RBF sítě

Organizační dynamika:

- Dvouvrstvá síť (výstupní neuron má bias)
- vstupy: $\vec{x} = (x_1, \ldots, x_n)$
- m skrytých neuronů

  Jejich hodnoty značíme $\phi_0, \phi_1, \ldots, \phi_m$ kde $\phi_0 = 1$ je formální jednotkový vstup pro výstupní neuron (viz. aktivní dynamika)

  Občas budu psát $\phi_0(\vec{x}), \phi_1(\vec{x}), \ldots, \phi_m(\vec{x})$ abych zdůraznil, že se jedná o hodnoty skrytých neuronů pro vstup sítě $\vec{x}$.
- jeden výstup: y
  (pro jednoduchost; lze zobecnit na libovolný počet)

Parametry sítě:

- Výstupní neuron je jako ADALINE: má váhový vektor $\vec{w} \in \mathbb{R}^{m+1}$
- Každý skrytý neuron i má tzv. střed $\vec{c}_i \in \mathbb{R}^n$ (odpovídá vahám ve standardní vícevrstvé síti) a šířku $\sigma_i \in \mathbb{R}^n$ (odpovídá strmosti ve standardní síti, tj. je to parametr aktivační funkce)

RBF sítě

Aktivní dynamika: Vnitřní potenciály skrytých neuronů:

$$\xi_j = \left\| \vec{x} - \vec{c}_j \right\|$$

Zde $\vec{c}_j$ je **střed** skrytého neuronu j.

Aktivační funkce skrytých neuronů $\varphi_0, \varphi_1, \ldots, \varphi_m$ jsou obecně libovolné diferencovatelné (a $\varphi_0 \equiv 1$). Budeme uvažovat:

$$\varphi_j(\xi_j) = \exp\left(-\frac{\xi_j^2}{2\sigma_j^2}\right) \qquad \left(= \phi_j(\vec{x}) \right)$$

Zde σ_j je **šířka** skrytého neuronu j.

Vnitřní potenciál výstupního neuronu:

$$\xi = \sum_{j=0}^{m} w_j \cdot \phi_j$$

Zde w_j jsou reálné **váhy**.

Aktivační funkce výstupu je identita: $y = \xi$.

RBF sítě

Funkce sítě:

$$y(\vec{x}) = w_0 + \sum_{j=1}^{m} w_j \cdot \exp\left(-\frac{\left\|\vec{x} - \vec{c}_j\right\|^2}{2\sigma_j^2}\right)$$

Tvrzení: Pro každou spojitou funkci $f : \mathbb{R}^n \to \mathbb{R}$ a každé $\varepsilon > 0$ existuje RBF síť taková, že její funkce y splňuje:

$$|f(\vec{x}) - y(\vec{x})| < \varepsilon \qquad \forall \vec{x} \in \mathbb{R}^n$$

RBF sítě

Adaptivní dynamika: Dána množina **tréninkových vzorů**

$$\mathcal{T} = \left\{\left(\vec{x}_1, d_1\right), \left(\vec{x}_2, d_2\right), \ldots, \left(\vec{x}_p, d_p\right)\right\}$$

Zde $\vec{x}_k = (x_{k1} \ldots, x_{kn}) \in \mathbb{R}^n$ je vstup k-tého vzoru a $d_k \in \mathbb{R}$ je očekávaný výstup.

Chybová funkce:

$$E = \frac{1}{2} \sum_{k=1}^{p} \left(y(\vec{x}_k) - d_k\right)^2$$

Cílem je nalézt následující parametry tak, aby byla chyba E minimalizována:

- váhy w_j výstupního neuronu
- středy $\vec{c}_j$ skrytých neuronů
- šířky σ_j skrytých neuronů

