BP
- jakou má derivaci sigmoid?
- jak vypadá update rule?

First-order-methods
- update rule AdaGradu

Second-order-methods
- how does look discret step in quickprop?

PCA:

mám n dimenzí, chci pouze k.

Hledám tedy n-k nejméně důležitých, nebo naopak k nejdůležitějších dimenzí.

chci docílit nejmenší kvadratické odchylky $\varepsilon_k^2 = \| x_k - y_k \|^2$, kde $y_k = \sum_{i=1}^{p} c_{ki} e_i$

ortonormální vektory v X^p

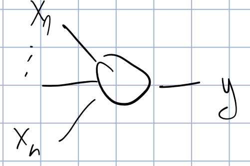
tohle je definice toho zobrazení do menšího prostoru

Získám tak, že si vytvořím kovarianční matici V.

$$v_{ij} = v_{ji} = \frac{1}{K} \sum_{k=1}^{K} (x_{ki} - \mu_i) \cdot (x_{kj} - \mu_j) \quad , \quad \mu_i = \sum_{k=1}^{K} x_{kj} / K$$

Nejdůležitější dimenze odpovídají největším vlastním číslům a jejich vektorům v kovarianční matici.

Perceptron:



$$\xi = \sum_i x_i w_i + \gamma$$

bias

$$y = \sigma(x^T w)$$ → extended vectors

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

A,B jsou lin. sep., pokud existuje dělící nadrovina, neboli

$$\exists w_1 \_ w_n, \gamma \quad \forall (x_1 \_ x_n) \in A \ \sum w_i x_i \geq \gamma \ \text{a}$$
$$\forall (y_1 \_ y_n) \in B \ \sum w_i y_i < \gamma$$

algoritmus:

```
$w(0)$ — inicializuj náhodně
while true:
    $y(t) = sgn(x^T w)$
    $w(t+1) = w(t) + \alpha \cdot x(t) \cdot (target - y(t))$
```

konvergence: - pokud jsou množiny separabilní pak dostanu váhy dostatečně blízko řešení w^*.

uvažme, že je něco špatně klasifikováno:

$$\cos\rho \ \frac{w^* \cdot w_{t+1}}{\|w_{t+1}\|} \longrightarrow w^* \cdot w_{t+1} \geq w^* \cdot w_0 + (t+1)\delta$$

$\nearrow \min \{ w^* \cdot \vec{p} ; \forall \vec{p} \in P \}$

$\searrow \|w_{t+1}\|^2 \leq \|w_0\|^2 + (t+1)$

$w_i \cdot p_i \leq 0$ - protože misklas.

$\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$

protože p_i je normal.

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

$\cos\rho \leq 1, \ \delta > 0$

pravá část roste s t, tudíž musí existovat maximum po konečně mnoha krocích.

PCA:

destruktivní zobrazení $X^n \to X^p$, kde $p < n$

hledání takové zobrazení, kde minimalizuje čtvercovou chybu: $\varepsilon_k^2 = \|x_k - y_k\|^2$

$\nearrow y_k = \sum_{i=1}^{p} c_{ki} \cdot e_i$

$\downarrow$ normalizovaný vektor z X^p

lze hledat pomocí kovariační matice a vlastních vektorů:

$$v_{ij} = v_{ji} = \frac{1}{n} \sum_{k=1}^{n} (x_{ki} - \mu_i) \cdot (x_{kj} - \mu_j), \text{ kde } \mu_j = \frac{1}{n} \sum_{k=1}^{n} x_{kj}$$

$$X = \begin{pmatrix} & \vdots & \\ — & x_{ij} & — \\ & \vdots & \end{pmatrix}$$

$$\mu = \begin{pmatrix} — & \mu_i & — \end{pmatrix}$$

LVQ:

prototypování a posouvání od nebo k sobě podle toho, jestli je shodná třída.

$\forall x$:

$$q = \underset{p}{\arg\min} \underbrace{\sqrt{\sum_i (x_i - w_i(p))^2}}$$

$\nearrow$ nejmenší vzdálenost od vstupního vektoru

$\to$ můžu použít lib. dist. fci.

