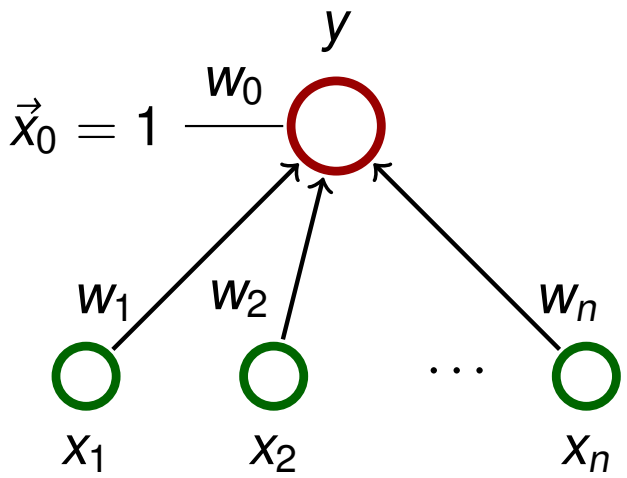
ADALINE

Organizační dynamika:



$\vec{w} = (w_0, w_1, \ldots, w_n)$ a $\vec{X} = (x_0, x_1, \ldots, x_n)$ kde $x_0 = 1$.

Aktivní dynamika:

- vnitřní potenciál: $\xi = w_0 + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{X}$
- aktivační funkce: $\sigma(\xi) = \xi$
- funkce sítě: $y[\vec{w}](x_1, \ldots, x_n) = \sigma(\xi) = \vec{w} \cdot \vec{X}$

ADALINE - statistické učení

Uvažme následující modifikaci **adaptivní dynamiky**:

- Síti je předkládána (nekonečná) posloupnost tréninkových vzorů $\left(\vec{x}_1, d_1\right), \left(\vec{x}_2, d_2\right), \ldots$ kde
  - vstupy $\vec{x}_k = (x_{k1}, \ldots, x_{kn}) \in \mathbb{R}^n$ jsou generovány náhodně s daným rozdělením pravděpodobností
  - každý vstup $\vec{x}_k$ má přiřazen požadovaný výstup $d_k \in \mathbb{R}$.
- Označme $\vec{X}_k = (x_{k0}, x_{k1}, \ldots, x_{kn})$ kde $x_{k0} = 1$.
  Výstup sítě pro k-tý vzor je potom $\vec{w} \cdot \vec{X}_k$.
- Pro danou konfiguraci $\vec{w}$ definujeme chybu:

$$E(\vec{w}) = \lim_{p \to \infty} \frac{1}{p} \cdot \sum_{k=1}^{p} \frac{1}{2} \left(\vec{w} \cdot \vec{X}_k - d_k\right)^2$$

Poznámka: Podle zákona o velkých číslech je $E(\vec{w})$ střední hodnota proměnné, která vrací $\frac{1}{2}\left(\vec{w} \cdot \vec{X}_k - d_k\right)^2$, tedy $E(\vec{w}) = \mathbb{E}\left[\frac{1}{2}\left(\vec{w} \cdot \vec{X}_k - d_k\right)^2\right]$

ADALINE - statistické učení

Vypočteme gradient $\nabla E(\vec{w}) = \left(\frac{\partial E}{\partial w_0}(\vec{w}), \ldots, \frac{\partial E}{\partial w_n}(\vec{w})\right)$:

$$\begin{aligned}\frac{\partial E}{\partial w_i}(\vec{w}) &= \lim_{p\to\infty} \frac{1}{p} \cdot \sum_{k=1}^{p} \left(\vec{w} \cdot \vec{X}_k - d_k\right) \cdot x_{ki} \\ &= \sum_{r=0}^{n} w_r \cdot A_{ri} - C_i\end{aligned}$$

kde

$$A_{ri} = \lim_{p\to\infty} \frac{1}{p} \cdot \sum_{k=1}^{p} x_{kr} \cdot x_{ki}$$

$$C_i = \lim_{p\to\infty} \frac{1}{p} \cdot \sum_{k=1}^{p} d_k \cdot x_{ki}$$

ADALINE - statistické učení

Hledáme minimum $E(\vec{w})$, tedy $\vec{w}^*$ takové, že $\nabla E(\vec{w}^*) = 0$.