Sítě typu RBF - gradient

Učení lze provádět pomocí standardního gradientního sestupu. Gradient lze snadno spočítat přímým derivováním:

$$\frac{\delta E}{\delta w_j} = \sum_{k=1}^{p} (y(\vec{x}_k) - d_k) \cdot \phi_j$$

$$\frac{\delta E}{\delta \sigma_j} = \sum_{k=1}^{p} (y(\vec{x}_k) - d_k) \cdot w_j \cdot \exp\left(-\frac{\left\| \vec{x}_k - \vec{c}_j \right\|^2}{2\sigma_j^2} \right) \cdot \frac{\left\| \vec{x}_k - \vec{c}_j \right\|^2}{\sigma_j^3}$$

$$\frac{\delta E}{\delta c_{ji}} = \sum_{k=1}^{p} (y(\vec{x}_k) - d_k) \cdot w_j \cdot \exp\left(-\frac{\left\| \vec{x}_k - \vec{c}_j \right\|^2}{2\sigma_j^2} \right) \cdot \frac{(x_{ki} - c_{ji})^2}{\sigma_j^2}$$

Rychlost tohoto učení je srovnatelná s rychlostí zpětné propagace pro standardní (sigmoidální) sítě.

RBF - dvoufázový trénink

Velkou výhodou RBF sítí je, že jejich trénink lze rozdělit do dvou (téměř) nezávislých fází:

- trénink skrytých neuronů, tj.
  - umístění středů $\vec{c}_j$
  - nastavení velikosti šířek σ_j
- trénink vah $\vec{w}$ výstupního neuronu

RBF - středy a šířky

Rozmístění středů:
Pozorování: Máme velké množství dat (vstupů), pro malou část z nich máme předepsán požadovaný výstup (tréninková množina). Je dobré umístit středy neuronů do oblastí, které obsahují hodně datových bodů.

Středy lze proto umístit pomocí učení bez učitele - Kohonenovo učení (tj. k-means clustering), mapy apod.

Toto lze provádět na mnohem větší množině dat, pro které nepotřebujeme předepsaný výstup

Často se ovšem používá i jednoduché rovnoměrné rozmístění středů (pokud víme, že data jsou více méně pravidelně rozmístěna) nebo umístění středů do pozice náhodně vybraných vstupů.

Nastavení šířek: Aktivační funkce by neměly být ani příliš strmé ani příliš ploché. Údajně dobrou heuristikou je jednotná šířka $\sigma = D_{\max}/\sqrt{2m}$ kde $D_{\max}$ je maximální (Euklidovská) vzdálenost středů a m je počet skrytých neuronů.

RBF - váhy

Pro fixní hodnoty parametrů skrytých neuronů (tj. fixní středy a šířky) se jedná o ADALINE. Můžeme proto použít příslušné metody k minimalizaci chyby E.

Analytické řešení: Fixujeme všechny středy a šířky, chyba E je funkcí $\vec{w}$ (píšeme $E(\vec{w})$). Gradient funkce $E(\vec{w})$ je roven

$$\Phi^T \cdot (\Phi \cdot \vec{w} - \vec{d})$$

kde $\vec{d} = (d_1, \ldots, d_p)^\top$, $\vec{w} = (w_0, \ldots, w_m)^\top$ a Φ je matice $p \times (m+1)$ jejíž i-tý řádek obsahuje hodnoty skrytých neuronů pro i-tý vstup, tj. pro každé $i = 1, \ldots, p$ a $j = 1, \ldots, m$ máme

$$\Phi_{i1} = 1 \qquad \text{a} \qquad \Phi_{i(j+1)} = \phi_j(\vec{x}_i) = \exp\left(-\frac{\left\|\vec{x}_i - \vec{c}_j\right\|}{2\sigma_j^2}\right)$$

Vektor $\vec{w} = \Phi^+ \cdot \vec{d}$ kde $\Phi^+ = (\Phi^\top \cdot \Phi)^{-1} \cdot \Phi^\top$ potom řeší $\Phi^\top \cdot (\Phi \cdot \vec{w} - \vec{d}) = 0$ a tedy minimalizuje $E(\vec{w})$.

RBF sítě - regularizace

Naučená síť by měla dobře generalizovat. Neměla by příliš „opisovat“ tréninkové vzory (tj. neměla by být přetrénovaná).

Intuice: Funkce sítě, která příliš opisuje vzory se hodně kroutí – omezíme kroucení.

Definujeme novou chybovou fci

$$E' = E + \gamma \frac{1}{2} \sum_{k=1}^{p} \sum_{i=1}^{n} \left(\frac{\delta y}{\delta x_i^2}(\vec{x}_k) \right)^2$$

Zde $\gamma > 0$ je míra vlivu regularizace.