$C_{w_q} = C_{x_i}$:

$$w_q \mathrel{+}= \mu(t) \cdot (x_i - w_q)$$

$C_{w_q} \neq C_{x_i}$:

$$w_q \mathrel{-}= \mu(t) \cdot (x_i - w_q)$$

Perceptron

$y = \mathrm{sgn}(x^T w)$ – extended vector

$w(t+1) = w(t) + \alpha \cdot x \cdot (\mathrm{target} - y)$ → přibližuji / oddaluji se od daného bodu

separabilita: $\exists w_1 - w_n, \gamma : \forall x_1 - x_n \sum_i x_i w_i \geq \gamma \wedge$

$\forall y_1 - y_n \sum_i y_i w_i < \gamma$ pak jsou lin. separabilní.

$$\cos \rho = \frac{w^* \cdot w_{t+1}}{\|w_{t+1}\|} \qquad w^* \cdot w_{t+1} \geq w^* \cdot w_0 + (t+1)\delta \qquad \delta = \min\{w_0 \cdot p,\ p \in P'\}$$

$$\|w_{t+1}\|^2 \leq \|w_0\|^2 + (t+1)$$

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

→ $\cos \rho \leq 1$, takže musím po konečně mnoha krocích narazit na max, protože ten výraz pořád roste s t.

LVQ:

pracuji s prototypem, každý reprezentuje jeden cluster, třídu.

Začnu s náhodným přiřazením vah. Pak postupně upravuji podle vstupů.

Inferenci dělám tak, vrátím takovou třídu, jejíž prototyp je vstupu nejblíž.

$\forall x \in P$:

→ $d(x,y)$ je vzdálenostní fce, například třeba eukl. $d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$

$q = \arg\min_i d(x, w_i)$

if $C_{x_i} = C_q$ (stejná třída) → přiblížím třídu k bodu

$\Delta w_q = +\mu \cdot (x_i - w_q)$

if $C_{x_i} \neq C_q$ (jiná třída) → vzdálím třídu od bodu

$\Delta w_q = -\mu \cdot (x_i - w_q)$

$w_q(t+1) = w_q(t) + \Delta w_q$

BP

transfer function: $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ ... sigmoid

$y_j = \sigma\left(\sum_i \bar{y}_i w_{ij} + \gamma\right)$, kde $\bar{y}_i$ je výstup předchozí vrstvy (tedy to může být i vstup)

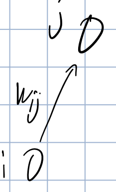
chci minimalizovat chybu na výstupní vrstvě: $E = \frac{1}{2}\sum_p \sum_j (y_{pj} - d_{pj})^2$

při tréninku musím chybu propagovat skrz celou síť na úpravu všech vah.

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

→ aplikace chain-rule

$$\Delta_E w_{ij} = -\frac{\partial E}{\partial w_{ij}} = \frac{\partial}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial \xi_j}{\partial w_{ij}}$$

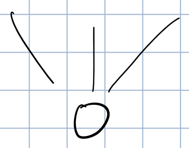


$$f'(x) = \lambda \cdot y_j \cdot (1 - y_j)$$

w_{ij} → váha mezi neuronem i a neuronem j

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

$$\delta_j \begin{cases} (d_j - y_j) \cdot \lambda \cdot y_j \cdot (1 - y_j) & \text{for output} \\ \left(\sum_h \delta_h \cdot w_{jh}\right) \cdot \lambda \cdot y_j \cdot (1 - y_j) & \text{for hidden layer} \end{cases}$$



for output: $-\frac{\partial E}{\partial w_{ij}} = \boxed{-\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j}} \cdot \frac{\partial \xi_j}{\partial w_{ij}} = \boxed{} \cdot \frac{\partial}{\partial w_{ij}} \cdot \sum_k w_{kj} \cdot y_k = \boxed{} \cdot y_i$

$$= -\frac{\partial E}{\partial y_j} \cdot f'(\xi_j) \cdot y_i = -(y_j - d_j) \cdot f'(\xi_j) \cdot y_i$$

for hidden: $-\frac{\partial E}{\partial w_{ij}} = \boxed{\boxed{-\left(\sum_h \frac{\partial E}{\partial \xi_h} \cdot \frac{\partial \xi_h}{\partial y_j}\right)} \cdot \frac{\partial y_j}{\partial \xi_j}} \cdot \frac{\partial \xi_j}{\partial w_{ij}} = \boxed{} \cdot y_i =$

$$\boxed{} \cdot \boxed{f'(\xi_j) \cdot y_i} = -\left(\sum_h \frac{\partial E}{\partial \xi_h} \cdot \frac{\partial}{\partial y_j} \cdot \sum_{h'} \underset{\overset{\|}{w_{jh}}}{w_{h'h}} \cdot y_{h'}\right) \cdot \boxed{} = -\left(\sum_h \delta_h \cdot w_{jh}\right) \cdot f'(\xi_j) \cdot y_i$$

First-order-methods

- chceme zrychlit trénink -> tzn. dělat takové úpravy vah, abych se rychleji dostal k optimu.

