

Vytěžování Dat

Přednáška 10 – Neuronové sítě s dopřednou architekturou

Miroslav Čepek

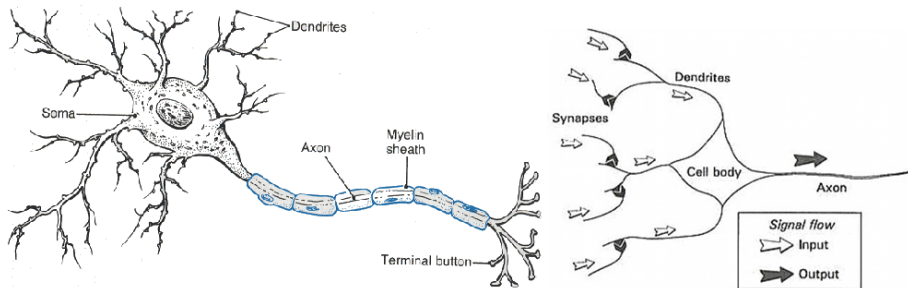
Fakulta Elektrotechnická, ČVUT

9.12.2011

Umělé neuronové sítě

- Umělá neuronová síť je matematickým přiblížením biologické neuronové sítě. A snahou je napodobovat funkci biologických neuronových sítí.
- Umělé neuronové sítě představují jiný přístup k řešení problémů než konvenční výpočetní technika.
- Ale mezi schopnostmi umělých neuronových sítí a mozku je přecejen ještě obrovský rozdíl.

Biologická neuronová síť



- Více si můžete přečíst například na <http://www.mindcreators.com/NeuronBasics.htm> nebo <http://www.generation5.org/content/2000/nn00.asp>.

Umělá neuronová síť

- Se skládá ze vzájemně propojených jednotek – neuronů.
- Každý neuron transformuje své vstupy na výstup. Čili umí jen velmi málo, ale jejich síla je v jejich spolupráci.
- Konekcionistivký přístup k umělé inteligenci – představa, že spoluprací mnoha hloupých jednotek dosáhnou chování, které je kvalitativně "chytřejší" než jen součet možností jednotlivých elementů.
- Neuron při transformaci může uplantit svou lokální paměť.
- Výstupy neuronu jsou přivedeny na vstupy dalších neuronů.

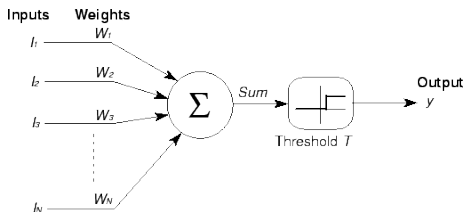
Umělá neuronová síť

- Se skládá ze vzájemně propojených jednotek – neuronů.
- Každý neuron transformuje své vstupy na výstup. Čili umí jen velmi málo, ale jejich síla je v jejich spolupráci.
- Konekcionistivký přístup k umělé inteligenci – představa, že spoluprací mnoha hloupých jednotek dosáhnou chování, které je kvalitativně "chytřejší" než jen součet možností jednotlivých elementů.
- Neuron při transformaci může uplantit svou lokální paměť.
- Výstupy neuronu jsou přivedeny na vstupy dalších neuronů.

Umělá neuronová síť

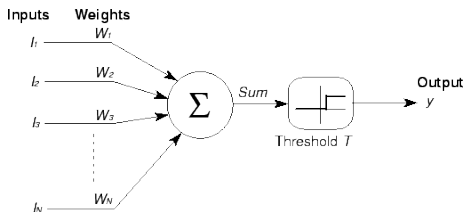
- Se skládá ze vzájemně propojených jednotek – neuronů.
- Každý neuron transformuje své vstupy na výstup. Čili umí jen velmi málo, ale jejich síla je v jejich spolupráci.
- Konekcionistivký přístup k umělé inteligenci – představa, že spoluprací mnoha hloupých jednotek dosáhnou chování, které je kvalitativně "chytřejší" než jen součet možností jednotlivých elementů.
- Neuron při transformaci může uplantit svou lokální paměť.
- Výstupy neuronu jsou přivedeny na vstupy dalších neuronů.