Váhy, které minimalizují $E'(\vec{w})$, jsou řešením: $M \cdot \vec{w} = \Phi^\top \cdot \vec{d}$ kde

$$M_{ij} = \sum_{k=1}^{p} \left(\phi_i(\vec{x}_k)\phi_j(\vec{x}_k) + \gamma \sum_{\ell=1}^{n} \left(\frac{\delta \phi_i}{\delta x_\ell^2}(\vec{x}_k) \frac{\delta \phi_j}{\delta x_\ell^2}(\vec{x}_k) \right) \right)$$

(Pozn. pro $\gamma = 0$ dostaneme $M = \Phi^\top \cdot \Phi$, tj. minimalizaci $E(\vec{w})$)

RBF sítě - výhody

Srovnejme s vícevrstvou sítí se sigmoidálními jednotkami (dále budu značit MP (multi-layer perceptron))

- Teoreticky lze aproximovat libovolnou spojitou fci (stejně jako MP)
- Dvoufázové učení: jednotlivé fáze jsou řádově rychlejší než učení MP (tj. zpětná propagace)
- RBF sítě jsou vhodné pro klasifikaci (více než MP) díky jejich lokálnímu charakteru. Vektory, které jsou daleko od tréninkových vzorů, dostanou malou hodnotu (toto nemusí platit pro MP).
- jsou vhodné pro aplikace s měnícími se daty (rychlé učení umožňuje on-line adaptaci na nová data - téměř nemožné s MP)
- snadná regularizace (lineární)

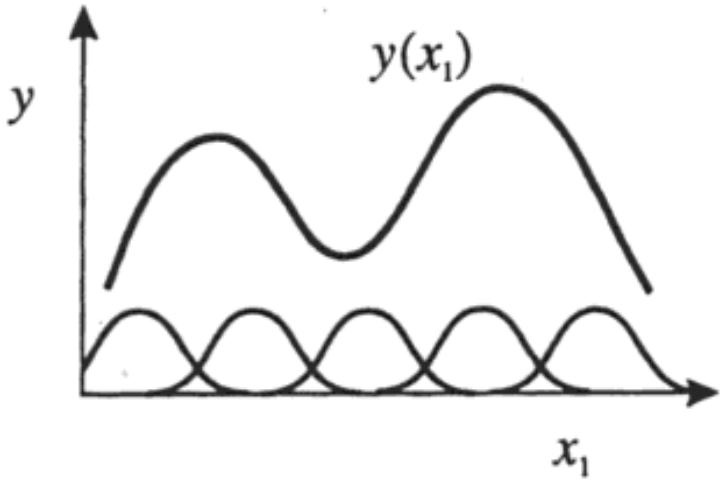
RBF sítě - nevýhody

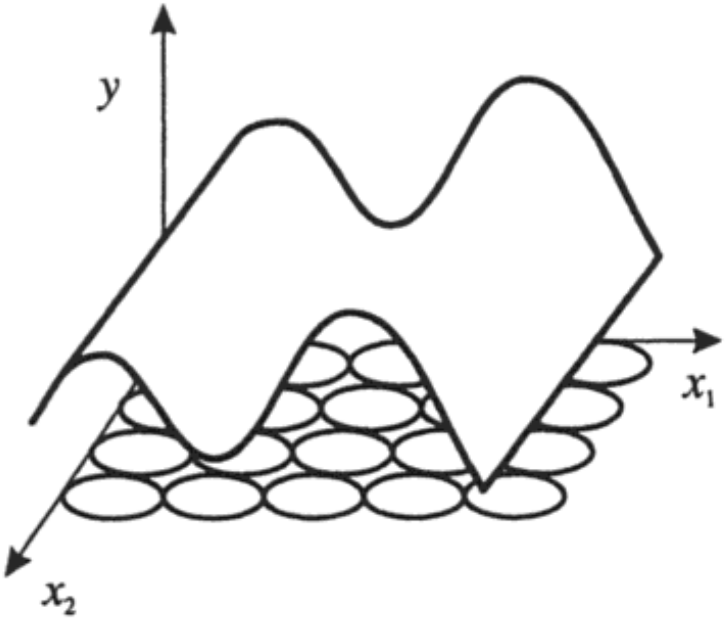
Opět srovnejme s MP.

