Perceptron: $y = f\left(\sum_i x_i w_i + \theta \right)$ $\quad \to\ t \in \{0,1\}$

$$w(t+1) = w(t) + \alpha \cdot \left(x_i \cdot (d_i - t_i) \right)$$

Convergence:

$$\cos \rho = \frac{\vec{w}^* \cdot \vec{w}_{t+1}}{\|\vec{w}_{t+1}\|} \longrightarrow \geq \vec{w}^* \cdot \vec{w}_0 + (t+1)\delta,$$

$$\delta = \min\{ \vec{w}^* \cdot \vec{p} \,;\, \forall p \in P' \}$$

$$\|w_{t+1}\|^2 \leq \|w_0\|^2 + (t+1) \quad \to \text{protože } p' \text{ je normalizované}$$

$\vdots$

$$\cos \rho \geq \frac{\vec{w}^* \cdot \vec{w}_0 + (t+1)\delta}{\sqrt{\|w_0\|^2 + (t+1)}}$$

– jelikož $\cos x \leq 1 \ \forall x$ a $\delta > 0 \implies$ roste mi to s $\sqrt{t}$, tudíž musím po konečném množství krocích narazit na mezní hodnotu t. $\boxed{X}$

BP – network:

Minimizing: $E = \frac{1}{2} \sum_p \sum_j \left(y_{jp} - t_{jp} \right)^2$

derivace sigmoidy $\sim f(x)$

$$sigm'(\xi_j) = \lambda y_j \cdot (1 - y_j)$$

update rule: $w_{ij}(t+1) = w_{ij}(t) + \Delta_E w_{ij}(t)$

$$\Delta_E w_{ij}(t) = -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial \xi_j}{\partial w_{ij}} \quad \to \text{chain-rule}$$

$\to$ momentum techniques

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j y_i + \alpha_m \left(w_{ij}(t) - w_{ij}(t-1) \right)$$

this is output from hidden layer

$$\to \Delta_E w_{ij} = -\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial}{\partial w_{ij}} \cdot \sum_k w_{kj} y_k = -\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot y_i = -\frac{\partial E}{\partial y_j} \cdot f'(\xi_j) \cdot y_i = -(y_j - d_j) \cdot f'(\xi_j) \cdot y_i$$

$$f'(\xi_j) \qquad y_i$$
$$\parallel \qquad \parallel$$

$$\delta_j \begin{cases} \text{output: } (d_j - y_j) \cdot \lambda y_j \cdot (1 - y_j) \\ \text{hidden: } \left(\sum_k \delta_k w_{jk} \right) \lambda y_j (1 - y_j) \end{cases}$$

$$\to \Delta_E w_{ij} = -\frac{\partial E}{\partial w_{ij}} = -\left(\sum_k \frac{\partial E}{\partial \xi_k} \cdot \frac{\partial \xi_k}{\partial y_j} \right) \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial \xi_j}{\partial w_{ij}} = -\left(\sum_k \frac{\partial E}{\partial \xi_k} \cdot \frac{\partial}{\partial y_j} \cdot \sum_{j'} w_{j'k} y_{j'} \right) \cdot \ldots$$

$$= -\left(\sum_k \frac{\partial E}{\partial \xi_k} \cdot w_{jk} \right) \cdot f'(\xi_j) \cdot y_i = -\left(\sum_k \delta_k w_{jk} \right) \cdot f'(\xi_j) \cdot y_i$$

$$y_j = f(\xi_j) = \frac{1}{1 + e^{-\lambda \xi_j}}, \text{ where } \xi_j = \sum_i w_{ij} \cdot y_i$$

$\longrightarrow$ that all serves as an input to another layer...

AdaGrad:

adaptive weight update:

$$A_i(t+1) = A_i(t) + \left(\frac{\partial E}{\partial w_i}\right)^2$$

$$w_i(t+1) = w_i(t) - \frac{\alpha}{\sqrt{A_i}} \cdot \frac{\partial E}{\partial w_i}$$

tzn. že si lépe pamatuje, kde hodně chyboval a podle toho upravuje váhy

Second-order-algs:

$$E(\vec{w}+\vec{h}) \approx E(\vec{w}) + \nabla E(\vec{w})^T \vec{h} + \frac{1}{2} h^T \nabla^2 E(\vec{w}) \vec{h}$$

derivace podle h: $$\frac{\partial E(\vec{w}+\vec{h})}{\partial \vec{h}} \approx \nabla E(\vec{w}) + \vec{h} \nabla^2 E(\vec{w})$$

to chci nulový, takže: $$\vec{h} = -\underline{\left(\nabla^2 E(\vec{w})\right)^{-1}} \cdot \nabla E(\vec{w})$$

tohle může být náročný získat

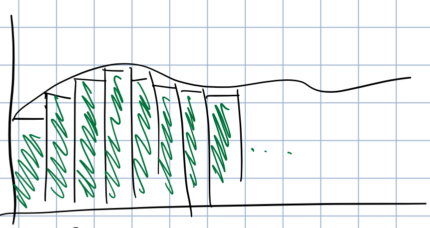
pseudo-newton: $$w_i^{(k+1)} = w_i^{(k)} - \frac{\nabla_i E(\vec{w})}{\frac{\partial^2 E(\vec{w})}{\partial w_i^2}}$$

uvažuju pouze diagonály

Quickprop: $$w_i^{(k+1)} = w_i^{(k)} + \underline{\Delta^{(k)} w_i} \quad - \frac{\nabla_i E^{(k)}}{\frac{\nabla_i E^{(k)} - \nabla_i E^{(k-1)}}{\Delta^{(k-1)} w_i}}$$

aproximace $\frac{\partial^2 E(w_i)}{\partial w_i^2}$

$$\int_0^1 \left| f(x) - f'(x) \right| dx < \varepsilon$$



[0,1] rozdělím na N

$$\varphi_N(x) = \min\{ f(x') : x' \in [x_i; x_{i+1}],\ x_i \le x < x_{i+1} \}$$

$$E_N = \int_0^1 \left| f(x) - \varphi_N(x) \right| dx$$ protože $f(x) \ge \varphi_N(x) \Rightarrow E_N = \int_0^1 f(x)\,dx - \int_0^1 \varphi_N(x)\,dx$

Asociativní paměti:

$\vec{e} = \vec{x} \cdot W$, hledáme takové W, že $XW = Y$ $\rightarrow$ takže chci něco jako $W = X^{-1} \cdot Y$

Se zpětnou vazbou:

Hledám fixed point, že $\xi_j = \xi_j \cdot W$ $\rightarrow$ definice stabilního stavu.

Eigenvalue Automata

pokud mám v matici n-nezávislých vlastních vektorů, pak platí:

$$\vec{x}^i \cdot W = \lambda_i \vec{x}^i$$

$\Rightarrow$ pokud je B.Ú.N.O. λ_n největší, tak se po konečné mnoho iteracích stane $\vec{x}^i$ atraktorem.

Asociativní učení

- přesně tyhle atraktory si chci zapamatovat.

Hebbian learning