- První, kdo se začal zabývat výzkumem umělých neuronových sítí byl v roce 1943 jistý profesor McCulloch a jeho žák Pitts. Oba v prvním plánu hledali matematický model biologického neuronu.
- Tento jejich model se dodnes používá a nazývá se McCulloch-Pittsův neuron.
- Občas se označuje také jako Perceptron.



<http://wwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node12.html>

- První, kdo se začal zabývat výzkumem umělých neuronových sítí byl v roce 1943 jistý profesor McCulloch a jeho žák Pitts. Oba v prvním plánu hledali matematický model biologického neuronu.
- Tento jejich model se dodnes používá a nazývá se McCulloch-Pittsův neuron.
- Občas se označuje také jako Perceptron.



<http://wwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node12.html>

Princip práce McCulloch-Pittsova neuronu

- Na vstupy x_1, x_2, \dots, x_n (na předchozím odrážku označeny jako l_1, l_2, \dots, l_n) předložím pozorované hodnoty.
- Tyto hodnoty přenásobím vahami w_1, w_2, \dots, w_n a spočítám tzv. aktivaci neuronu.

$$a = \sum_{i=1}^n w_i x_i$$

- Aktivace odpovídá aktivaci biologického neuronu. A ten "vypálí" výstup jen v případě, že aktivace překročí určitý práh.
- To se v umělých neuronových sítích simuluje pomocí aktivační funkce.
- V případě perceptronu skokovou funkcí. $f(a) = 0$, pro $a < konst$, jinak $f(a) = 1$.
- Nepřipomíná vám to něco?

Princip práce McCulloch-Pittsova neuronu

- Na vstupy x_1, x_2, \dots, x_n (na předchozím odrážku označeny jako I_1, I_2, \dots, I_n) předložím pozorované hodnoty.
- Tyto hodnoty přenásobím vahami w_1, w_2, \dots, w_n a spočítám tzv. aktivaci neuronu.

$$a = \sum_{i=1}^n w_i x_i$$

- Aktivace odpovídá aktivaci biologického neuronu. A ten "vypálí" výstup jen v případě, že aktivace překročí určitý práh.
- To se v umělých neuronových sítích simuluje pomocí aktivační funkce.
- V případě perceptronu skokovou funkcí. $f(a) = 0$, pro $a < konst$, jinak $f(a) = 1$.
- Nepřipomíná vám to něco?

Princip práce McCulloch-Pittsova neuronu

- Na vstupy x_1, x_2, \dots, x_n (na předchozím odrážku označeny jako I_1, I_2, \dots, I_n) předložím pozorované hodnoty.
- Tyto hodnoty přenásobím vahami w_1, w_2, \dots, w_n a spočítám tzv. aktivaci neuronu.

$$a = \sum_{i=1}^n w_i x_i$$

- Aktivace odpovídá aktivaci biologického neuronu. A ten "vypálí" výstup jen v případě, že aktivace překročí určitý práh.
- To se v umělých neuronových sítích simuluje pomocí aktivační funkce.
- V případě perceptronu skokovou funkcí. $f(a) = 0$, pro $a < konst$, jinak $f(a) = 1$.
- Nepřipomíná vám to něco?

Rozhodovací hranice a učení

- Jak vypadá rozhodovací hranice Perceptronu (McCulloch-Pittsova neuronu)?
- A jak nastavíme váhy?
- Přece pomocí perceptronového algoritmu!
- Jaké jsou limity perceptronu?

Rozhodovací hranice a učení

- Jak vypadá rozhodovací hranice Perceptronu (McCulloch-Pittsova neuronu)?
- A jak nastavíme váhy?
- Přece pomocí perceptronového algoritmu!
- Jaké jsou limity perceptronu?