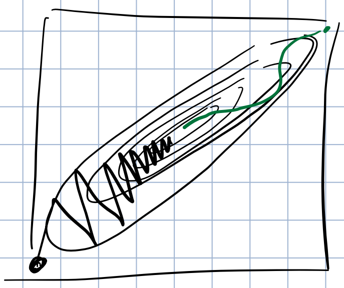
Momentum:

tohle je ten moment

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \Delta_E w(t) + \boxed{\alpha_m \left(w_{ij}(t) + w_{ij}(t-1) \right)}$$

držím si nějakou historii předchozího směru úpravy, abych byl konzistentní.

Musím citlivě nastavit, abych neignoroval změnu...



AdaGrade

při každém updatu si ukládám

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

-> tady si agreguju změnu pod druhou mocninou -> je to teda velmi citlivé na velké změny, zároveň to tu změnu dlouho udrží

pak

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

Second-order-algs

- principem je, že pouze informace, zdali jsem se zlepšil/zhoršil, chci vědět sklon té objective function -> abych vybral ten směr, který mi nejstrmější trajektorii -> to mi ideálně dostane nejrychleji k cíli.

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

$$\nabla E = \begin{pmatrix} \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_n} \end{pmatrix} \qquad \nabla^2 E = \begin{pmatrix} \frac{\partial E}{\partial w_1 \partial w_1} & \cdots & \\ \frac{\partial E}{\partial w_1 \partial w_2} & \ddots & \\ \vdots & & \ddots \end{pmatrix}$$

$$\nabla E(w+h) \text{ w.r.t. } h = \nabla E(w) + h^T \nabla^2 E(w)$$ -> mi hledáme, kde je 0

$$h = -\left(\nabla^2 E(w) \right)^{-1} \nabla E(w)$$ ---> jenže je problém, že jí nemusím najít inverz takové matice

Newton method

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

Pseudo-newton's method

- uvažujeme pouze prvky na diagonále, ostatní jsou 0.

$$w(t+1) = w(t) - \frac{\nabla E(w)}{\frac{\partial^2 E(w)}{\partial w^2}}$$

Quickprop:

$$w(t+1) = w(t) + \Delta w(t)$$

$$\Delta w(t) = - \frac{\nabla E(w)^{(t)}}{\frac{\nabla E(w)^{(t)} - \nabla E(w)^{(t-1)}}{\Delta w(t-1)}}$$

BP

$$E = \frac{1}{2} \sum_p \sum_j (d_{pj} - y_{pj})^2 \qquad f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \quad f'(\xi) = \lambda y_j \cdot (1 - y_j)$$

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

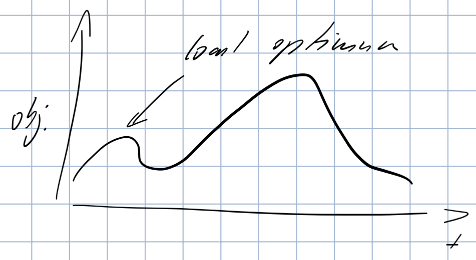
$$\nabla w_{ij}(t) = - \frac{\partial E}{\partial w_{ij}} = \overset{\text{chain-rule}}{\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial \xi_j}{\partial w_{ij}}}$$

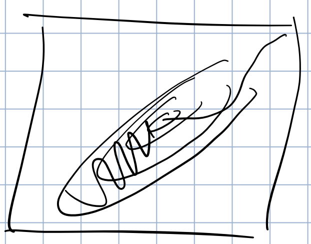
$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j y_i .$$

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

First-order-methods to speedup training

Momentum





$$\ldots + \alpha_m \left(w(t) - w(t-1) \right)$$

AdaGrade

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

a potom

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

Second-order-derivatives

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

$$\nabla E(w+h) = \nabla E(w) + h^T \nabla^2 E(w)$$

?