Pokud dokážeme statisticky odhadnout A_{ri} a C_i pak můžeme rovnou položit:

$$0 = \frac{\partial E}{\partial w_i}(\vec{w}) = \sum_{r=0}^{n} w_r \cdot A_{ri} - C_i$$

a dostat $\vec{w}^*$ jako řešení systému lineárních rovnic.

Toto není příliš praktické protože

- řešení nemusí existovat (přestože minimum funkce E vždy existuje)
- systém rovnic je obvykle hodně velký

ADALINE - statistické učení

Obvykle se používá gradientní sestup stejně jako v případě normálního ADALINE učení:

- 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:

$$\begin{aligned} \vec{w}^{(t)} &= \vec{w}^{(t-1)} - \varepsilon \cdot \nabla E(\vec{w}^{(t-1)}) \\ &= \vec{w}^{(t-1)} - \varepsilon \cdot \lim_{p \to \infty} \frac{1}{p} \cdot \sum_{k=1}^{p} \left(\vec{w}^{(t-1)} \cdot \vec{X}_k - d_k \right) \cdot \vec{X}_k \end{aligned}$$

Problém: Gradient je průměr nekonečně mnoha hodnot!

ADALINE - statistické učení

Naštěstí funguje online algoritmus (Widrow & Hoff), který je úplně stejný jako v předchozí „nestatistické“ variantě!

- 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:

$$\vec{w}^{(t)} = \vec{w}^{(t-1)} - \varepsilon(t) \cdot \left(\vec{w}^{(t-1)} \cdot \vec{X}_k - d_k\right) \cdot \vec{X}_k$$

kde $k = ((t-1) \mod p) + 1$ a $0 < \varepsilon(t) \leq 1$ je rychlost učení v t-tém kroku.

Věta (Widrow & Hoff)

Pro $\varepsilon(t) = \frac{1}{t}$ posloupnost $\vec{w}^{(0)}, \vec{w}^{(1)}, \vec{w}^{(2)}, \ldots$ konverguje ke globálnímu minimu chybové funkce $E(\vec{w})$.

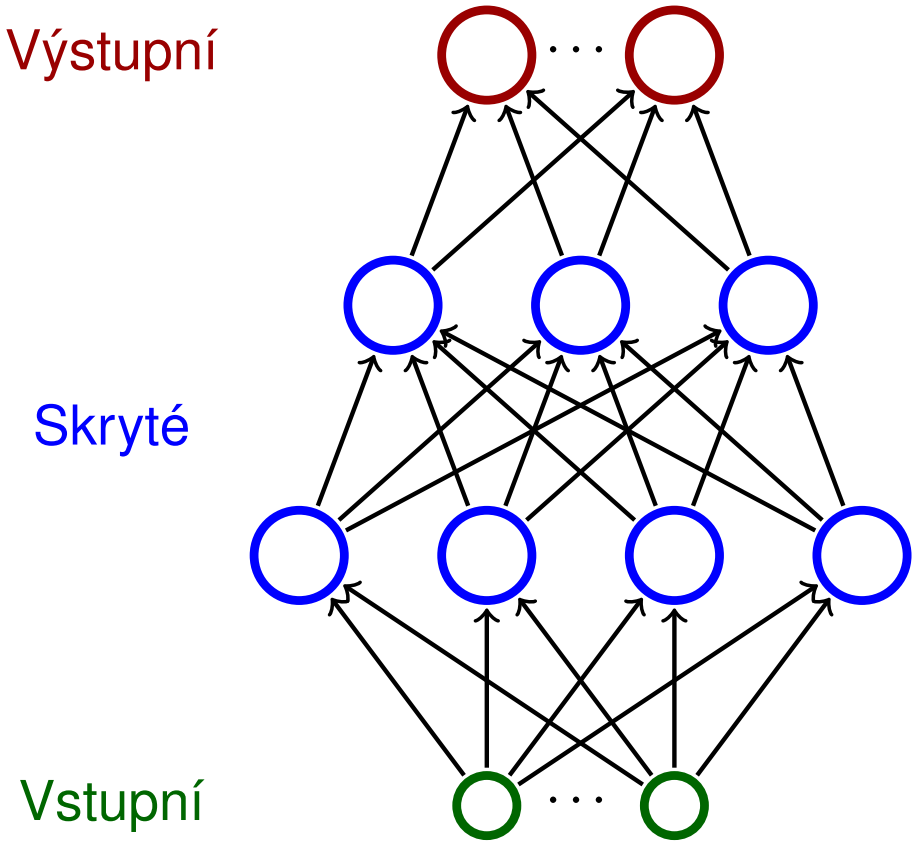
Matematická poznámka: zde se nejedná o konvergenci po složkách, ale o konvergenci průměru čtverců vzdálenosti (mean square convergence).