Rozhodovací hranice a učení

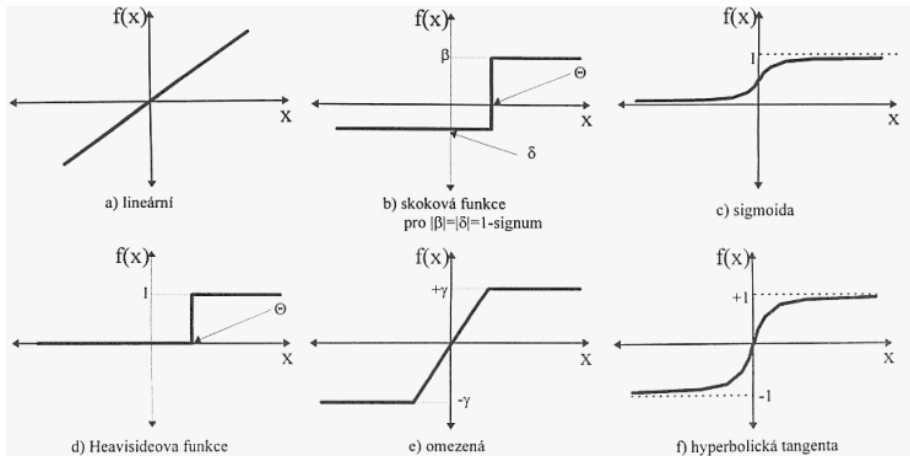
- Jak vypadá rozhodovací hranice Perceptronu (McCulloch-Pittsova neuronu)?
- A jak nastavíme váhy?
- Přece pomocí perceptronového algoritmu!
- Jaké jsou limity perceptronu?

Rozhodovací hranice a učení

- Jak vypadá rozhodovací hranice Perceptronu (McCulloch-Pittsova neuronu)?
- A jak nastavíme váhy?
- Přece pomocí perceptronového algoritmu!
- Jaké jsou limity perceptronu?

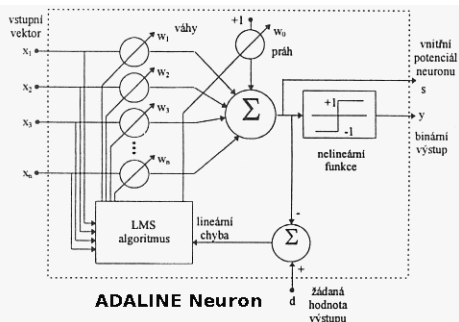
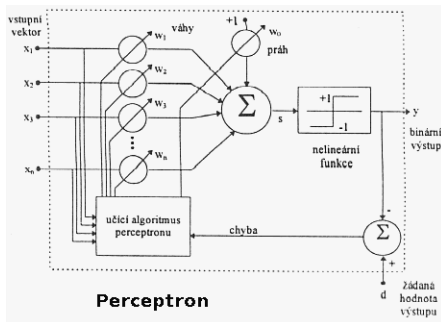
Aktivační funkce

- Prvním vylepšením perceptronu jsou různé aktivační funkce, kterými se transformuje aktivaci na výstup.



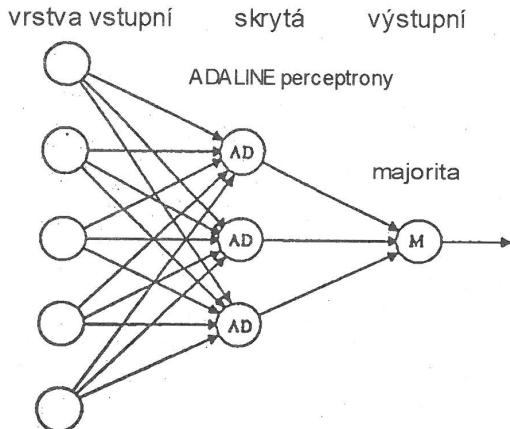
ADALINE

- Drobné vylepšení perceptronu, které se týká učícího algoritmu.
- Při učení se chyba nepočítá přímo z výstupu, ale z aktivace. Což znamená, že musím znát požadovanou aktivaci.



Vícevrstvá síť perceptronů – Madaline

- **MADALINE** – Multiple Adaptive Linear Element – síť s jednou skrytou vrstvou a jednou výstupní vrstvou.



- Výpočet odpovědi sítě na předložený vstupní vzor je jednoduchý.
- Přivedu na vstupy všech neuronů odpovídající hodnoty vstupů.
- Pro každý neuron přenásobím vstupní hodnoty vstupními vahami a sečtu. Tím získám aktivaci neuronu.
- Aktivaci transformuji aktivační funkcí na výstup.
- Když takto vypočítám výstupy všech neuronů, aplikuji majoritu (spočítám kolik neuronů zařadilo vzor do "třídy" 0 a kolik do "třídy" 1). A podle toho předpovím výstupní třídu sítě.