- Teoretické výsledky o aproximaci nefungují v praxi (odhady na potřebný počet neuronů jsou nadsazené podobně jako u MP)
- Dvoufázové učení může být deformované pokud požadované funkční hodnoty nerespektují rozmístění vstupů (např. pokud je požadovaná funkce konstantní v oblasti s mnoha vzory, ale velmi variabilní v oblasti s málo vzory)
- RBF potřebují mnohem více dat i neuronů pro stejnou přesnost jako MP.
  - MP aproximují funkci globálně: každý vzor adaptuje většinu neuronů a informace je tedy distribuována po síti
  - RBF aproximují lokálně: pouze několik málo neuronů reaguje na daný vstup

  Z toho také plynou lepší extrapolační vlastnosti MP.
- RBF sítě mají větší problém s prokletím dimenzionality (exponenciální nárust jednotek s počtem dimenzí). Nejsou schopny odhalit nízkou variabilitu předepsané funkce v daném směru.

RBF sítě - nevýhody





kvadratický nárust počtu neuronů;
RBF neumí poznat, že je funkce (v podstatě) konstantní v x_2 (MP to umí)

Omezený Boltzmannův stroj (OBS)

Organizační dynamika:

- Cyklická síť se symetrickými spoji, neurony jsou rozděleny do dvou skupin:
  - V - viditelné
  - S - skryté

  Množina spojů je $V \times S$ (tj. úplný bipartitní graf)
- Množinu všech neuronů značíme N
- označme ξ_j vnitřní potenciál a y_j výstup (stav) neuronu j
- stav stroje: $\vec{y} \in \{0,1\}^{|N|}$.
- označme w_{ji} váhu spoje od neuronu i k neuronu j.
- obvykle se uvažuje bias, pro zjednodušení jej vynecháme.

Omezený Botzmannův stroj

Aktivní dynamika: Hodnoty viditelných neuronů jsou iniciálně nastaveny na hodnoty z množiny $\{0, 1\}$.

V t-tém kroku aktualizujeme neurony takto:

- t liché: náhodně zvolíme nové hodnoty skrytých neuronů, pro $j \in S$

$$\mathbf{P}\left[y_j^{(t)} = 1\right] = 1 \Big/ \left(1 + \exp\left(-\sum_{i \in V} w_{ji} y_i\right)\right)$$

- t sudé: náhodně zvolíme nové hodnoty viditelných neuronů, pro $j \in V$

$$\mathbf{P}\left[y_j^{(t)} = 1\right] = 1 \Big/ \left(1 + \exp\left(-\sum_{i \in S} w_{ji} y_i\right)\right)$$

Rovnovážný stav

Omezený Boltzmannův stroj se po jisté době dostane do *termální rovnováhy*. Tj. existuje čas t^* takový, že pro libovolný stav stroje $\gamma^* \in \{0,1\}^{|N|}$ platí

$$\mathbf{P}\left[\vec{y}^{(t^*)} = \gamma^*\right] \approx p_N(\gamma^*)$$

Zde $p_N(\gamma^*) = \frac{1}{Z} e^{-E(\gamma^*)/T}$ kde

$$Z = \sum_{\gamma \in \{0,1\}^{|N|}} e^{-E(\gamma)/T}$$

tj. Boltzmannovo rozložení

Síť tedy reprezentuje rozložení p_N.

Omezený Boltzmannův stroj - učení

Pro daný stav viditelných neuronů $\alpha \in \{-1, 1\}^{|V|}$ označme

$$p_V(\alpha) = \sum_{\beta \in \{-1,1\}^{|S|}} p_N(\alpha, \beta)$$

pravděpodobnost stavu viditelných neuronů α v termální rovnováze bez ohledu na stav skrytých neuronů.

Adaptivní dynamika:
Nechť p_d je pravděpodobnostní rozložení na množině stavů viditelných neuronů, tj. na $\{0, 1\}^{|V|}$.

