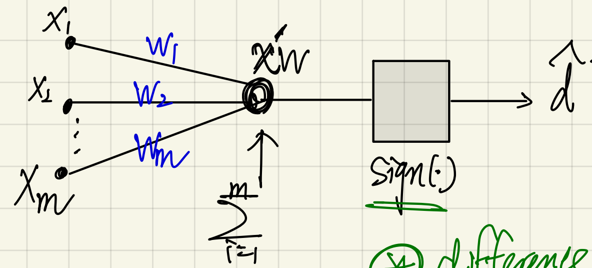
§ Neural Network.

(*) Multiple layers



(*) difference
switch to other $\sigma(\cdot)$

- "Neurons" generalized a linear classifier.

$$\begin{cases} \text{linear classifier: } \hat{d} = \text{sign}(x'w) \\ \text{Neuron: } \hat{d} = \sigma(x'w) \end{cases}$$

non-linear $\sigma(\cdot)$ function

$\Rightarrow$ popular choices of $\sigma(\cdot)$:

1. ReLU: $\sigma(z) = \max\{0, z\}$
2. Logistic: $\sigma(z) = \frac{1}{1+e^{-z}} = (1+e^{-z})^{-1}$
3. sign: $\sigma(z) = \text{sign}(z)$

1. 2. 3.

- A "neural network": 3 layer framework. $\begin{cases} \text{Need to decide \# of I/H/O layers.} \\ \text{Choosing right weights (nonconvex opt.)} \end{cases}$

$$\boxed{\begin{array}{ccccc} \text{Input layer} & \longrightarrow & \text{hidden layers} & \longrightarrow & \text{Output layer.} \\ (x) & & (h) \ \text{nodes} & & (\hat{d}) \end{array}}$$

$h_\ell = \sigma\left(\sum_{j=1}^{M} w_{\ell j} x_{\ell j}\right)$

$\hat{d}_m = \sigma\left(\sum_{\ell=1}^{P} v_{m\ell} h_\ell\right) = \sigma\left(\sum_{\ell=1}^{P} v_{m\ell}\, \sigma\left(\sum_{j=1}^{M} w_{\ell j} x_{\ell j}\right)\right)$

(*) Universal Approx. Thm (1991): given enough hidden nodes & correct weights.

* Multiple outputs to solve multiple problems
* "Deep learning" refers to "hidden layers" (how many we have?)
* Input is the "shallowest" layer; Output is the "deepest" layer
* Use SGD & Backpropagation

- Backpropagation: work back from deep (output) to shallow (input) layer.

$\Rightarrow$ for N training (input) samples & Q outputs : $\min\limits_{W_{ej}, V_{ej}} \underbrace{\sum_{i=1}^{N} \sum_{q=1}^{Q}}_{\text{backward}} \frac{1}{2}(\hat{d}_{iq} - d_{iq})^2$

1°. Initialization: guess W_{ej}^0, V_{ej}^0

2° SGD process: choose inputs i randomly.

3° Calculate $h_{i,p}$, $\hat{d}_{i,q}$ for data i.

4° GD updates: first V_{ej}, then W_{ej}. (deep back to shallow)

§. Backpropagation for NN. (BPPG)

- BPPG. uses SGD to train weights (W & V)

$\hat{d}_q = \sigma\left(\sum_{i=1}^{P} V_{qn} h_n\right)$ where $h_n = \sigma\left(\sum_{\ell=1}^{M} W_{n\ell} x_\ell\right)$

We take $\sigma(z) = (1+e^{-z})^{-1} = \frac{1}{1+e^{-z}}$ ≪Activation function≫

N training samples (inputs) $(x_1^i, x_2^i \cdots x_M^i ; d^i)$ $i=1,\cdots,M$

1° Initialization: Set $W_{n\ell}$, V_{qn}

2° For $t=1,2,3,\cdots$ Select $i_t \in \{1,\cdots,M\}$ randomly

Compute $h_j^{i_t}$, $\hat{d}_k^{i_t}$.

$\Rightarrow$ Calculate: $V_{k\ell}^{(t+1)} = V_{k\ell}^{(t)} - \alpha_t \left(\frac{df^{i_t}}{V_{k\ell}}\right)$

$W_{mj}^{(t+1)} = W_{mj}^{(t)} - \alpha_t \left(\frac{df^{i_t}}{dW_{mj}}\right)$

where α_t is step size.

f^{i_t}: loss function.

- Gradients from Chain Rule.

