Řešení úloh I.

Generování vektoru se dvěma shluky

- Napište funkci randv2n(n1,S1,R1,n2,S2,R2), která vygeneruje řádkový vektor obsahující n1 hodnot s normálním rozdělením se střední hodnotou S1 a rozptylem R1 a n2 hodnot s normálním rozdělením se střední hodnotou S2 a rozptylem R2. Vytvořte histogram vygenerovaných dat.

Řešení

- ```
  function v = randv2n(n1,S1,R1,n2,S2,R2)
    x = S1 + sqrt(R1)*randn(1,n1);
    y = S2 + sqrt(R2)*randn(1,n2);
    v = [x y];
  end
  v = randv2n(50,3,1,50,-1,0.5);
  hist(v,20);
  ```
  ```

Řešení úloh II.

Generování shluků ve 2D

- Napište funkci randv2D(p), která vygeneruje n shluků podle daného předpisu. p je matice $n \times 5$. $p(i,1)$ je počet hodnot se střední hodnotou $p(i,2)$ a a rozptylem $p(i,3)$ v první souřadnici a se střední hodnotou $p(i,4)$ a a rozptylem $p(i,5)$ v druhé souřadnici. Zobrazte vygenerované shluky do 2D grafu.

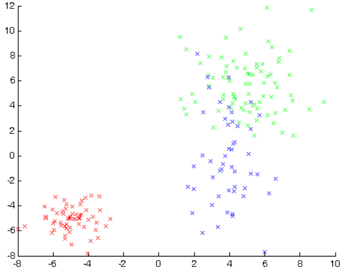
Řešení

- ```
  function v = randv2D(p)
    v = [];
    for i = 1 : size(p,1)
      n = p(i,1);
      Sx = p(i,2); Rx = p(i,3); Sy = p(i,4); Ry = p(i,5);
     y = [Sx + sqrt(Rx)*randn(1,n); Sy + sqrt(Ry)*randn(1,n)];
      v = [v y];
    end
  end
  ```
  ```

Řešení úloh II.

Generování shluků ve 2D - volání

- v=randv2D([60,-5,1,-5,1; 80,5,2,5,2; 50, 4,1,0,4]);
  hold on;
  plot(v(1,1:60),v(2,1:60),'rx');
  plot(v(1,61:140),v(2,61:140),'gx');
  plot(v(1,141:end),v(2,141:end),'bx');
  hold off;



Řešení úloh III.

Generování vzorků

- Navrhnětě funkci select(x,k), která z matice x náhodně vybere k řádků. Zobrazte původní a vybraná data do grafu rozdílnými barvami nebo značkami.
- Užitečná funkce *randperm(N)*

Řešení

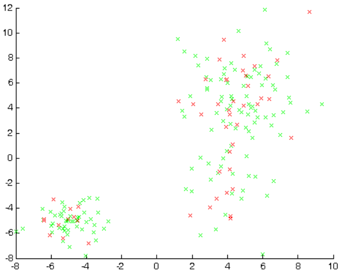
- function v = selectk(x,k)

```
function v = selectk(x,k)
  p = randperm(size(x,1));
  p = p(1,1:k);
  v = x(p,:);
end
```

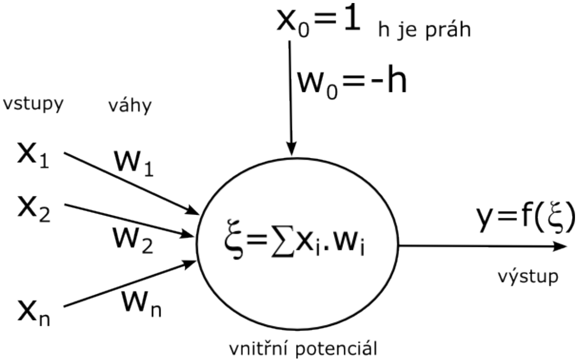
Řešení úloh III.

