Algoritmy a struktury neuropočítačů
ASN – P3

- **Algoritmus učení zpětného šíření šíření chyby (BPG) – modifikace**

Problémy při nevhodném výběru algoritmu učení:

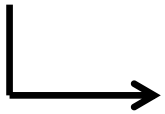
- **nevhodná volba parametrů učení**
- **nereprezentativní tréninková množina**
- **nevhodná inicializace vah a prahů**

→ **nevhodná volba rychlosti učení**
learning rate

oscilace – přeskočení malých lokálních minim

Modifikace – příklady:

momentové učení, adaptivní rychlost, optimalizace



- **rychlejší konvergence**
- **snížení pravděpodobnosti uvíznutí v tzv. lokálním minimu**
- **snížení citlivosti na detaily chybové funkce**
- **dovoluje sledovat trend chybové křivky**

Realizace momentu: přídavný člen v rovnici pro BPG

Změna vah a prahů: závisí na sumě podílu poslední a nové změny

mc

- **0 … změna vah je založena výhradně na gradientu**
- **1 … nová změna vah je rovna poslední změně vah (gradient je ignorován)**
- **0.6 - 0.95 … obvyklé hodnoty**

Doba trénování – sníží se pomocí adaptivního koeficientu učení
pomáhá udržet stabilitu učení

Rozdíl mezi základní metodou učení BPG a jejími modifikacemi

- **Základní učení využívá pouze adaptaci synaptických vah, prahy a parametry přenosových funkcí se nemění.**

- **Pouze jeden optimalizační parametr – rychlost učení (délka kroku)**

- **Modifikované metody dovolují adaptace prahů i sklon přenosových funkcí, používají více optimalizačních parametrů**

↓

rozhodují o výkonnosti procesu učení

- **Heuristické optimalizační metody – momentové učení, učení s proměnnou rychlostí učení, pružné učení (resilient)**

- **Numerické optimalizační metody – metoda konjugovaného gradientu, kvazi-Newtonova metoda, Levenberg-Marquardtův algoritmus**

BPG s momentem – MOBP *Back-Propagation with Momentum*

$$W_{i,j}(t+1) = W_{i,j}(t) + \alpha \sum_k \delta(k)\, y_j(k) + m_c \left[W_{i,j}(t) - W_{i,j}(t-1)\right] + \varepsilon(t)$$

α … 0 – 0.3 m_c **… 0.6 – 0.9**

náhodná hodnota šumu
užití – při uvíznutí v lokálním minimu

Poměr „nová chyba / stará chyba“ je větší, než maximální hodnota poměru chyb (obvykle 1.04 … tj. 4 %) ------- vyřazení momentu

Rychlé BPG

konverguje 10 – 100 krát rychleji

- **základ – stejný jako u klasického BPG učení**
- **poměr „nová chyba / stará chyba“ je větší, než maximální hodnota poměru chyb (obvykle 1.04 … tj. 4 %)**
- **nové váhy a prahy, výstup a chyba jsou zavrženy**
- **koeficient učení je snížen (obvykle vynásobením hodnotou 0.7) - *decrement***

- **výpočet nových vah a prahů, nového výstupu a nové chyby**
- **nová chyba je menší, než stará chyba**
- **koeficient učení je zvýšen (obvykle vynásobením hodnotou 1.05) - *increment***

batch režim … !!! vždy !!!

Proč proměnná rychlost učení ?

Velká rychlost … nestabilní učení, oscilace
Malá rychlost … pomalá konvergence

Rychlost učení se určuje experimentálně.