Rozložení p_d může být dáno tréninkovou posloupností

$$\mathcal{T} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_p$$

tak, že

$$p_d(\alpha) = \#(\alpha, \mathcal{T})/p$$

kde $\#(\alpha, \mathcal{T})$ je počet výskytů α v posloupnosti $\mathcal{T}$

Cílem je nalézt konfiguraci sítě W takovou, že p_V odpovídá p_d.

Omezený Boltzmannův stroj - učení

Vhodnou mírou rozdílu mezi rozděleními p_V a p_d je relativní entropie zvážená pravděpodobnostmi vzorů (tzv. Kullback Leibler distance)

$$\mathcal{E}(\vec{w}) = \sum_{\alpha \in \{0,1\}^{|V|}} p_d(\alpha) \ln \frac{p_d(\alpha)}{p_V(\alpha)}$$

(Odpovídá maximální věrohodnosti vůči posloupnosti $\mathcal{T}$ v případě, že p_d je definováno pomocí $\mathcal{T}$)

Omezený Boltzmannův stroj - učení

$\mathcal{E}(\vec{w})$ budeme minimalizovat pomocí gradientního sestupu, tj. budeme počítat posloupnost vektorů vah $\vec{w}^{(0)}, \vec{w}^{(1)}, \ldots$

- váhy v $\vec{w}^{(0)}$ jsou inicializovány náhodně blízko 0
- v t-tém kroku (zde $t = 1, 2, \ldots$) je $\vec{w}^{(t)}$ vypočteno takto:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} + \Delta w_{ji}^{(t)}$$

kde

$$\Delta w_{ji}^{(t)} = -\varepsilon(t) \cdot \frac{\partial \mathcal{E}}{\partial w_{ji}}(\vec{w}^{(t-1)})$$

je změna váhy w_{ji} v t-tém kroku a $0 < \varepsilon(t) \leq 1$ je rychlost učení v t-tém kroku.

Zbývá spočítat (odhadnout) $\frac{\partial \mathcal{E}}{\partial w_{ji}}(\vec{w})$.

Omezený Boltzmannův stroj - učení

Lze ukázat, že

$$\frac{\partial \mathcal{E}}{\partial w_{ji}} = \left\langle y_j y_i \right\rangle_{fixed} - \left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}$$

- $\left\langle y_j y_i \right\rangle_{fixed}$ je průměrná hodnota $y_j y_i$ po jednom kroku výpočtu za předpokladu, že hodnoty viditelných neuronů jsou fixovány dle rozložení p_d.
- $\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}$ je průměrná hodnota $y_j^{(t^*)} y_i^{(t^*)}$ v termální rovnováze bez fixace viditelných neuronů.

Problém: výpočet $\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}$ trvá dlouho (musíme opakovaně přivést stroj do termální rovnováhy).

$\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}$ se proto nahrazuje $\left\langle y_j y_i \right\rangle_{recon}$ což je průměrná hodnota $y_j^{(2)} y_i^{(2)}$ za předpokladu, že iniciální hodnoty viditelných neuronů jsou voleny dle p_d.

Omezený Boltzmannův stroj - učení

Tedy

$$\Delta w_{ji}^{(t)} = -\varepsilon(t) \cdot \left(\left\langle y_j y_i \right\rangle_{fixed} - \left\langle y_j y_i \right\rangle_{recon} \right)$$

- $\left\langle y_j y_i \right\rangle_{fixed}$ se vypočte takto: Polož $\mathcal{Y} := 0$ a opakuj q krát:
  - fixuj náhodně hodnoty viditelných neuronů dle p_d
  - simuluj jeden krok výpočtu a přičti aktuální hodnotu $y_j y_i$ k $\mathcal{Y}$