Generování vzorků - volání (pro 2D)

- ```
  v=randv2D([60,-5,1,-5,1; 80,5,2,5,2; 50, 4,1,0,4]);
  z =selectk(v',50)';
  hold on;
  plot(v(1,:),v(2,:),'gx');
  plot(z(1,:),z(2,:),'rx');
  hold off;
  ```


  ```

Formální (matematický) model neuronu

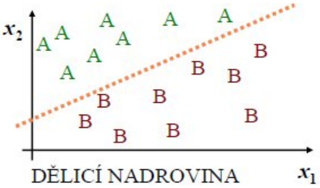


- vnitřní potenciál $\xi = \sum_{i=1}^{n} w_i.x_i - h = \sum_{i=0}^{n} w_i.x_i = \vec{w}\,.\,\vec{x}$

Neuron se skokovou přenosovou funkcí (perceptron)

hardlim, hardlims

- vnitřní potenciál $\xi = \sum_{i=1}^{n} w_i.x_i - h = \sum_{i=0}^{n} w_i.x_i$
- $f(\xi) = 1$ pro $\xi \geq 0$ … neuron je aktivní
- $f(\xi) = 0$ pro $\xi < 0$ … neuron je pasivní



- dělící nadrovina $x_2 = -\frac{w_1}{w_2}x_1 + \frac{h}{w_2}$
- $0 = w_1x_1 + w_2x_2 - h$

Lineární separabilita

Definice:
Množiny A,B jsou lineárně separabilní v n-rozměrném prostoru, pokud existují čísla $w_1, \dots, w_n, h$ taková, že pro každý bod $\vec{x} \in A$ platí $\sum_{i=1}^{n} w_i.x_i - h \geq 0$ a pro každý bod $\vec{x} \in B$ platí $\sum_{i=1}^{n} w_i.x_i - h < 0$

Pro Boolovský prostor:

- $n = 2 \to 14$ z $2^4 = 16$ Boolovských funkcí je lineárně separabilních.
  cvičení: Které dvě nejsou?
- $n = 3 \to 104$ z $2^8 = 256$
- $n = 4 \to 1882$ z $2^{16} = 65536$
- n obecné ... ??

Lineární separabilita

Definice:
Dělicí nadrovina určená (n+1)–rozměrným váhovým vektorem $\vec{w}$ je množina všech bodů $\vec{x} \in R^{n+1}$, pro které $\vec{w}.\vec{x} = 0$

Problém:
Nalézt takové váhy, resp. práh, které by umožnily separaci (oddělení) dvou množin vzorů (pomocí dělící nadroviny).

Možné řešení:
Perceptronový algoritmus učení

Perceptron - algoritmus učení

Máme k dispozici

- Trénovací množina T s trénovacími vzory (x^p, d^p).
  - $x^p = (x_1^p, ..., x_n^p)$ ... vstupní vzor
  - $d^p \in \{0, 1\}$ ... požadovaný výstup
- T můžeme rozdělit do dvou množin P a N:
  - P ... pozitivní vzory $(d^p = 1)$
  - N ... negativní vzory $(d^p = 0)$

Problém

- Nastavit váhy a práh neuronu tak, aby:
  - $\sum_{i=1}^n w_i.x_i - h < 0$ ... pro trénovací vzory z N
  - $\sum_{i=1}^n w_i.x_i - h > 0$ ... pro trénovací vzory z P

Perceptron- algoritmus učení

Značení:

- $y^p = F(x^p)$ ... skutečná odezva (výstup) neuronu pro vstupní vzor x^p

Cílová (chybová) funkce

- počet chybně klasifikovaných vzorů
- $E = \sum_{x \in P}(1 - F(x)) + \sum_{x \in N} F(x)$

Cíl učení

- Minimalizace E v prostoru vah. Nejlépe $E = 0$.

Perceptron- algoritmus učení

Myšlenka a odvození:

- ...

Perceptron - Algoritmus učení (Rosenblatt, 1959)

1. Inicializuj váhy a práh malými náhodnými hodnotami:
   $(w_1^0, ..., w_n^0)$ ... vektor vah v čase 0
   h^0 ... práh v čase 0
2. Předlož trénovací vzor (x^t, d^t):
   $x^t = (x_1^t, ..., x_n^t)$ ... vstupní vzor
   d^t ... požadovaný výstup
3. Spočti skutečný výstup (odezvu sítě):
   $y^t = sgn(\sum_{i=1}^n w_i^t . x_i^t - h^t)$
4. Adaptuj váhy:
   $w_i^{t+1} =$
   - w_i^t ... pokud $y^t = d^t$
   - $w_i^t + \alpha x_i^t$ ... pokud $y^t = 0, d^t = 1$
   - $w_i^t - \alpha x_i^t$ ... pokud $y^t = 1, d^t = 0$

   jinak napsáno $w_i^{t+1} = w_i^t + \alpha x_i^t (d_i^t - y_i^t)$
   α ... parametr učení
5. Pokud t nedosáhl maximální hodnoty, přejdi ke kroku 2.

Perceptron - Algoritmus učení

Jak správně inicializovat váhy?

- Heuristika: např. průměr vzorů z P - průměr vzorů z N

Jak ovlivní volba parametru učení výsledek?

- $\alpha \in < 0, 1 >$
- Nejlépe: α zpočátku velké, postupně $\alpha \to 0$

Jak zvolit počet adaptačních kroků?

Perceptron - Algoritmus učení

Výhody

- Triviální algoritmus
- Pro lineárně separabilní množiny algoritmus konverguje (nalezne řešení v konečném počtu kroků) *(Rosenblatt, 1959)*

Nevýhody

- Velmi pomalý algoritmus