- Učení jen nastíním – předložím vstupní vzor a spočítám výstup sítě.
- Pokud se výstup shoduje s požadovaným, nedělám nic.
- Pokud se vypočítaný a skutečný výstup liší, musím změnit váhy "nějakého" neuronu, který předpovídá špatnou třídu, aby příště "změnil názor" a tím se majorita přiklonila na správnou stranu.
- A nejjednodušší je "přesvědčit" neuron, který má aktivaci nejbliž nule. Tomuto neuronu přepočítám váhy.

- Učení jen nastíním – předložím vstupní vzor a spočítám výstup sítě.
- Pokud se výstup shoduje s požadovaným, nedělám nic.
- Pokud se vypočítaný a skutečný výstup liší, musím změnit váhy "nějakého" neuronu, který předpovídá špatnou třídu, aby příště "změnil názor" a tím se majorita přiklonila na správnou stranu.
- A nejjednodušší je "přesvědčit" neuron, který má aktivaci nejbliž nule. Tomuto neuronu přepočítám váhy.

- Učení jen nastíním – předložím vstupní vzor a spočítám výstup sítě.
- Pokud se výstup shoduje s požadovaným, nedělám nic.
- Pokud se vypočítaný a skutečný výstup liší, musím změnit váhy "nějakého" neuronu, který předpovídá špatnou třídu, aby příště "změnil názor" a tím se majorita přiklonila na správnou stranu.
- A nejjednodušší je "přesvědčit" neuron, který má aktivaci nejbliž nule. Tomuto neuronu přepočítám váhy.

Použití MADALINE

- Je dnes spíše ilustrativní.
- Učící algoritmus nezvládá komplexní úpravy a nedokáže měnit váhy neuronů dostatečně flexibilně.
- Druhý a stejně závažný problém – MADALINE z principu fungování nemůže zvládnout lineárně neseparabilné problémy. Vymyslíte proč?
- Problém dělá ta majorita...

Použití MADALINE

- Je dnes spíše ilustrativní.
- Učící algoritmus nezvládá komplexní úpravy a nedokáže měnit váhy neuronů dostatečně flexibilně.
- Druhý a stejně závažný problém – MADALINE z principu fungování nemůže zvládnout lineárně neseparabilné problémy. Vymyslíte proč?
- Problém dělá ta majorita...

Back Propagation

- Neuronová síť typu back propagation představuje elegantní cestu, jak se vypořádat s problémy sítě MADALINE a zároveň zvládnout učení několika skrytých vrstev.
- Podle jména dochází k zpětnému šíření – konkrétně ke zpětnému šíření chyby.
- Nejedná se v pravém slova smyslu o jednu neuronovou síť, ale označuje se tímto pojmem libovolná síť, ve které se při učení šíří chyba z výstupu zpět na vstup. Můžete narazit i na jiné back propagation síť, než o které budu vyprávět já.
- BP jsou prakticky jediným typem sítí, který je znám mimo výzkumnou komunitu zabývající se NS.

Back Propagation

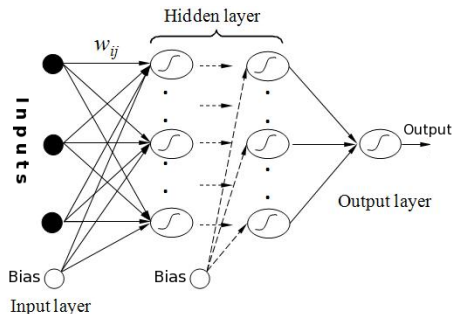
- Neuronová síť typu back propagation představuje elegantní cestu, jak se vypořádat s problémy sítě MADALINE a zároveň zvládnout učení několika skrytých vrstev.
- Podle jména dochází k zpětnému šíření – konkrétně ke zpětnému šíření chyby.
- Nejedná se v pravém slova smyslu o jednu neuronovou síť, ale označuje se tímto pojmem libovolná síť, ve které se při učení šíří chyba z výstupu zpět na vstup. Můžete narazit i na jiné back propagation sítě, než o které budu vyprávět já.
- BP jsou prakticky jediným typem sítí, který je znám mimo výzkumnou komunitu zabývající se NS.