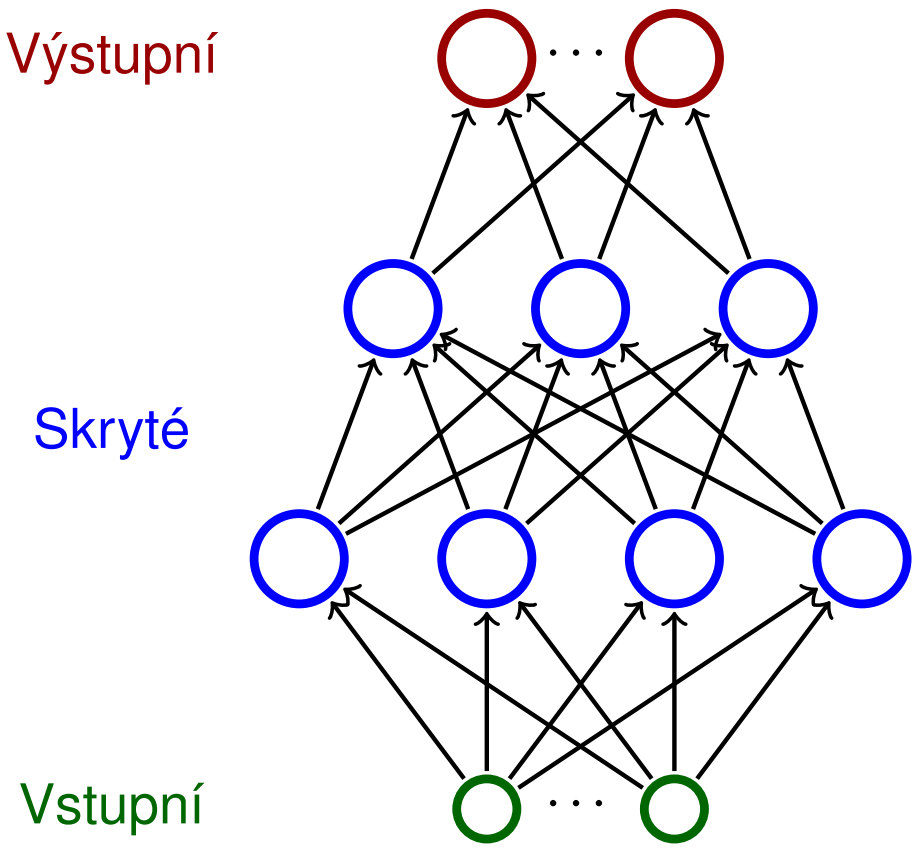
  Pro vhodné q bude $\mathcal{Y}/q$ dobrým odhadem $\left\langle y_j y_i \right\rangle_{fixed}$
- $\left\langle y_j y_i \right\rangle_{recon}$ se vypočte takto: Polož $\mathcal{Y} := 0$ a opakuj q krát:
  - nastav náhodně hodnoty viditelných neuronů dle p_d
  - simuluj dva kroky výpočtu a přičti aktuální hodnotu $y_j y_i$ k $\mathcal{Y}$ (tj. vypočti hodnoty skrytých neuronů, potom hodnoty viditelných (tzv. rekonstrukci vstupu))

  Pro vhodné q bude $\mathcal{Y}/q$ dobrým odhadem $\left\langle y_j y_i \right\rangle_{recon}$

Hluboké sítě

Standardní vícevrstvá síť

Organizační dynamika:



- Neurony jsou rozděleny do **vrstev** (vstupní a výstupní vrstva, obecně několik skrytých vrstev)
- Vrstvy číslujeme od 0; vstupní vrstva je nultá
  - Např. třívrstvá síť se skládá z jedné vstupní, dvou skrytých a jedné výstupní vrstvy.
- Neurony v ℓ-té vrstvě jsou spojeny se všemi neurony ve vrstvě $\ell + 1$.

Hluboké sítě

Značení:

- Označme
  - X množinu vstupních neuronů
  - Y množinu výstupních neuronů
  - Z množinu všech neuronů (tedy $X, Y \subseteq Z$)
- jednotlivé neurony budeme značit indexy i, j apod.
- ξ_j je vnitřní potenciál neuronu j po skončení výpočtu
- y_j je stav (výstup) neuronu j po skončení výpočtu
- w_{ji} je váha spoje **od** neuronu i **do** neuronu j
- $j_{\leftarrow}$ je množina všech neuronů, **z nichž** vede spoj do j (zejména $0 \in j_{\leftarrow}$)
- $j^{\rightarrow}$ je množina všech neuronů, **do nichž** vede spoj z j

Hluboké sítě

Aktivní dynamika:

- vnitřní potenciál neuronu j:

$$\xi_j = \sum_{i \in j_{\leftarrow}} w_{ji} y_i$$

- aktivační funkce

$$\sigma = \frac{1}{1 + e^{-\xi}}$$

pro všechny neurony stejná!