Vícevrstvá síť a zpětná propagace

- Vícevrstvé sítě - značení
- učící pravidlo zpětné propagace
- algoritmus zpětné propagace

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 $\ell + 1$.
- Vícevrstvou síť lze zadat počty neuronů v jednotlivých vrstvách (např. 2-4-3-2)

Vícevrstvá síť

Zde je formálnější definice vícevrstvé sítě:

Definice

***K*-vrstvá síť** *je acyklická síť jejíž množinu neuronů (s biasy) je možné zapsat jako* $V_0 \cup V_1 \cup \ldots \cup V_K$ *kde*

- *množiny* V_ℓ *jsou vzájemně disjunktní*
- *všechny neurony z* V_0 *jsou vstupní*
- *všechny neurony z* V_K *jsou výstupní*
- *všechny neurony z* $V_1 \cup \ldots \cup V_{K-1}$ *jsou skryté (tedy zejména nejsou ani vstupní ani výstupní)*
- *pro každé* $\ell = 0, \ldots, K-1$ *platí, že výstup každého neuronu z* V_ℓ *je vstupem každého neuronu z* $V_{\ell+1}$ *(a v síti nejsou žádné další spoje)*

Vícevrstvá síť

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

  (zde definujeme $y_0 = 1$ jako hodnotu formálního jednotkového vstupu)
- w_{ji} je váha spoje **od** neuronu i **do** neuronu j

  (zejména w_{j0} je váha speciálního jednotkového vstupu, tj. $w_{j0} = -b_j$ kde b_j je bias 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

Vícevrstvá síť

Aktivní dynamika:

- vnitřní potenciál neuronu j:

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

- aktivační funkce σ_j pro neuron j:

$$\sigma_j(\xi) = \frac{1}{1 + e^{-\lambda_j \xi}}$$

(obvykle se uvažuje $\lambda_j = 1$ pro všechna j, ale obecně mohou být různé)

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

$$y_j = \sigma_j(\xi_j) = \frac{1}{1 + e^{-\lambda_j \xi_j}}$$

(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.

Vícevrstvá síť

Adaptivní dynamika:

- Dána množina $\mathcal{T}$ **tréninkových vzorů** tvaru

$$\left\{ \left(\vec{x}_k, \vec{d}_k \right) \quad \middle| \quad k = 1, \ldots, p \right\}$$

kde každé $\vec{x}_k \in \mathbb{R}^{|X|}$ je vstupní vektor a každé $\vec{d}_k \in \mathbb{R}^{|Y|}$ 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

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

Vícevrstvá síť - učící algoritmus

Dávkový algoritmus (gradientní sestup):

Algoritmus počítá 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

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

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

Všimněte si, že $\frac{\partial E}{\partial w_{ji}}(\vec{w}^{(t-1)})$ je komponenta gradientu ∇E, tedy změnu vah v t-tém kroku lze zapsat také takto: $\vec{w}^{(t)} = \vec{w}^{(t-1)} - \varepsilon(t) \cdot \nabla E(\vec{w}^{(t-1)})$.

Vícevrstvá síť - gradient chybové funkce

Pro každé w_{ji} máme

$$\frac{\partial E}{\partial w_{ji}} = \sum_{k=1}^{p} \frac{\partial E_k}{\partial w_{ji}}$$

kde pro každé $k = 1, \ldots, p$ platí

$$\frac{\partial E_k}{\partial w_{ji}} = \frac{\partial E_k}{\partial y_j} \lambda_j y_j (1 - y_j) y_i$$

a pro každé $j \in Z \setminus X$ dostaneme

$$\frac{\partial E_k}{\partial y_j} = y_j - d_{kj} \qquad \text{pro } j \in Y$$

$$\frac{\partial E_k}{\partial y_j} = \sum_{r \in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r} \lambda_r y_r (1 - y_r) w_{rj} \qquad \text{pro } j \in Z \setminus (Y \cup X)$$

(Zde všechna y_j jsou ve skutečnosti $y_j(\vec{w}, \vec{x}_k)$).