Back Propagation

- Neuronová síť typu back propagation představuje elegantní cestu, jak se vypořádat s problémy sítě MADALINE a zároveň zvládnout učení několika skrytých vrstev.
- Podle jména dochází k zpětnému šíření – konkrétně ke zpětnému šíření chyby.
- Nejedná se v pravém slova smyslu o jednu neuronovou síť, ale označuje se tímto pojmem libovolná síť, ve které se při učení šíří chyba z výstupu zpět na vstup. Můžete narazit i na jiné back propagation síť, než o které budu vyprávět já.
- BP jsou prakticky jediným typem sítí, který je znám mimo výzkumnou komunitu zabývající se NS.

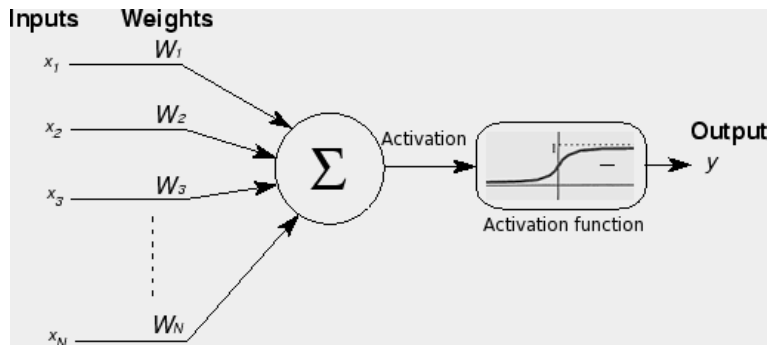
Struktura Back Propagation

- Jedná se o dopřednou síť s jednou nebo dvěma skrytými vrstvami.
- Vrstva se skládá z neuronů – jejich počet v jednotlivých vrstvách (a počet vrstev) je jedním z parametrů, který musíte určit experimentálně – tj. zkusit.
- Existuje celá teorie, která dokazuje, že k aproximaci libovolné funkce (libovolné rozhodovací hranice) stačí dvě skryté vrstvy.

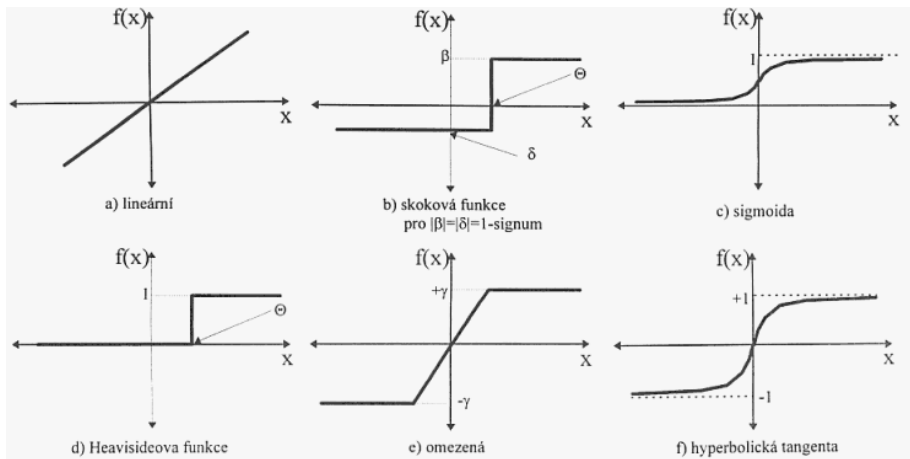


Back Propagation neuron

- Jedná se o standardní Perceptron se spojitou (!!)
- aktivační funkcí. Spojitost funkce budeme používat při učení – budeme počítat derivace.



Back Propagation aktivační funkce



Back Propagation učení (1)

- Učící algoritmus je trochu podobný tomu, co jsme již viděli.
- Nejprve se spočítá odpověď (výstup) sítě na předložený vstup, pak se spočítá rozdíl mezi vypočítaným a požadovaným výstupem.
- Co je úplně jiné – je výpočet změny vah.
- Chyba se totiž distribuuje zpět ke vstupům a postupně se upravují váhy ve skrytých vrstvách.
- A pak se pokračuje předložením dalšího vzoru, až do té doby než dosáhneme zastavujícího kritéria.

Back Propagation učení (1)

- Učící algoritmus je trochu podobný tomu, co jsme již viděli.
- Nejprve se spočítá odpověď (výstup) sítě na předložený vstup, pak se spočítá rozdíl mezi vypočítaným a požadovaným výstupem.
- Co je úplně jiné – je výpočet změny vah.
- Chyba se totiž distribuuje zpět ke vstupům a postupně se upravují váhy ve skrytých vrstvách.
- A pak se pokračuje předložením dalšího vzoru, až do té doby než dosáhneme zastavujícího kritéria.