- Umí klasifikovat jen lineárně separabilní množiny
- Chybí rozšíření pro více neuronů (vrstev)
- Špatné zobecňování

Přihrádkový algoritmus (Gallant, 1990)

Idea

- Použiji perceptronový algoritmus učení
- Nejlepší doposud nalezený vektor vah je v přihrádce
- Pokud najdu lepší váhový vektor, uložím ho do přihrádky

Výhody

- Pokud je trénovací množina konečná a složky váhového a příznakových vektorů jsou racionální, lze ukázat, že přihrádkový algoritmus konverguje k optimálnímu řešení s pravděpodobností 1.

Adaline (Widrow, 1959)

Neurony s lineární přenosovou funkcí

- $y^p = f(\xi^p) = \xi^p = \sum_{i=1}^{n} w_i.x_i^p - h$... skutečná odezva (výstup) neuronu pro vstupní vzor x^p

Cílová (chybová) funkce

- MSE ... střední hodnota čtverců chyb:
  pro jeden trénovací vzor:
  $E^p = \frac{1}{2}(d^p - y^p)^2 = \frac{1}{2}(d^p - \xi^p)^2 = \frac{1}{2}(d^p - \sum_{i=1}^{n} w_i.x_i^p - h)^2$
  pro celou trénovací množinu:
  $E = \frac{1}{N}\sum_p E^p = \frac{1}{2N}\sum_p (d^p - \xi^p)^2$

Adaline (Widrow, 1959)

Jiné adaptační pravidlo

- delta pravidlo ... aktualizace vah a prahu proti směru gradientu chybové funkce
- $w_i^{t+1} = w_i^t - \alpha \frac{\partial E^t}{\partial w_i^t}$
- $w_i^{t+1} = w_i^t + \alpha x_i^t (d_i^t - \xi_i^t)$

Výhody

- Jednoduchý algoritmus
- Může se učit průběžně (on-line)
- Predikce i klasifikace
  - model je ekvivalentní linární regresi

Cvičení - Perceptron v Matlabu

User Guide → Perceptrons

- % p ... vstupní vzory $n \times N$, t ... výstupní vzory $1 \times N$
  net = newp(p,t);
  % mohu nastavit váhy a práh na počáteční hodnotu:
  % net.IW{1,1}= [1 1]; net.b{1,1} = 0;
  % nastavení parametrů train: net.trainParam
  $[net1, tr]$ = train(net,p,t);
  % tr ... training report ... tr.perf
  % výstup (odezva) sítě
  y = sim(net1,p);
  % počet chybně klasifikovaných příkladů:
  e = sum(abs(y-t));

Strategie učení

Adaptace vah a prahů může probíhat dvojím způsobem:

- sekvenčně pro každý vzor zvlášť ... *trains*
- v cyklu pro každý vzor zvlášť ... *trainc*
- v cyklu najednou pro celou trénovací množinu ... *trainb* stabilnější

Pojmy:

- iterace = předložení jednoho trénovacího vzoru
- epocha (cyklus) = iterace přes celou trénovací množinu

Cvičení - Lineární neuron v Matlabu

User Guide → Linear Filters

- net = newlin(p,t);
  ...
  % počet chybně klasifikovaných příkladů:
  e = sum(abs(y-t)>0.5);

Cvičení - Perceptron a lineární neuron v Matlabu

- P = [2 1 -2 -1;2 -2 2 1];
  T = [0 1 0 1];
  net = newlin(P,T);
  net.trainParam.goal= 0.1;
  $[net1, tr1]$ = train(net,P,T);
  netp = newp(P,T);
  $[net1p, tr1p]$ = train(netp,P,T);
  Otázky:
- Čím se liší net a netp?
- Kolik bylo potřeba cyklů k naučení?
- Jaké jsou výstupy sítí po naučení a chyba klasifikace v obou případech?
- Zobrazte data a vytvořené dělicí nadroviny do grafu.

Perceptronová síť jako logický prahový obvod

Pomocí perceptronu lze realizovat základní logické funkce

- AND ... průnik konvexních útvarů
- OR ... sjednocení konvexních útvarů
- NOT, ID

Věta

Každá booleovská formule lze vyjádřit v disjunktně konjunktním tvaru, kde atomy tvoří literály nebo jejich negace.

- $F = K_1 v K_2 v ... v K_n$
- $K_i = A_{i1} AND A_{i2} AND ... AND A_{in_i}$
- $A_{ij} = L$ nebo $A_{ij} = notL$

→ Každou boolovskou funkci mohu vyjádřit pomocí perceptronové sítě (kolik vrstev stačí?).

Perceptronová síť jako logický prahový obvod

Cvičení 1

Navrhněte váhy a prahy neuronu pro realizaci binárních funkcí (pro binární a biparitní model):

- AND ... průnik konvexních útvarů
- OR ... sjednocení konvexních útvarů
- NOT, ID

Cvičení 2

- Navrhněte (co nejmenší) architekturu, váhy a prahy perceptronové sítě pro realizaci funkce XOR (pro binární a biparitní model)