1 Normalizácia dát

Máme vstupné dáta $X = \{x_1, x_2, \ldots, x_N\}$.

1.1 Min-max normalizácia na interval $< A, B >$

Napíšte funkciu `mmscale(x)` , ktorá transformuje vektor `x` tak, že jeho zložky lineárne namapuje na interval $< A, B >$. Typicky sa robí min-max normalizácia na interval $< 0, 1 >$ alebo na interval $< -1, 1 >$.

Napríklad

```
>> mmscale([0.12 3 -123],-1,1)
ans =

   0.95429   1.00000  -1.00000

>> mmscale([0.12 3 -123],0,1)
ans =

  0.97714  1.00000  0.00000
```

Pomocou príkazu `plot` zobrazte do jedného grafu `x` a `mmscale(x,A,B)` – napr. `x` na x-ovú a `mmscale(x,A,B)` na y-ovú osu.

1.2 Normalizácia podľa smerodajnej odchýlky

Smerodajná odchýlka (standard deviation) je funkcia

$$sd(X) = \sigma_X = \sqrt{\frac{\sum (X - \bar{X})^2}{N - 1}},$$

kde $\bar{X} = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i$ je stredná hodnota.

Táto normalizácia urobí takú transformáciu, aby dáta mali strednú hodnotu 0 a rozptyl 1.

Napíšte funkciu `sdscale(x)`, ktorá transformuje vektor `x` tak, že jeho zložky budú mať strednú hodnotu 0 a rozptyl 1.

Napríklad

```
>> sdscale([1 2 3])
ans =

        -1         0         1

>> sdscale([-2 2 7])
ans =

  -0.96099  -0.07392   1.03491
```

1.3 Sigmoidálna normalizácia

Sigmoida je funkcia

$$f(x) = \frac{1}{1+e^{-\lambda x}}, \text{ kde reálne číslo } \lambda \text{ sa nazýva } \textit{strmost'}.$$

Sigmoida má definičný obor $(-\infty, +\infty)$ a obor hodnôt $(0, 1)$.

Jej graf pre $\lambda = 1$ si môžeme zobraziť napr. príkazmi

```
>> x=-10:0.2:10;
>> plot(x,1./(1+exp(-x)))
```

Napíšte funkciu `sigmscale(x,l)`, ktorá transformuje prvky vektora `x` podľa sigmoidy so strmosťou `l`.

Napríklad

```
>> sigmscale(-3:3,2) ans =

  0.00247  0.01799  0.11920  0.50000  0.88080  0.98201  0.99753
```

Naprogramujte aj transformáciu `sigmscale_inv(x,l)` inverznú k transformácii `sigmscale(x,l)`.

2 Generovanie testovacích dát

Pre programovanie a testovanie rôznych metód učenia neurónových sietí je užitočné vedieť generovať dáta so zadanými vlastnosťami a tiež vedieť generovať "náhodné" dáta, resp. náhodne vyberať dáta z danej množiny.

Na generovanie náhodných dát je treba poznať rozloženie pravdepodobnosti (probability distribution) pre dáta:

(a) pre veličinu V s rovnomerným rozložením na intervale $< A, B >$ platí, že pravdepodobnosť, že $V = x$ je

$$P(V = x) = \begin{cases} \frac{1}{B-A} & \text{pre } x \in < A, B > \\ 0 & \text{inak} \end{cases} \quad (1)$$

Matica veľkosti $m \times n$ dát s rovnomerným rozložením (uniform distribution) z intervalu A, B sa vygeneruje príkazom

```
>> A=3; B=10;
>> m=3; n=4;
>> (B-A)*rand(m,n)+A
ans =

  3.31198  6.28767  7.37211  8.75329
  5.75145  9.82846  8.29843  3.21501
  7.42435  4.56690  5.99315  6.79303
```

(b) pre veličinu V s normálnym rozložením so strednou hodnotou μ a rozptylom σ^2 platí, že pravdepodobnosť, že $V = x$ je

$$P(V = x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Pre veličinu V platí, že s pravdepodobnosťou 95% bude jej hodnota z intervalu $< \mu - 2\sigma, \mu + 2\sigma >$.

Funkcia `randn` generuje náhodné hodnoty s normálnym rozložením so strednou hodnotou 0 a rozptylom 1 (tj. smerodajnou odchýlkou $\sqrt{1} = 1$). Matica veľkosti $m \times n$ dát s normálnym rozložením (normal distribution) so strednou hodnotou S a rozptylom R (tj. smerodajnou odchýlkou $\sqrt{\mathtt{R}}$) sa vygeneruje príkazom

```
>> S=2; R=3;
>> m=3; n=4;
>> sqrt(R)*randn(m,n)+S
ans =

   1.76953   1.02155   2.90496   3.54724
   1.81023   9.41794  -1.39530   3.38295
   4.53440  -2.64642   3.07368   0.76086
```

2.1 Generovanie vektora s dvomi zhlukmi

Napíšte funkciu `randv2n(n1,S1,R1,n2,S2,R2)`, ktorá vygeneruje vektor (riadkový) obsahujúci `n1` hodnôt s normálnym rozdelením so strednou hodnotou `S1` a rozptylom `R1` a ďalej `n2` hodnôt s normálnym rozdelením so strednou hodnotou `S2` a rozptylom `R2`.