nastavím rovno nule

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

Newtons method:

$$w(t+1) = w(t) + \Delta w(t)$$

problémem je, že nemusím vždy najít inverz

Pseudo newton method:

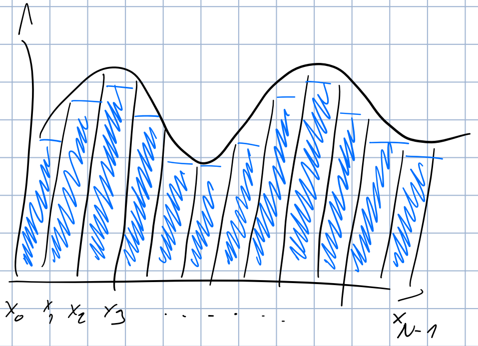
$$\frac{\partial^2 E}{\partial w_i^2}$$

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

quickprop má „secant steps", což jsou diskrétní aproximace

$$- \frac{\nabla_i E^{(t)}}{\frac{\nabla_i E^{(t)} - \nabla_i E^{(t-1)}}{\Delta w_i(t-1)}}$$

Real value approx:



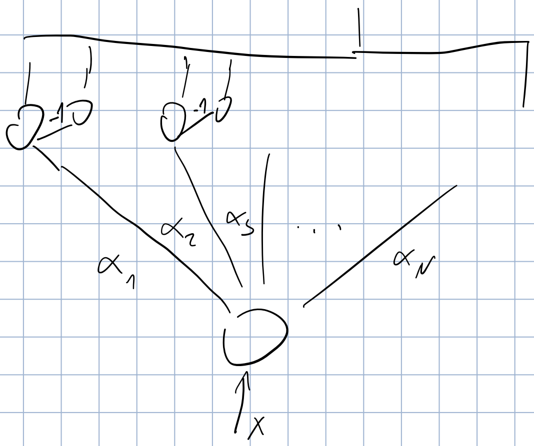
$$\varphi_N(x) = \min\{f(x') : x \in [x_i, x_{i+1}] \text{ for } x_i \le x \le x_{i+1}\},$$

$$E = \int_0^1 |f(x) - \tilde{f}(x)|\, dx < \varepsilon$$

$$\forall \varepsilon\ \exists n_0 : N \ge n_0 : \int_0^1 f(x)dx - \int_0^1 \varphi^N(x)dx < \varepsilon$$

$$\varepsilon = \int_0^1 f(x)\,dx - \int_0^1 \varphi^N(x)$$

pak mám vždy dvojici neuronů, které je aktivní, pokud $\overset{\text{neuron 1}}{x_i} \le x < \overset{\text{neuron 2}}{x_{i+1}}$



→ sum up

Asociativní paměť

$e = x \cdot W$ → bez zpětné vazby

recurrent:

hledám takový bod, aby $\xi = \xi \cdot W$ → tzn. že to je vlastní vektor vlastního čísla 1.

Eigenvalue automata:

$x^i \cdot W = \lambda_i \cdot x^i$ → pro vlastní vektory a jejich vlastní čísla

$a_t = \alpha_1 \cdot \lambda_1^t x_1 + \alpha_2 \lambda_2^t x_2 + \dots$

→ tak po dostatečně mnoha iteracích bude převládat jedno $\alpha_i \lambda_i x_i$

tudíž x_i budeme označovat jako atraktor

Hebbian learning

$$W = \sum_{i}^{m} W^{i} \qquad W^{i} = n \times k \text{ correlation matrix}$$

$$W^{\ell} = [x_i^{\ell} y_j^{\ell}]_{n \times k}$$

$$x^{p} \cdot W = x^{p} \cdot (W^{1} + W^{2} + \dots + W^{m})$$

$$= x^{p} W^{p} + \sum_{i \neq p} x^{p} W^{i}$$

$$= y^{p} x^{p} x^{p} + \overbrace{\sum_{i \neq p} y^{i} x^{i} x^{p}}^{\text{crosstalk}}$$

$$x^{p} x^{p} \geq 0 \Rightarrow \quad \text{sgn}(x \cdot W) = \text{sgn}\left(y^{p} + \sum_{i \neq p} \frac{x^{i} x^{p}}{x^{p} x^{p}} \right) \overset{\text{chci}}{=} y^{p}$$

