Neuron

- dendrites (inputs)
- axon (single output)
- synapses (branching)
- $8{,}6 \times 10^{10}$ neurons ($8 \times 10^4 / mm^3$)
- short-memory = signal circulation
- long-term memory = weights updating

Standardization

- interval scaling $\begin{cases} [-1,1] \\ [0,1] \end{cases}$

$$V_{ij} = \sum_k (x_{ki} - \mu_i) \cdot (x_{kj} - \mu_j)$$

- decimal scaling: dělím nejmenším násobkem 10, aby $\forall x : x \in [-1,1]$
- MAD-scale: „mean absolute deviation" $\frac{1}{n}\sum |x - \bar{x}|$
- MSD-scale: „mean standard deviation" $\sqrt{\frac{\sum (x-\bar{x})^2}{n-1}}$

$$= \frac{x - \text{mean}}{\text{deviation}}$$

ML Transform ~ PCA

$X^m \to X^p : m \gg p$

$$X = \begin{pmatrix} \cdots & x_{1j} & \cdots \\ & \vdots & \\ \cdots & x_{uj} & \cdots \end{pmatrix}$$

Vždy itemji přes sloupce, protože se snažím zjistit, které vyhodit.

$$\mu_j = \frac{1}{u} \sum_i^u x_{ij} \qquad \mu = (\cdots \mu_j \cdots)$$

Kovarianční matice: $w_{ij} = w_{ji} = \frac{1}{u} \sum_k^u (x_{ki} - \mu_i) \cdot (x_{kj} - \mu_j)$

nejvýznamnější vlastní čísla a jejich vlastní vektory odpovídají nejvýznamnějším směrům

redukované vektory $y = V^T x$

Normální rozdělení:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$E(x) = \mu$, rozptyl σ^2

Hypoteses testing

$$Error_S(h) = \underset{x \in S}{P}\left[f(x) \neq h(x)\right] = \frac{r}{|S|}$$

(r … mismatched)

- pst, že jsem se netrefil

When having binomial distr:

95% conf. int:

$$error_S(h) \pm 1{,}96 \sqrt{\frac{err_S(h) \cdot (1 - err_S(h))}{n}}$$

K-Fold validation:

- pro jednotlivé podmnožiny počítám error dvou odlišných hypotéz

$$\delta_i = err_{T_i}(A) - err_{T_i}(B)$$

$$\bar{\delta} = \frac{1}{k} \sum_i \delta_i$$

$$\text{conf. int: } \bar{\delta} \pm t_{N,k-1} \cdot \frac{\sqrt{\frac{1}{k-1} \sum_i (\delta_i - \bar{\delta})^2}}{\sqrt{k}}$$

(↓ estimated std. dev.)

↑ Must be on identical test sets.

Can compare two hypos:

$$\hat{d} = err_{S_1}(h_1) - err_{S_2}(h_2)$$

$$\text{conf. int: } d \pm z_N \sqrt{\frac{err_{S_1}(h_1) \cdot (1 - err_{S_1}(h_1))}{n_1} + \text{—||—}}$$

(—||— pro 2)

↑ Requires independent testsets. ↑

Perceptron

$$x^T w - \gamma \geq 0 \implies y = 1$$

$$x^T w - \gamma < 0 \implies y = 0$$

neuron:
- active: $\sigma(w^T x - \gamma) = 1$
- silent: $\sigma(w^T x - \gamma) = \frac{1}{2}$
- passive: $\sigma(w^T x - \gamma) = 0$

$$w' = (w^{**}, -\gamma)$$
$$x = (x^{**}, 1)$$
$$\longrightarrow y = 1 \iff x^T w \geq 0$$

Lineární separabilita:

→ pokud jde perceptronem všechny rozdělit na A, B.