Consider "sq error loss": $f(v,w) = \sum_{i=1}^{N} \underbrace{\left(\frac{1}{2}\sum_{q=1}^{Q}(\hat{d}_q^i - d_q^i)^2\right)}_{f^i(v,w)}$

$$* \quad \frac{\partial f^i}{\partial V_{k,\ell}} = \left(\frac{\partial f^i}{\partial \hat{d}_k^i}\right)\frac{\partial \hat{d}_k^i}{\partial V_{k,\ell}}$$

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

$\Rightarrow \sigma'(z) = \sigma(z)(1-\sigma(z))$

$$① \quad \boxed{\frac{\partial f^i}{\partial V_{k\ell}}} = \left(\underbrace{(\hat{d}_k^i - d_k^i)}_{\text{relevant term.}} \cdot \underbrace{\sigma'\left(\sum_{n=1}^{P} h_n^i V_{kn}\right)}_{\text{chain rule.}}\right) \cdot h_\ell^i$$

$(\partial \hat{d}_k^i / \partial V_{k,\ell})$

(=0 when correct label)

$d = \sigma(\cdot)$

$$= \underbrace{(\hat{d}_k^i - d_k^i)\hat{d}_k^i(1-\hat{d}_k^i)}_{\equiv \delta_k^i} h_\ell^i \equiv \boxed{\delta_k^i h_\ell^i}$$

where $\begin{cases} \delta_k^i : \text{error related to output.} \\ h_\ell^i : \text{layer input.} \end{cases}$

<update>

$$\Rightarrow \boxed{V_{k\ell}^{(t+1)} = V_{k\ell}^{(t)} - \alpha_t \delta_k^i h_\ell^i}$$

② for W_{mj}.

$$\frac{\partial f^i}{\partial W_{mj}} = \sum_{q=1}^{Q}\left(\frac{\partial f^i}{\partial \hat{d}_q^i}\right)\left(\frac{\partial \hat{d}_q^i}{\partial h_m^i}\right)\left(\frac{\partial h_m^i}{\partial W_{mj}}\right).$$

loss/output output/hidden hidden/input.

(i) loss/output: $\frac{\partial f^i}{\partial \hat{d}_q^i} = \hat{d}_q^i - d_q^i$.

(ii) output/hidden: $\frac{\partial \hat{d}_q^i}{\partial h_m^i} = \frac{\partial}{\partial h_m^i}\sigma\left(\sum_{n=1}^{P} h_n^i V_{qn}^i\right) = V_{qm}^i \cdot \sigma'\left(\sum_{n=1}^{P} h_n^i V_{qn}^i\right)$

$$= V_{qm}^i \cdot \left[\hat{d}_q^i(1-\hat{d}_q^i)\right].$$

(iii) hidden/input: $\frac{\partial h_m^i}{\partial W_{mj}} = \frac{\partial}{\partial W_{mj}}\sigma\left(\sum_{\ell=1}^{M} W_{m,\ell} x_\ell^i\right) = W_{mj} \cdot \sigma'\left(\sum_{\ell=1}^{M} W_{m\ell} x_\ell^i\right)$

$$= x_j^i \cdot \left[h_m^i(1-h_m^i)\right]$$

$\Rightarrow$ So, $\frac{\partial f^i}{\partial W_{mj}} = \sum_{q=1}^{Q} \underbrace{\left[(\hat{d}_q^i - d_q^i) V_{qm}^i \cdot \hat{d}_q^i(1-\hat{d}_q^i) \cdot h_m^i(1-h_m^i)\right]}_{\equiv \gamma_m^i = \delta_q^i V_{qm}^i h_m^i(1-h_m^i)} x_j^i$

$$\equiv \boxed{\gamma_m^i x_j^i} \quad \text{(another weighted sum of } x_j^i\text{)}$$

$\Rightarrow$ ⟨Update⟩ $\boxed{W_{mj}^{(t+1)} = W_{mj}^{(t)} - \alpha_t \underline{\gamma_m^i x_j^i}}$.

⊛ How to "Backpropagation"?

1° Initialize W_{mj}^{0}, V_{kl}^{0}.

2° Iteration; for $t = 0, 1, 2, \cdots$

<SGD> Choose $i_t \in \{1, \cdots, N\}$. randomly.

<Forward thru NN> Compute $h_m^{i_t}$ by $(X_j^{i_t}, W_{mj}^{t})$

$\hat{d}_q^{i_t}$ by $(h_m^{i_t}, V_{kl}^{t})$

<Backward updates> $\delta_k^{i_t} = (\hat{d}_k^{i_t} - d_k^{i_t})\, \hat{d}_k^{i_t} (1 - \hat{d}_k^{i_t})$

$\Rightarrow V_{kl}^{(t+1)} = V_{kl}^{(t)} - \alpha_t \delta_k^{i_t} h_l^{i_t}$

$\Rightarrow \gamma_m^{i_t} = \sum_{q=1}^{Q} \delta_q^{i_t} V_{qm}^{t} h_m^{i_t} (1 - h_m^{i_t})$

$\Rightarrow W_{mj}^{(t+1)} = W_{mj}^{(t)} - \alpha_t \gamma_m^{i_t} x_j^{i_t}$

⊛ Cost/Loss func is non convex function of weights!

* May converge to local minima.
* Some empirical tricks: "Batch SGD", "Normalization"
* Can add regularization.