to je pouze v případě,
že je crosstalk < 1

$$a_1 = \alpha_1 \lambda_1^{+} x_1 + \dots$$

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

$$W = W^{1} + W^{2} + \dots + W^{n}$$

$$W^{\ell} = [x_i^{\ell} y_j^{\ell}]_{n \times k} \longrightarrow \text{kovariační matice}$$

$$x^{p} \cdot W = x^{p} W^{p} + \sum_{i \neq p} x^{p} W^{i}$$

$$= y^{p} x^{p} x^{p} + \overbrace{\sum_{i \neq p} y^{i} x^{i} x^{p}}^{\text{crosstalk}}$$

$$\text{sgn}(x^{p} W) = \text{sgn}\left(y^{p} + \sum_{i \neq p} y^{i} \frac{x^{i} x^{p}}{x^{p} x^{p}} \right) = \text{pokud chci} = y^{p} \text{ tak crosstalk} < 1$$

$$m \approx 0{,}18n$$

BAM

$$y_0 = \mathrm{sgn}(x_0 W)$$

$$x_1 = \mathrm{sgn}(W y_0^T)$$

$$y_1 = \mathrm{sgn}(x_1 W)$$

$$x_2 = \mathrm{sgn}(W y_1^T)$$

Hebbian learning: $W = x^T y$

$$y = \mathrm{sgn}(xW) = \mathrm{sgn}(xxy) = \mathrm{sgn}(\|x\|^2 y) = y$$

$$x^T = \mathrm{sgn}(W y^T) = \mathrm{sgn}(x^T y y^T) = \mathrm{sgn}(x^T \|y\|^2) = x^T$$

energy function: $E(x_i, y_i) = -\frac{1}{2} x_i W y_i^T$

generalized en. func. $E(x_i, y_i) = -\frac{1}{2} x_i W y_i^T + \frac{1}{2} \theta_l y_i^T + \frac{1}{2} x_i \theta_r^T$

$\hookrightarrow$ extended BAM $W' = \left(\begin{array}{c|c} -\theta_l- & \\ \hline W & \begin{array}{c} 1 \\ \theta_r \\ 1 \end{array} \end{array}\right)$

convergence:

$$E(x,y) - E(x',y') = -\frac{1}{2} g_i (x_i - x_i') \quad \swarrow \rightarrow \text{async update, pouze tady byla změna}$$

$E(x,y) - E(x',y') > 0$, protože $\mathrm{sgn}(g_i) \neq \mathrm{sgn}(x_i - x')$

tak nový stav (x',y') má nižší energii. Každý krok tedy redukuje energii.

$\Leftrightarrow$ máme pouze konečný počet kombinací.

Hopfield network:

fully connected network again

$$w_{ij} = \begin{cases} \sum_{s=1}^{m} x_i^s x_j^s & i \neq j \\ 0 & i = j \end{cases}$$

$$y_i(0) = x_i$$

$$y_j(t+1) = f_h\left(\sum_{i=1}^{n} w_{ij} y_i(t)\right)$$

state is stable:

$$\varepsilon = x_1 \cdot W = x_1 \cdot (x_1^T x_1 + \dots + x_m^T x_m - mI)$$

$$= \underbrace{x_1 x_1^T}_{n} x_1 + \underbrace{x_1 x_2^T}_{\alpha_{12}} x_2 + \dots + \underbrace{x_1 x_m^T}_{\alpha_{1m}} x_m - m x_1 I$$

$$= (n-m) \cdot x_1 + \overbrace{\sum_{j=2}^{m} \alpha_{1j} x_j}^{\text{perturbation}}$$

x_1 je stabilní, pokud $\sum_{j=2}^{m} \alpha_{1j} x_j$ je malý

$$\mathrm{sgn}(\varepsilon) = \mathrm{sgn}(x_1)$$

energy function: $-\frac{1}{2} x W x^T + \theta x^T$

Convergence

$$E = -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ij} x_i x_j$$

nechť se k-tý neuron změnil

$$x' = (x_1 \_\_ x'_k \_\_ x_n)$$

$$E(x') = -\frac{1}{2} \sum_{j \neq k}^{n} \sum_{i \neq k}^{m} w_{ij} x_i x_j - \sum_{i}^{n} w_{ik} x_i x'_k$$

$$E(x) - E(x') = -\sum_{i}^{n} w_{ij} x_i x_k - \left(-\sum_{i}^{n} w_{ij} x_i x'_k\right)$$