Napríklad

```
>> randv2n(3,-10,1,4,10,1)
ans =

  1.0e+01  *

   0.86684   0.97688  -0.94214  -0.85999   1.09027   1.09225  -1.19626
```

Všimnite si, že vo vygenerovanom vektore sú čísla z oboch zhlukov náhodne zamiešané. Urobte histogram vygenerovaných dát. Ako sa dá nastaviť počet stĺpcov histogramu?

2.2 Generovanie zhlukov v 2D

Naprogramujte funkciu `randn2d(p)`, ktorá má ako parameter maticu `p` tvaru $n \times 5$. Táto funkcia vygeneruje maticu rozmerov $\sum_{i=1}^{n} \mathtt{p}(i) \times 2$ obsahujúcu náhodné body v 2D, pričom `p(`i`,1)` hodnôt je zo zhluku s normálnym rozdelením so strednou hodnotou `p(`i`,2)` v prvej súradnici a strednou hodnotou `p(`i`,3)` v druhej súradnici, rozptylom `p(`i`,4)` v prvej súradnici a rozptylom `p(`i`,5)` v druhej súradnici (pre $i = 1, \ldots, n$).

Napríklad:

```
>> rm=randn2d([6,-10,-10,1,1; 8,10,10,2,2])
rm =

  1.0e+01  *

  -0.81834  -1.08722
   1.24685   0.79264
  -0.92659  -0.99046
  -0.96725  -1.02791
   1.00210   1.14543
   1.05893   0.97251
   0.99114   1.02167
  -1.05398  -0.93580
   0.75075   1.27756
  -1.03174  -1.05069
   0.90385   0.86900
   0.95572   0.85197
   0.99392   0.95680
  -0.94696  -1.15904
```

Rozšírte funkciu o voliteľný druhý parameter. Ak bude funkcia `randn2d` zavolaná s dvomi parametrami, tak pred vrátením výsledkov vygenerované zhluky zobrazí do 2D grafu.

2.3 Generovanie vzorky z dát

Navrhnite funkciu `selectk(x,k)`, ktorá z matice `x` náhodne vyberie `k` stĺpcov.

Napríklad:

```
>> rm=rand(2,8)
rm =

  0.59672  0.25610  0.40015  0.22067  0.08944  0.20306  0.96964  0.80791
  0.48115  0.60608  0.13022  0.57005  0.33733  0.21467  0.57249  0.33636

>> srm=selectk(rm,4)
srm =

  0.22067  0.80791  0.25610  0.08944
  0.57005  0.33636  0.60608  0.33733
```

Zobrazte pôvodné a vybrané dáta do grafu rozdielnymi farbami alebo značkami.

3 Testovanie algoritmov učenia

Úloha: Nech X je množina, napr. všetky d-rozmerné vektory so zložkami z intervalu $< 0, 1 >$. Na množine X je definovaná cieľová funkcia $f : X \to \{0, 1\}$. Cieľom algoritmu učenia je nájsť funkciu $h : X \to \{0, 1\}$ z nejakej možnej množiny funkcií H takú, aby funkcia h čo najlepšie aproximovala funkciu f. Algoritmus učenia má

pritom k dispozícii iba vzorku dát $S \subset X$ (nazývanú trénovacia množina), spolu so správnou hodnotou cieľovej funkcie. Vzorka je n-prvková množina prvkov náhodne vybraných z X podľa pravdepodobnostného rozloženia D.

Keď algoritmus učenia vyberie nejakú funkciu h, tak ľahko určíme na koľkých prvkoch z trénovacej množiny naučená funkcia h dáva rovnakú hodnotu ako cieľová funkcia f. Chyba hypotézy h na trénovacej množine S je:

$$\mathsf{Error}_S(h) = \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x)) \ ,$$

kde n je počet prvkov v S a chyba klasifikácie je

$$\delta(f(x), h(x)) = \begin{cases} 1 & \text{pre } f(x) \neq h(x) \\ 0 & \text{pre } f(x) = h(x) \end{cases}$$

Takže

$$\mathsf{Error}_S(h) = \frac{r}{n} \ ,$$

kde r je počet prvkov z S, ktoré boli klasifikované nesprávne.