Pokud $t_i = 0$, ale $y_i = 1$: $w_i = w_i - \alpha x_i$

$t_i = 1$, ale $y_i = 0$: $w_i = w_i + \alpha x_i$

MLP

error function:

$$\frac{1}{2}\sum_p \sum_j (y_{j,p} - d_{j,p})^2$$

adjustment:

$$w_{ij} = w_{ij} + \alpha \Delta_E w_{ij}$$

→ může být obohaceno o moment $\alpha_m(E(w(t)) - E(w(t+1)))$

$= y_i \cdot \delta_j$

δ_j: $(d_j - y_j) \cdot y_j \cdot (1 - y_j)$ — output neuron

δ_j: $\left(\sum_k \delta_k w_{jk}\right) \cdot y_j \cdot (1 - y_j)$ — hidden neuron

Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

softmax $\sigma(x)_j = \frac{e^{-x_j}}{\sum_i e^{-x_i}}$

$$\frac{\partial y_j}{\partial x_j} = y_j(1 - y_j)$$

$$\frac{\partial y_j}{\partial x_k} = -y_j \cdot y_k$$

Ideal initial weights:

$$\left[-\frac{3}{\sqrt{N}} \cdot A_i ; +\frac{3}{\sqrt{N}} A\right]$$

N… number of examples

Learning rate decay:

- exp: $\alpha_i = e^{-kt}$
- inverse: $\frac{\alpha_0}{1+kt}$

Momentum – learning:

$$w(i+1) = w(i) - \alpha \frac{\partial E}{\partial w(i)} + \alpha_m (w(i) - w(i-1))$$

keeping the previous direction — to not oscilate too much

Nestrov momentum:

$$w(i+1) = w(i) - \alpha \frac{\partial E(w + \alpha_m \Delta w(i))}{\partial w} + \alpha_m \Delta w(i)$$

($\Delta w_i(i+1)$)

where would I get using the weights from previous iteration

Silva & Almeida:

minimizing $c_i^2 w_i^2 + k_1 w_1 + k_2$

accelerate / decelerate

nothing changed — derivatives changed

$$\alpha_i^{(k+1)} = \begin{cases} \alpha_i^{(k)} u & \text{pokud } \nabla E_i^{(k)} > 0 \\ \alpha_i^{(k)} d & \text{pokud } \nabla E_i^{(k)} < 0 \end{cases}$$

$d < 1 < u$

Delta-Bar-Delta

$$\alpha_i^{(k+1)} = \begin{cases} \alpha_i^k + u & : \nabla_i E^k \cdot \bar{\delta}^{(k-1)} > 0 \\ \alpha_i^k \cdot d & : \nabla_i E^k \cdot \bar{\delta}^{(k-1)} > 0 \\ \alpha_i^k & \text{else} \end{cases}$$

where $\bar{\delta}_i^k$:

$$\bar{\delta}_i^k = (1-\phi)\nabla_i E^k + \phi \bar{\delta}_i^{k-1}$$

(Super) SAB

let: α^+ increasing $(=1{,}05)$
α^- decreasing $(=2)$
α_{START} initial $(=1{,}2)$
α_m momentum $(=0{,}3)$

If error sign didn't change, increase: $\alpha_{ij}(t+1) = \alpha^+ \cdot \alpha_{ij}(t)$

else: $\alpha_{ij}(t+1) = \alpha_{ij}(t) / \alpha^-$

$\Delta w_{ij}(t+1) = 0$

RMSProp

Alternativa AdaGradu:

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

← tohle dělá stejné

AdaGrad:

$$A_i = \frac{\partial E}{\partial w_i}$$ → vlastně mi drží RMS složku ∂E

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

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

Adam

$$N_i(t+1) = \beta_1 N_i(t) + (1-\beta_1)\cdot \frac{\partial E}{\partial w_i}$$ → first order

$$A_i(t+1) \; \beta_2 N_i(t) + (1-\beta_2)\left(\frac{\partial E}{\partial w_i}\right)^2$$ → second order

$$\hat{N}_i(t+1) = N_i(t+1) / (1-\beta_1^{(t+1)})$$

$$\hat{A}_i(t+1) = A_i(t+1) / (1-\beta_2^{(t+1)})$$

$$w_i(t+1) = w_i(t) - \alpha \frac{\hat{N}_i(t+1)}{\sqrt{\hat{A}(t+1)} + \varepsilon}$$