Back Propagation učení (1)

- Učící algoritmus je trochu podobný tomu, co jsme již viděli.
- Nejprve se spočítá odpověď (výstup) sítě na předložený vstup, pak se spočítá rozdíl mezi vypočítaným a požadovaným výstupem.
- Co je úplně jiné – je výpočet změny vah.
- Chyba se totiž distribuuje zpět ke vstupům a postupně se upravují váhy ve skrytých vrstvách.
- A pak se pokračuje předložením dalšího vzoru, až do té doby než dosáhneme zastavujícího kritéria.

Back Propagation učení (2)

- 1 Inicializuj všechny váhy všech neuronů na náhodné hodnoty.
- 2 Vytvoř permutaci trénovací množiny.
- 3 Vyber další vzor x_i z vytvořené permutace.
- 4 Vypočítej výstup sítě y_i .
- 5 Spočítej chybu (rozdíl) mezi y_i a skutečnou hodnotou d_i .
- 6 Postupně propaguj chybu zpět a modifikuj váhy.
- 7 Pokud jsi dosud nepředložil všechny prvky z trénovací množiny pokračuj bodem 3.
- 8 Pokud chyba přes všechny trénovací vzory je větší než zvolené kritérium, pokračuj bodem 2.

Back Propagation učení (3)

- Výpočet výstupu sítě se provádí již známým způsobem.
- Pro každý neuron, u něž známe všechny vstupy spočítáme aktivaci.

$$a = \sum_{i=1}^n w_i x_i + \Theta$$

- Pokud znám aktivaci můžu spočítat výstup pomocí zvolené aktivační funkce.
- Pro, například, logistickou funkci tedy

$$y = \frac{1}{1 + e^{-\gamma a}}$$

- Takto postupuji pro všechny neurony, až zjistím výstup výstupního neuronu.

Back Propagation učení (4)

- Když zjistím výstup pro všechny vzory v trénovací množině, můžu zjistit globální chybu (energii) sítě.

$$E = \frac{1}{2} \sum_{i=1}^n (y_i - d_i)^2$$

- A snažíme se chybu (energii) minimalizovat. To se dá dělat různě a jednou z variant je gradientní sestup.
- Gradientní metoda mění váhy tak, aby energie klesla co možná nejmíň.

Back Propagation učení (4)

- Když zjistím výstup pro všechny vzory v trénovací množině, můžu zjistit globální chybu (energii) sítě.

$$E = \frac{1}{2} \sum_{i=1}^n (y_i - d_i)^2$$

- A snažíme se chybu (energii) minimalizovat. To se dá dělat různě a jednou z variant je gradientní sestup.
- Gradientní metoda mění váhy tak, aby energie klesla co možná nejvíc.

Back Propagation učení (4)

- "Na horách se chci dostat do údolí. Podívám se, kterým směrem se kopec nejvíc snižuje a udělám krok tím směrem. A zase se podívám, kam je to nejvíc z kopce a udělám krok. A tak dále, až jednou zjistím, že to dál nejde, čili jsem v údolí."
- V matematice existuje možnost, jak zjistit směr největšího vzestupu hodnoty funkce - gradient. To je první parciální derivace podle všech dimenzí. Když půjdu přesně opačně, půjdu ve směru největšího poklesu.

Back Propagation učení (4)

- "Na horách se chci dostat do údolí. Podívám se, kterým směrem se kopec nejvíc snižuje a udělám krok tím směrem. A zase se podívám, kam je to nejvíc z kopce a udělám krok. A tak dále, až jednou zjistím, že to dál nejde, čili jsem v údolí."
- V matematice existuje možnost, jak zjistit směr největšího vzestupu hodnoty funkce - gradient. To je první parciální derivace podle všech dimenzí. Když půjdu přesně opačně, půjdu ve směru největšího poklesu.