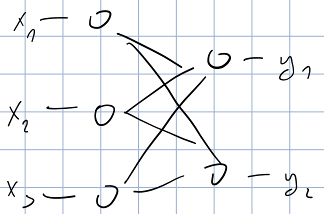
$$= -(x_k - x'_k) \cdot \sum_{i}^{n} w_{ik} x_i > 0$$

takže zase každým krokem jsem snížil tu energy function

BAM

$$y = xW$$

$$x = Wy^T$$



fully connected

$$y_0 = \text{sgn}(x_0 W)$$

$$x_1 = \text{sgn}(W y_0^T)$$

$$y_1 = \text{sgn}(x_1 W)$$

opět nastavujem $W = x^T y$

$$y = \text{sgn}(xW) = \text{sgn}(x x^T y) = \text{sgn}(\|x\|^2 y) = y$$

$$x^T = \text{sgn}(W y^T) = \text{sgn}(x^T y y^T) = \text{sgn}(x^T \|y\|^2) = x^T$$

$$E = -\frac{1}{2} x W y^T + \theta_\ell y^T + \theta_r^T x$$

$$W' = \left(\begin{array}{c|c} -\theta_\ell- & \\ \hline W & \begin{array}{c} | \\ \theta_r \\ | \end{array} \end{array}\right)$$

Konverguje to? async update, takže se mi změní jen jeden stav

$$E = -\frac{1}{2} (x_1 \_\_ x_n) \cdot \begin{pmatrix} g_1 \\ | \\ g_n \end{pmatrix}$$

mohl se mi změnit jen jeden x_i

$$E(x) - E(x') = -\frac{1}{2} g_i \cdot (x_i - x'_i)$$

$\text{sgn}(g_i) \neq \text{sgn}(x_i - x'_i)$, takže

$E(x) - E(x') > 0$, takže jsem snížil energy function.

Mám jen konečně mnoho stavů, každým krokem se zlepšuji.

Hopfield network:

$$w_{ij} = \begin{cases} \sum_{s=1}^{n} x_i^s x_j^s & i \neq j \\ 0 & i = j \end{cases}$$

→ tohle je $\boxed{\text{training}}$

$$y_i(0) = x_i$$

$$y_i(t+1) = f\left(\sum_{j=1}^{n} w_{ji} y_j(t)\right)$$

→ tohleto je $\boxed{\text{inference}}$

kdy je stav stabilní?

$$\mathcal{E} = X_1 \cdot W = X_1 \cdot \left(X_1^T X_1 + X_2^T X_2 + \dots + X_m^T X_m - mI\right)$$

$$= \underbrace{X_1 X_1^T}_{n} X_1 + \underbrace{X_1 X_2^T}_{\alpha_{12}} X_2 + \dots + \underbrace{X_1 X_m^T}_{\alpha_{1n}} X_m - mX_1 I$$

$$= (n-m) X_1 + \underbrace{\sum_{j=2}^{m} \alpha_{1j} X_j}_{\text{perturbation}}$$

je stabilní, pokud perturbation je malá, $n > m$

$$\text{sgn}(\mathcal{E}) = \text{sgn}(X_1)$$

energy function:

$$E = -\frac{1}{2} x W x^T + \theta x^T$$

converguje to? async dynamics

$$E = -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij} x_i x_j \qquad x' = (x_1 \_\_ x_k' \_\_ x_n)$$

$$E(x') = \underbrace{-\frac{1}{2} \sum_{j \neq k}^{n} \sum_{i \neq k}^{n} w_{ij} x_i x_j}_{\text{stojí}} - \sum_{i}^{n} w_{ik} x_i x_k'$$

$$E(x) - E(x') = -\sum_{i}^{n} w_{ik} x_i x_k - \left(-\sum_{i}^{n} w_{ik} x_i x_k'\right)$$

$$= -(x_k - x_k') \cdot \underbrace{\sum_{i}^{n} w_{ik} x_i}_{\text{potenciál}} > 0$$

tak ta energy function mi klesá v každém kroku, takže se blížím výsledku.