Vícevrstvá síť - výpočet gradientu

Algoritmicky lze $\frac{\partial E}{\partial w_{ji}}$ spočítat takto:

Polož $\mathcal{E}_{ji} := 0$ (na konci výpočtu bude $\mathcal{E}_{ji} = \frac{\partial E}{\partial w_{ji}}$)

Pro každé $k = 1, \ldots, p$ udělej následující

1. spočítej y_j pro každé $j \in Z$ a k-tý vzor, tedy $y_j = y_j(\vec{w}, \vec{x}_k)$
   (tj. vyhodnoť síť ve standardním aktivním režimu)
2. pro všechna $j \in Z$ spočítej $\frac{\partial E_k}{\partial y_j}$ pomocí zpětného šíření
   (viz. následující slajd!)
3. spočítej $\frac{\partial E_k}{\partial w_{ji}}$ pro všechna w_{ji} pomocí vzorce

$$\frac{\partial E_k}{\partial w_{ji}} := \frac{\partial E_k}{\partial y_j} \lambda_j y_j (1 - y_j) y_i$$

4. $\mathcal{E}_{ji} := \mathcal{E}_{ji} + \frac{\partial E_k}{\partial w_{ji}}$

Výsledné $\mathcal{E}_{ji}$ se rovná $\frac{\partial E}{\partial w_{ji}}$.

Vícevrstvá síť - zpětné šíření

$\frac{\partial E_k}{\partial y_j}$ lze spočítat v lineárním čase (s jednotkovou aritmetikou) pomocí zpětného šíření:

- spočítáme $\frac{\partial E_k}{\partial y_j}$ pro $j \in Y$ pomocí vzorce $\frac{\partial E_k}{\partial y_j} = y_j - d_{kj}$
- rekurzivně spočítáme zbylé $\frac{\partial E_k}{\partial y_j}$:

  Nechť j je v ℓ-té vrstvě a předpokládejme, že $\frac{\partial E_k}{\partial y_r}$ už máme spočítáno pro všechny neurony z vyšších vrstev (tedy vrstev $\ell + 1, \ell + 2, \ldots$).
  Pak lze $\frac{\partial E_k}{\partial y_j}$ spočítat pomocí vzorce

$$\frac{\partial E_k}{\partial y_j} = \sum_{r \in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r} \lambda_r y_r (1 - y_r) w_{rj}$$

  protože všechny neurony $r \in j^{\rightarrow}$ patří do vrstvy $\ell + 1$.

Složitost dávkového algoritmu

Výpočet hodnoty $\frac{\partial E}{\partial w_{ji}}(\vec{w}^{(t-1)})$ probíhá v lineárním čase vzhledem k velikosti sítě a počtu tréninkových vzorů.

(předpokládáme jednotkovou cenu aritmetických operací)

Zdůvodnění: Algoritmus provede p krát následující

1. vyhodnocení sítě (tj. výpočet $y_j(\vec{w}, \vec{x}_k)$)
2. výpočet $\frac{\partial E_k}{\partial y_j}$ zpětným šířením
3. výpočet $\frac{\partial E_k}{\partial w_{ji}}$
4. přičtení $\frac{\partial E_k}{\partial w_{ji}}$ k $\mathcal{E}_{ji}$

Kroky 1. - 3. proběhnou v lineárním čase a krok 4. v konstantním vzhledem k velikosti sítě.

Počet iterací gradientního sestupu pro přiblížení se (lokálnímu) minimu může být velký ...

Vícevrstvá síť - učící algoritmus

Online algoritmus:

Algoritmus počítá 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:

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

kde

$$\Delta w_{ji}^{(t)} = -\varepsilon(t) \cdot \frac{\partial E_k}{\partial w_{ji}}(w_{ji}^{(t-1)})$$

kde $k = ((t-1) \mod p) + 1$ a $0 < \varepsilon(t) \leq 1$ je rychlost učení v t-tém kroku.

Lze použít i **stochastickou verzi** tohoto algoritmu, v níž je k voleno náhodně z $\{1, \ldots, p\}$.