\theta L(\alpha) = \frac{\partial L}{\partial \theta}^T(\alpha) = \frac{\partial h}{\partial \theta}^T(\alpha)\,\frac{\partial L}{\partial h}^T(h(\alpha))$$

$$= J_\theta^T(\alpha)\,\nabla_h L(\alpha)$$

而 ml 里则对于 $\mathbb{R}^M \to \mathbb{R}$ 的函数，用 $\frac{\partial L}{\partial \theta}$ 表示 $\nabla_\theta L$

对于 $\mathbb{R}^M \to \mathbb{R}^N$ 的函数，$\frac{\partial h}{\partial \theta}$ 仍表示 Dh，而非 $\nabla_\theta h$

$\nabla_\theta h = \left(\frac{\partial h}{\partial \theta}\right)^T$，和正常一样

这导致 chain rule 变为

$$\underbrace{\frac{\partial L}{\partial \theta}(\alpha)}_{\text{gradient}} = \underbrace{\frac{\partial h}{\partial \theta}(\alpha)}_{\text{derivative}}{}^{T} \underbrace{\frac{\partial L}{\partial h}(h(\alpha))}_{\text{gradient}}$$

更加 gross 的是，对于 $\mathbb{R}^M \to \mathbb{R}^N$ 的函数，教材里有时会用 $\nabla_\theta h$ 来表示 Jacobian.

即对于 $\mathbb{R}^M \to \mathbb{R}$ 的函数，$\frac{\partial L}{\partial \theta}$ 和 $\nabla_\theta L$ 都表示 gradient

对于 $\mathbb{R}^M \to \mathbb{R}^N$ 的函数，$\frac{\partial h}{\partial \theta}$ 和 $\nabla_\theta h$ 都表示 derivative.

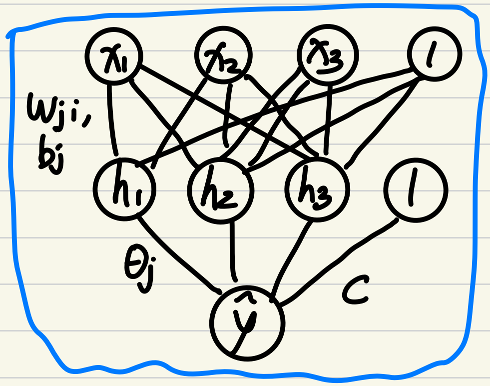
ex 3-layers NN (i.e. 1 hidden layer)

input : $x \in \mathbb{R}^3$

output : $\hat{y} \in \mathbb{R}$

loss : $L(y;\hat{y}) = (\hat{y}-y)^2$

$$x$$
$$\downarrow$$
$$h_j = f\left(\sum_i w_{ji} x_i + b_j\right)$$
$$\downarrow$$
$$\hat{y} = \sum_j \theta_j h_j + c$$



Sol $\dfrac{\partial L}{\partial \hat{y}} = 2(\hat{y}-y)$

$$\frac{\partial L}{\partial \theta_j} = \frac{\partial \hat{y}}{\partial \theta_j}\frac{\partial L}{\partial \hat{y}} = h_j \frac{\partial L}{\partial \hat{y}}$$

$$\frac{\partial L}{\partial h_j} = \frac{\partial \hat{y}}{\partial h_j}\frac{\partial L}{\partial \hat{y}} = \theta_j \frac{\partial L}{\partial \hat{y}}$$

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial h_j}{\partial w_{ji}}\frac{\partial L}{\partial h_j} = f'\left(\sum_i w_{ji} x_i + b_j\right) x_i \frac{\partial L}{\partial h_j}$$

$$= x_i f'(\ldots)\, \theta_j \frac{\partial L}{\partial \hat{y}}$$

vectorization form: $y, \hat{y} \in \mathbb{R}^N$

input: $X \in \mathbb{R}^{N \times d}$ (N sample, each of dim d)

$W \in \mathbb{R}^{M \times d}$ (for each dim, get M weights for $h_1, \dots, h_M$)

hidden: $H \in \mathbb{R}^{N \times M}$ (M hidden neurons for each sample $x^{(1)}, \dots, x^{(N)}$)

$\theta \in \mathbb{R}^M$, $c \in \mathbb{R}$

output: $\hat{y} \in \mathbb{R}^N$

<u>Sol</u> we have $H = f(XW^T + b)$ (f elementwise)

$$\hat{y} = H\theta + c$$

$$L = (\hat{y} - y)^T (\hat{y} - y)$$

$$\Rightarrow \nabla_{\hat{y}} L = 2(\hat{y} - y) \in \mathbb{R}^N$$

$$\nabla_\theta L = \underbrace{\left(\frac{\partial \hat{y}}{\partial \theta}\right)^T}_{= H^T \text{ since } D_\theta \hat{y} = H} \nabla_{\hat{y}} L = H^T \nabla_{\hat{y}} L = 2H^T(\hat{y} - y)$$

$$\nabla_c L = \sum_i \frac{\partial L}{\partial \hat{y}_i} = \mathbb{1}_N \nabla_{\hat{y}} L = 2 \mathbb{1}_N (\hat{y} - y)$$

Since $\frac{\partial L}{\partial H_{ij}} = \theta_j \frac{\partial L}{\partial y^{(i)}} \Rightarrow \nabla_H L = \left[\theta_1 \frac{\partial L}{\partial y} \cdots \theta_M \frac{\partial L}{\partial y}\right]$

$$= \frac{\partial L}{\partial y} \theta^T = 2(\hat{y} - y)\theta^T$$

...