Back Propagation učení (5)

- Po chvíli derivování spočítáme (viz například skripta Miroslav Šnorek: "Neuronové sítě a neuropočítače"), dojdeme k následujícím vzorcům.

$$\delta_i^o(t) = (y_i^o - d_i)S'(\varphi)$$

- Kde $\delta_i^o(t)$ je požadovaná změna neuronu i ve výstupní vrstvě o při skutečném výstupu y_i oproti skutečnému výstupu d_i . S' je funkce inverzní k aktivační funkci a φ je aktivace neuronu.
- Změnu váhy mezi neuronem i ve vstupní vrstvě a neuronem j v první skryté vrstvě pak spočítám takto:

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

$$\Delta w_{ij}^o(t) = \eta \delta_i^o(t) y_j^h$$

- Kde y_j^h je výstup j -tého neuronu ve druhé skryté vstvě.

Back Propagation učení (6)

- Nový práh neuronu i ve výstupní vrstvě o pak spočítám takto:

$$\Theta_i^o(t+1) = \Theta_i^o(t) + \Delta\Theta_i^o(t)$$

$$\Delta\Theta_i^o(t) = \eta\delta_i^o(t)$$

Back Propagation učení (7)

- V druhé skryté vrstvě pak změnu vah vypočítám podle následujících vzorců:

$$\delta_i^h(t) = y_i^h(1 - y_i^h) \sum w_{ik}^o \delta_k^o$$

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

$$\Delta w_{ij}^h(t) = \eta \delta_i^h(t) y_j^{h-1}$$

$$\Theta_i^h(t+1) = \Theta_i^h(t) + \Delta \Theta_i^h(t)$$

$$\Delta \Theta_i^h(t) = \eta \delta_i^h(t)$$

- Kde w_{ij}^h e váha mezi neuronem i ve druhé skryté vrstvě a j v první skryté vrstvě. y_j^{h-1} je výstup j -tého neuronu v první skryté vrstvě.

Back Propagation vylepšení trénovacího algoritmu

- Všechna vylepšení směřují k tomu, abych zvýšil pravděpodobnost toho, že (rychleji) najdu globální minimum energetické funkce.
- Lze si jednoduše představit situaci, kdy gradientní sestup – tak jak jsem o něm teď mluvil – skončí v lokálním extrému. (Všechny body ve vzdálenosti 1 krok jsou výše, ale rozhodně nejsem v globálním minimu).
- Setrvačnost – při hledání směru kroku v čase $t + 1$ zohledním nejen současný gradient, ale i směr a velikost minulého kroku.
- Restart – nejde o vylepšení učícího algoritmu v pravém smyslu slova, ale pokud se sestup energetické funkce zastaví, zapamatuji si výsledek a začnu znovu.
- Další podrobnosti viz. například skripta Miroslav Šnorek: Neuronové sítě a neuropočítače.

Back Propagation vylepšení trénovacího algoritmu

- Všechna vylepšení směřují k tomu, abych zvýšil pravděpodobnost toho, že (rychleji) najdu globální minimum energetické funkce.
- Lze si jednoduše představit situaci, kdy gradientní sestup – tak jak jsem o něm teď mluvil – skončí v lokálním extrému. (Všechny body ve vzdálenosti 1 krok jsou výše, ale rozhodně nejsem v globálním minimu).
- Setrvačnost – při hledání směru kroku v čase $t + 1$ zohledním nejen současný gradient, ale i směr a velikost minulého kroku.
- Restart – nejde o vylepšení učícího algoritmu v pravém smyslu slova, ale pokud se sestup energetické funkce zastaví, zapamatuji si výsledek a začnu znovu.
- Další podrobnosti viz. například skripta Miroslav Šnorek: Neuronové sítě a neuropočítače.

Back Propagation vylepšení trénovacího algoritmu

- Všechna vylepšení směřují k tomu, abych zvýšil pravděpodobnost toho, že (rychleji) najdu globální minimum energetické funkce.
- Lze si jednoduše představit situaci, kdy gradientní sestup – tak jak jsem o něm teď mluvil – skončí v lokálním extrému. (Všechny body ve vzdálenosti 1 krok jsou výše, ale rozhodně nejsem v globálním minimu).
- Setrvačnost – při hledání směru kroku v čase $t + 1$ zohledním nejen současný gradient, ale i směr a velikost minulého kroku.
- Restart – nejde o vylepšení učícího algoritmu v pravém smyslu slova, ale pokud se sestup energetické funkce zastaví, zapamatuji si výsledek a začnu znovu.
- Další podrobnosti viz. například skripta Miroslav Šnorek: Neuronové sítě a neuropočítače.