Proměnná rychlost učení VLBP

Variable Learning Rule
Adaptive Learning Rule

- **rychlejší než momentové učení**
- **dávkové učení – obvykle … vyžaduje více paměti**
- **robustní učení**
- **volba hodnot parametrů závisí na řešené problematice,**
- **ovlivňuje rychlost konvergence**
- **lze kombinovat momentové učení a učení s adaptivní rychlostí**

Pružné učení *(Resilient Backpropagation)*

Vícevrstvé NS - používají ve skrytých vrstvách aktivační funkc sigmoidálního typu … *squashing functions*

Funkce:

- **„stlačí“ nekonečný rozsah vstupních dat do konečného rozsahu výstupních dat**
- **pro velké vstupní hodnoty se sklon aktivační funkce blíží k 0 – malé změny vah a prahů (i když nejsou optimální)**
- **účelem pružného učení je odstranění těchto účinků hodnot parciálních derivací**
- **pro určení směru ukládaných vah se užívá pouze znamínko derivace**
- **velikost derivace nemá žádný vliv na uložení vah**
- **malé zvýšení požadavků na paměť**

Sdružené gradientní učení *(Conjugate Gradient Algorithms)*

- **Základní BPG algoritmus - nastavuje váhy ve směru negativního gradientu.**
- **Sdružený gradient - hledá směr s obecně rychlejší konvergencí.**

Existuje několik variant. Vhodné pro sítě s velkým počtem vah.
Pro různé aplikace je třeba volit různé varianty. Pracuje v dávkovém módu.

Quasi - Newton Algorithms

- **Newtonova metoda je alternativou k metodě sdruženého gradientu.**

- **Je založena na výpočtu druhých derivací (Hessovy matice).**
- **Vyznačuje se rychlejší optimalizací, ale obtížným výpočtem pro dopředné sítě.**
- **Quasi-Newtonovy metody jsou založeny na aproximaci Hessianovy matice v každé iteraci.**
- **Nové uložení vah je počítáno jako funkce gradientu.**

$$W_{i+1} = W_i - A_i^{-1} g_i$$

Tento algoritmus vyžaduje více výpočtů.

Aproximace Hessianu musí být uloženy, dimenze Hessianu je *n* x *n* (*n* se rovná počtu vah a prahů).

Pro velké sítě, může být lepší použít pružné učení (resilient) nebo jednu z variant sdruženého gradientu.

Pro menší sítě je možné použít jednu z variant Quasi-Newtonovy Metody.

Levenberg – Marquardtova aproximace

$$\Delta W = (J^T J + \mu T)^{-1} J^T e$$

pro malé sítě – cca několik tisíc vah a prahů, pro velkou přesnost, nejrychlejší metoda

J … Jacobian derivací chyb v závislosti na vahách
μ … skalární hodnota
e … vektor chyb
nevhodné pro rozpoznání, vhodné pro aproximaci funkcí

[HAM94] Hagan, M.T., Menhaj, M.: Training feedforward networks with the Marquardt algorithm. In.: IEEE Trans. On Neural networks, vol. 5, No.6, 1994

Metoda častého zastavování *Early stopping*

Data se dělí do tří skupin – tréninková, validační, testovací

Tréninková data – pro výpočet gradientu pro adaptaci vah a prahů
Validační data (kontrolní) – průběžné sledování chybové funkce (během celého tréninku)
ukončení procesu: v počátku stoupání chybové křivky
Testovací data – špatné rozdělení dat …. chyba dosahuje výrazně jiné hodnoty než pro validační data

Porovnání metod rychlého učení

1) algoritmus s adaptivním *lr* - obvykle mnohem pomalejší než další metody (může to být někdy užitečné - někdy je vhodná pomalá konvergence při použití "early stopping")

2) resilient BP - nejrychlejší algoritmus pro rozpoznávací problémy
- není vhodný pro aproximace funkcí
- požadavky na paměť jsou relativně malé v porovnání s dalšími algoritmy

3) sdružené gradientní algoritmy - jsou vhodné pro nejrůznější problémy (zvláště pro sítě s velkým množstvím vah)
- jsou téměř tak rychlé jako LM pro aproximaci funkcí (rychlejší pro velké články)
- pro rozpoznání jsou téměř tak rychlé jako resilient učení
- mají relativně skromné požadavky na paměť

4) LM - nejrychlejší konvergence pro aproximace funkcí
- pro velmi přesné trénování, jen pro malé sítě
- nehodí se pro rozpoznání
- potřeba velké paměti (redukce paměti možná za cenu zvýšen doby trénování)