- Stav nevstupního neuronu $j \in Z \setminus X$ po skončení výpočtu je

$$y_j = \sigma(\xi_j)$$

(y_j závisí na konfiguraci $\vec{w}$ a vstupu $\vec{x}$, proto budu občas psát $y_j(\vec{w}, \vec{x})$)

- Síť počítá funkci z $\mathbb{R}^{|X|}$ do $\mathbb{R}^{|Y|}$. Výpočet probíhá po vrstvách. Na začátku jsou hodnoty vstupních neuronů nastaveny na vstup sítě. V kroku ℓ jsou vyhodnoceny neurony z ℓ-té vrstvy.

Hluboké sítě – adaptivní dynamika

Tréninková posloupnost $\mathcal{T}$ vzorů tvaru

$$\left(\vec{x}_1, \vec{d}_1\right), \left(\vec{x}_2, \vec{d}_2\right), \ldots, \left(\vec{x}_p, \vec{d}_p\right)$$

kde každé $\vec{x}_k \in \{0,1\}^{|X|}$ je vstupní vektor a každé $\vec{d}_k \in \{0,1\}^{|Y|}$ je očekávaný výstup sítě. Pro každé $j \in Y$ označme d_{kj} očekávaný výstup neuronu j pro vstup $\vec{x}_k$
(vektor $\vec{d}_k$ lze tedy psát jako $\left(d_{kj}\right)_{j \in Y}$).

Chybová funkce

$$E(\vec{w}) = \sum_{k=1}^{p} E_k(\vec{w})$$

kde

$$E_k(\vec{w}) = \frac{1}{2} \sum_{j \in Y} \left(y_j(\vec{w}, \vec{x}_k) - d_{kj}\right)^2$$

Proč hluboké sítě

... když jedna vrstva stačí k aproximaci libovolné rozumné funkce?

- Jedna vrstva je často značně neefektivní, tj. může vyžadovat obrovský počet skrytých neuronů pro reprezentaci dané funkce

  Výsledky z teorie Booleovských obvodů ukazují, že nutný počet neuronů může být exponenciální vzhledem k dimenzi vstupu

... ok, proč neučit hluboké sítě pomocí obyčejné zpětné propagace?

- Rychlost učení rapidně klesá s počtem vrstev
- Hluboké sítě mají tendenci se snadno přetrénovat

Hluboké sítě – adaptivní dynamika

Předpokládejme k vrstvou síť.Označme

- W_i matici vah mezi vrstvami $i-1$ a i
- F_i funkci počítanou částí sítě s vrstvami $0, 1, \ldots, i$

  tedy F_1 je funkce počítaná jednovrstvou sítí skládající se ze vstupní a první vrstvy sítě, F_k je funkce celé sítě

Všimněte si, že pro každé i lze vrstvy $i-1$ a i společně s maticí W_i chápat jako omezený Boltzmannův stroj B_i (předpokládáme $T = 1$)

Učení ve dvou fázích:

- předtrénování bez učitele: Postupně pro každé $i = 1, \ldots, k$ trénuj OBS B_i na náhodně volených vstupech z posloupnosti

  $$F_{i-1}(\vec{x}_1), \ldots, F_{i-1}(\vec{x}_p)$$

  pomocí algoritmu pro učení OBS (zde $F_0(\vec{x}_i) = \vec{x}_i$)

  (tedy B_i se trénuje na tréninkových vzorech transformovaných vrstvami $0, \ldots, i-1$)
- doladění sítě s učitelem např. pomocí zpětné propagace

Hluboké sítě

Po první fázi dostaneme k vrstvou síť, která reprezentuje rozložení na datech. Z tohoto rozložení lze samplovat takto:

- přiveď nejvyšší OBS do termální rovnováhy (to dá hodnoty neuronů v nejvyšších dvou vrstvách)
- propaguj hodnoty do nižších vrstev (tj. proveď jeden krok aktualizace stavů mezilehlých OBS)
- stav neuronů v nejspodnější vrstvě potom bude představovat vzorek dat; pravděpodobnost s jakou se tam objeví konkrétní stav je pravděpodností onoho stavu v rozložení reprezentovaném sítí