- scale invariant

Second-order-algorithm

$$\nabla E(w) = \begin{pmatrix} \vdots \\ \frac{\partial E(w)}{\partial w_i} \\ \vdots \end{pmatrix} \quad \nabla^2 E(w) = \begin{pmatrix} \vdots & & \\ \cdots & \frac{\partial E(w)}{\partial w_i \partial w_j} & \cdots \\ \vdots & & \end{pmatrix}$$

$$w(k+1) = w(k) - \left(\nabla^2 E(w)\right)^{-1} \cdot \nabla E(w)$$

Quickprop

$$w_i(k+1) = w_i(k) + \nabla_k w_i$$

$$\nabla_k w_i = \nabla_{k-1} w_i \cdot \frac{\nabla_i E(k)}{\nabla_i E(k) - \nabla_i E(k-1)}$$

$$= \frac{\nabla_i E(k)}{\frac{\nabla_i E(k) - \nabla_i E(k-1)}{\nabla_{k-1} w_i}}$$

tohle je odhad $\frac{\partial^2 E(w)}{\partial w^2}$

Pseudo-Newton method

- only diagonal is computed

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

Weight perturbation

$$\nabla w_i = -\alpha \frac{E(w') - E(w)}{\beta}$$

Maxout activation

$$\max\{x_i^T w_1 \; ; \; x_i^T w_2\}$$

Batch normalization

- upravuju vždy průměrnou hodnotu více prvků.

$$\alpha_i = \gamma_i \cdot \overline{x} + \beta_i$$

($\overline{x}$ → mean of the lin. output)

Space partitioning

$R(m,n) :=$ # of regions defined by m hyperplanes in n-dimensional space.

$$R(1,n) = 2, \quad R(0,m) = 0 \; : \; n,m \geq 1$$

$$R(m,n) = R(m-1,n) + R(m-1,n-1)$$

$$= 2 \cdot \sum_{i=0}^{n-1} \binom{m-1}{i}$$

Minimal number of patterns

$$P \geq \left(\frac{W}{\varepsilon}\right) \log\left(\frac{N}{\varepsilon}\right)$$

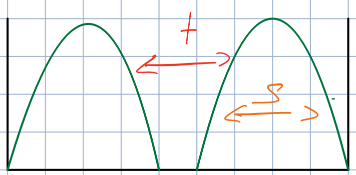
N ... neurons
W ... weight
P ... patterns

The optimal error fc enforcing condensed features

$$F = y^s (1-y)^s \cdot (y-0{,}5)^t$$

čím vyšší t, tím větší díra uprostřed

čím vyšší s, tím větší rozkrok kopců



L^2-reg.

$$L = \sum_{(y,t)} (t-y)^2 + \lambda \cdot \sum_i w_i^2$$

$$w_i = w_i - \alpha \lambda s_i - \alpha \frac{\partial L}{\partial w_i}$$

$$s_i = \begin{cases} -1 & w_i < 0 \\ +1 & w_i > 0 \end{cases}$$

$\forall n$: #treshold fun. $\leq$ # regions $\leq$ # boolean func

$<$ → if the dimension of vector is too big, we don't have suitable treshold func. for good approx.

VC-dimension

„kolik nejvíce máme těmi fcemi rozlišit výstupů"

Definition:

Let $C = \{f_i\}$ be a set of functions (concept class).
The set of m training patterns $\{t_k\}_{k=1,\dots,m}$ *can be shattered* by means of C, if for each of the 2^m possible labelings of these patterns with $1/0$, there exists at least one function, that satisfies this labeling.

Definition: → kolik nejvíc těmi funkcemi můžu rozlišit.

The *VC-dimension* V *of a set of functions* C is defined as the biggest m, for which a set of m training patterns exists that can be shattered.

Pokud množina C obsahuje funkce pro lib. m, je takový problém nenaučitelný.

Pruning BP

reduced layer:

- no uniform neuron $\forall x$
- no identical neuron repr. $\forall x$
- no inverse neuron repr. $\forall x$

→ this is reduced. → there is one for each BP network