Další neuronové sítě

- Prezentované sítě nejsou zdaleka jediné, které existují. I když Back propagation je nejznámější, neznamená to, že je nejlepší, nejzajímavější a nejlépe fungující.
- Existují například sítě, které při učení mění nejen váhy, ale i svou strukturu – sítě GMDH a GAME. Navíc mohou používat úplně jiné aktivační funkce.
- Síť s jiným typem neuronů (a v každé vrstvě jiné) – RBF síť.
- Sítě s velkými počty neuronů, kde váha mezi neurony i a j není určena zapamatovanou hodnotou, ale nějakou funkcí $w(i, j)$ – HyperNEAT (umožňuje tvořit sítě s tisíci neuronů).
- Sítě pro shlukování – SOM.

Další neuronové sítě

- Prezentované sítě nejsou zdaleka jediné, které existují. I když Back propagation je nejznámější, neznamena to, že je nejlepší, nejzajímavější a nejlépe fungující.
- Existují například sítě, které při učení mění nejen váhy, ale i svou strukturu – sítě GMDH a GAME. Navíc mohou používat úplně jiné aktivační funkce.
- Síť s jiným typem neuronů (a v každé vrstvě jiné) – RBF síť.
- Sítě s velkými počty neuronů, kde váha mezi neurony i a j není určena zapamatovanou hodnotou, ale nějakou funkcí $w(i, j)$ – HyperNEAT (umožňuje tvořit sítě s tisíci neuronů).
- Sítě pro shlukování – SOM.

Použití neuronových sítí

- Klasifikace
- Regrese
- Shlukování
- Kompresce dat
- Filtrování
- Řízení

- Umělé neuronové sítě (stejně jako cokoliv) nejsou všemocné. A na některé problémy se nehodí nebo jsou zbytečně složité. Předtím než začnete zkoušet NS zkuste se zamyslet nad následujícími otázkami:
- Dokáží jednoduše popsat řešení problému? Pokud ano, je mnohem jednodušší držet se tradičního programování.
- Mám dostatečné množství ohodnocených dat v trénovací množině?
- Musím získat absolutně přesné hodnoty na výstupu?
- Potřebuji být schopen jednoduše vysvětlit funkci modelu?

- Umělé neuronové sítě (stejně jako cokoliv) nejsou všemocné. A na některé problémy se nehodí nebo jsou zbytečně složité. Předtím než začnete zkoušet NS zkuste se zamyslet nad následujícími otázkami:
- Dokáží jednoduše popsat řešení problému? Pokud ano, je mnohem jednodušší držet se tradičního programování.
- Mám dostatečné množství ohodnocených dat v trénovací množině?
- Musím získat absolutně přesné hodnoty na výstupu?
- Potřebuji být schopen jednoduše vysvětlit funkci modelu?

- Umělé neuronové sítě (stejně jako cokoliv) nejsou všemocné. A na některé problémy se nehodí nebo jsou zbytečně složité. Předtím než začnete zkoušet NS zkuste se zamyslet nad následujícími otázkami:
- Dokáží jednoduše popsat řešení problému? Pokud ano, je mnohem jednodušší držet se tradičního programování.
- Mám dostatečné množství ohodnocených dat v trénovací množině?
- Musím získat absolutně přesné hodnoty na výstupu?
- Potřebuji být schopen jednoduše vysvětlit funkci modelu?

- Umělé neuronové sítě (stejně jako cokoliv) nejsou všemocné. A na některé problémy se nehodí nebo jsou zbytečně složité. Předtím než začnete zkoušet NS zkuste se zamyslet nad následujícími otázkami:
- Dokáží jednoduše popsat řešení problému? Pokud ano, je mnohem jednodušší držet se tradičního programování.
- Mám dostatečné množství ohodnocených dat v trénovací množině?
- Musím získat absolutně přesné hodnoty na výstupu?
- Potřebuji být schopen jednoduše vysvětlit funkci modelu?

- <http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=212>
- <http://neuron.felk.cvut.cz/courseware/html/chapters.html>
- skripta – Miroslav Šnorek: Neuronové sítě a neuropočítače.
- série knih – Mařík a spol: Umělá inteligence
- <http://www.mindcreators.com/TableOfContents.htm>