Multiflop:

using hopfield

$$E(x_1 \_ x_n) = \left(\sum_{i}^{n} x_i - 1\right)^2 = \sum_{i}^{n} x_i^2 + \sum_{i \neq j}^{n} x_i \cdot x_j - 2 \cdot \sum_{i=1}^{n} x_i + 1$$

$x_i^2 = x_i$ pro $x_i \in \{0, 1\}$

$$E = \sum_{i}^{n} x_i + \sum_{i \neq j}^{n} x_i \cdot x_j - 2 \sum_{i}^{n} x_i + 1 = \sum_{i \neq j}^{n} x_i \cdot x_j - \sum_{i}^{n} x_i + 1$$

$$= -\frac{1}{2} \sum_{i \neq j}^{n} (-2) \cdot x_i x_j + \sum_{i}^{n} (-1) \cdot x_i + 1$$

vynucují jeden nastavený na true

tohle jsou moje váhy

zakazují dva na true

simulated annealing:

$$P_{\Delta E} = \frac{1}{1 + e^{\Delta E / T}} \quad \text{-->} \quad x + \Delta x$$

bude updatnuto
s touto pravděpodobností

Multiflop: Can be solved with hopfield network:

$$E = \left(\sum_{i=1}^{n} x_i - 1\right)^2 = \sum_{i=1}^{n} x_i^2 + \sum_{i \neq j} x_i x_j - 2 \cdot \sum_{i}^{n} x_i + 1$$

$$= \sum_{i \neq j}^{n} x_i x_j - \sum_{i}^{n} x_i + 1$$

$$= -\frac{1}{2} \sum_{i \neq j}^{n} (-2) \cdot (x_i x_j) + \sum_{i}^{n} (-1) \cdot x_i + 1$$

Simulated annealing

$$P_{\Delta E} = \frac{1}{1 + e^{\Delta E / T}}$$

Self-organisation

we want to cluster data without supervision

problem is how to choose cluster prototype and how many we should have of those.

Competitive learning: winner takes it all, inhibition of neighbours, network plasticity

k-means

- grupuji do k clusterů
  - na začátku každý cluster obsahuje pouze jeden vektor
- jakmile přidám prvek do clusteru, upravím jeho centroid

$$c_i(new) = c_i(old) + \frac{1}{n_i}(x_i - c_i(old)))$$

Ojův alg.

- pro výpočet první komponenty PCA

mám na začátku náhodně zvolený w

pak iterativně upravuji:

$$w = w + \gamma\, \phi \cdot (x - \phi w)$$

Pokud existuje unikátní řešení, Oja ho najde:

$x - \phi w$ je ortogonální k w

jedna iterace attracts w to vectors from X

pokud $\|w\| = 1$, bude dáno do středu clusteru.

Nyní je potřeba ukázat, že se vektory samy normalizují aby bylo splněno toto

pokud $\|w\| > 1$

$(x - \phi w)$ je negativní projekce do w.

$$(x - (xw)w)w = x \cdot w - \|w\|^2 x \cdot w < 0$$

pokud $\|w\| < 1$

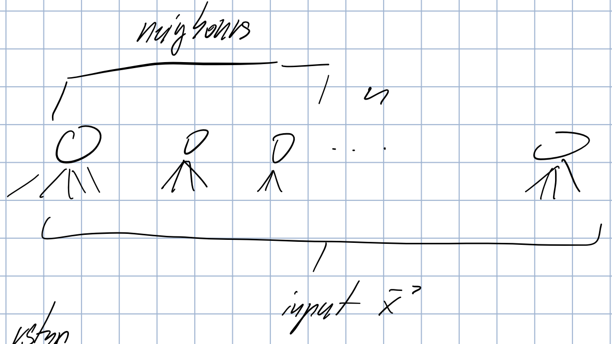
přesně obráceně

—> tedy pak to stlačuje hodnoty k 1 a dotáhne do prostředku clusteru.

$$c(new) = c(old) + \frac{1}{n_i}(x_i - c(old)))$$

→ čím větší je cluster, tím méně má nový prvek rozhýbat centroid

Kohonen maps:



Každý neuron dostane celý vstup

$$w(t+1) = w(t) + \alpha(t) \cdot \Phi(c,j) \cdot (x_i - w(t)))$$

0 1 0 0 1 0

0 1 * * * 0

$o(S)$ – počet fixů

$\delta(S)$ – vzdálenost mezi prvním a posledním fixem

$$P_i = \frac{f_i}{\sum_j f_j}$$

m' → očekávaný počet jedinců schematu H

$$m = m(H, t)$$

selection

$$m(H, t+1) = m(H,t) \cdot N \cdot \frac{f(H)}{\sum_j f_j}$$

→ pst, že schema vyhraje

crossover

$$P_S = 1 - \frac{\delta(H)}{l-1}$$

→ pst, že ind. přežije crossover

$$P_S = 1 - p_c \cdot \frac{\delta(H)}{l-1}$$

$$m(H, t+1) \geq m(H,t) \cdot N \cdot \frac{f(H)}{\sum_j f_j} \cdot \left(1 - p_c \cdot \frac{\delta(H)}{l-1}\right)$$

mutation

$$p[\text{přežije}] = (1-p_m)^{o(H)} \approx (1-p_m) \cdot o(H)$$

$p_m << 1$

$$m(H, t+1) = m(H,t) \cdot N \cdot \frac{f(H)}{\sum_j f_j} \cdot \left(1 - p_c \cdot \frac{\delta(H)}{l-1} - o(H) \cdot p_m\right)$$

Ojův alg.

- pro hledání hlavní komponenty

  nastavím w náhodně

  iterativně procházím prvky z X dokud mám čas

$$w = w + \gamma \cdot \phi \cdot (x - \phi w) \quad , \quad \Phi = x \cdot w$$

- jakmile skončím, tak w určuje směr hlavní komponenty z X.

Pokud existuje řešení, Oja ho najde.

$x - \phi w$ je ortogonální k w

Oja v každé iteraci posouvá w k prvkům z X.

Pokud $\|w\| = 1$, tak w je pouze posouváno doprostřed clusteru.

Co ale, pokud to tak není?

$\|w\| > 1$, pak $(x - \phi w)w$ je negativní projekce do w.

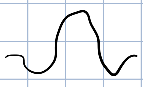
$$xw - \|w\|^2 xw < 0$$, takže to w bude zkracovat

$\|w\| < 1$, tak platí obrácená analogie.

Proto Oja se bude automaticky normalizovat.

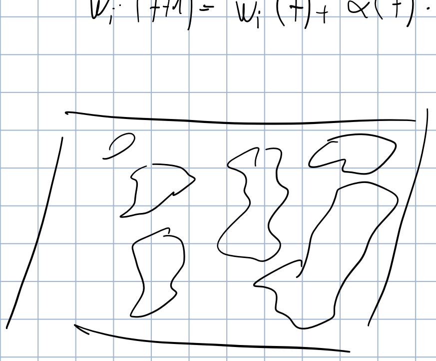
Kohonen maps

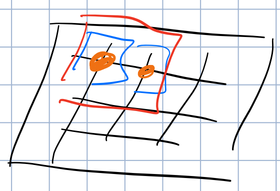
$h_{ij}(t)$ → neighbour function

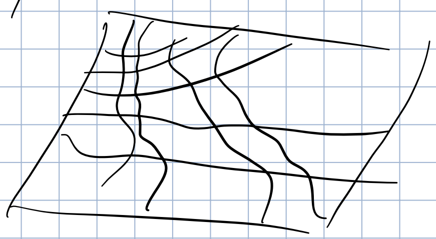


BMU → best matching unit

$$w_i(t+1) = w_i(t) + \alpha(t) \cdot h_{ij}(t) \cdot (x - w_i) \quad \rightarrow \text{BMU is } j$$







$$m = m(H,t)$$

→ selekce

$$m(H,t+1) = m(H,t) \cdot N \cdot \frac{f(H)}{\sum_j f_j}$$

→ crossover

$$m(H,t+1) \geq m(H,t) \cdot N \cdot \frac{f(H)}{\sum_j f_j} \cdot \left[1 - p_c \frac{\delta(H)}{l-1}\right]$$

$$(1 - p_m) \cdot o(H)$$

$$m(H,t+1) \geq m(H,t) \cdot N \cdot \frac{f(H)}{\sum_j f_j} \cdot \left[1 - p_c \frac{\delta(H)}{l-1} - p_m \cdot o(H)\right]$$

Second-order taylor

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

w.r.t. h

$$\nabla E(w+h) = \nabla E(w) + h^T \nabla^2 E(w)$$

→ setting to zero

Newton method

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

$$w(t+1) = w(t) - \overset{\uparrow}{\sigma}$$

tohle je ale moc kvůli tomu inverzu

uvažuju pouze $\frac{\partial^2 E}{\partial w_i^2}$ diagonální prvky

$$\frac{\nabla E(k) - \nabla E(k-1)}{\Delta w^{(k-1)}}$$ – sekant steps