Problémy:

1. Ako odhadnúť správnosť h pre ďalšie vzorky vybrané z X podľa toho istého pravdepodobnostného rozloženia D?

2. Ako dobrý (presný) je tento odhad správnosti?

Nás zaujíma chyba hypotézy h na celej množine X, t.j. pravdepodobnosť chybnej klasifikácie prvku $x \in X$ vybraného podľa pravdepodobnostného rozloženia D:

$$\mathsf{Error}_D = Pr_{x \in D}[f(x) \neq h(x)] \ .$$

Takáto chyba má binomické rozloženie, takže odhad tejto chyby sa spočíta ako podiel počtu chybne klasifikovaných vzorov k počtu všetkých vzorov. Ale pozor, tento odhad je správny iba ak ho počítame na množine nezávislej na trénovacej množine S! Preto:

> **chyba algoritmu učenia sa musí počítať na testovacej množine T ($\subset X$) nezávislej (disjunktnej) na S.**

Odhad počtu chýb testovaného algoritmu učenia na testovacej množine T s $n_T = |T|$ prvkami je $\mathsf{Error}_T(h) = r_T$ – počet prvkov z T, ktoré boli klasifikované nesprávne. Takže chyba hypotézy h odhadnutá z testovacej množiny T je:

$$\mathsf{Error}_T(h) = \frac{r_T}{n_T} \ .$$

Rozptyl odhadovaného počtu chýb na trénovacej množine (podľa binomického rozdelenia) je:

$$\mathsf{Var}_T = n_T \cdot \frac{r_T}{n_T} \cdot \left(1 - \frac{r_T}{n_T}\right) = n_T \cdot \mathsf{Error}_T(h) \cdot (1 - \mathsf{Error}_T(h)) \ .$$

Smerodajná odchýlka počtu chýb na testovacej množine T je potom:

$$\sigma'_T = \sqrt{n_T \cdot \mathsf{Error}_T(h) \cdot (1 - \mathsf{Error}_T(h))} \ .$$

Smerodajná odchýlka odhadu chyby hypotézy h na testovacej množine T je

$$\sigma_T = \frac{\sigma'_T}{n_T} = \sqrt{\frac{\mathsf{Error}_T(h) \cdot (1 - \mathsf{Error}_T(h))}{n_T}} \ .$$

Pomocou Matlabu ľahko spočítame i interval, v ktorom je skutočná chyba hypotézy h na celej množine X so zadanou spoľahlivosťou α. So spoľahlivosťou α urobí hypotéza h maximálne `k=binoinv(`α`,`n_T`,`r_T/n_T`)` chýb na množine veľkosti n_T. Teda so spoľahlivosťou α je chyba hypotézy h na X v intervale $< 0, \mathtt{k}/n_T >$.[1]

Pre dostatočne veľké n_T (napr. $n_T \geq 30$, resp. $n_T \cdot \mathsf{Error}_T(h) \cdot (1 - \mathsf{Error}_T(h)) \geq 5$) sa binomické rozdelenie dá aproximovať normálnym rozložením so strednou hodnotou $\mathsf{Error}_T(h)$ a rozptylom $\sigma_T(h)$. Potom sa obojstranný interval spoľahlivosti P % počíta podľa vzorca

$$\mathsf{Error}_T(h) \pm z_P \cdot \sigma_T \ ,$$

kde hodnoty z_P sa nazývajú kvantily a dajú sa nájsť v tabuľkách. V Matlabe je možné z_P jednoducho dopočítať:

```
>> a=[80,90,95,98,99]
a =
    80    90    95    98    99
>> b=(100+a)/2/100
b =
    0.9000    0.9500    0.9750    0.9900    0.9950
>> norminv(b,0,1)
ans =
    1.2816    1.6449    1.9600    2.3263    2.5758
```

Zobrazte v jednom grafe binomické rozloženie pravdepodobnosti pre $n = 40$ prvkov s pravdepodobnosťou padnutia panny $p = 0.1$ (jedná sa o diskrétne pravdepodobnostné rozloženie, takže úlohou je nakresliť pravdepodobnostnú funkciu v bodoch $1, \ldots, n$) a normálne rozloženie s rovnakou strednou hodnotou a rovnakým rozptylom (jedná sa o spojité rozloženie, takže úlohou je nakresliť spojitú hustotu rozdelenia).

Všeobecný postup pre odvodenie intervalu spoľahlivosti

1. Identifikácia parametru p, ktorý je potrebné odhadnúť – chyba hypotézy $\mathsf{Error}_D(h)$.
2. Definícia odhadu Y – v našom prípade $\mathsf{Error}_S(h)$. Je vhodné voliť nestranný odhad s minimálnym rozptylom.
3. Určiť pravdepodobnostné rozloženie D_Y pre odhad Y, vrátane strednej hodnoty a rozptylu.
4. Určiť N %-ný interval spoľahlivosti – t.j. nájsť medze L a U tak, aby N % prípadov vybraných s podľa pravdepodobnostného rozloženia D_Y padlo medzi L a U.

[1] V Octave je k funkcii Matlabu `binoinv` ekvivalentná funkcia `binomial_inv`.

4 Programovanie algoritmov učenia

Nech $X \subseteq \mathbb{R}^d$ je množina d-rozmerných vektorov reálnych čísel a $f : X \to \{0, 1\}$ je cieľová funkcia. Cieľom algoritmu učenia je nájsť funkciu $h : X \to \{0, 1\}$ z nejakej možnej množiny funkcií H takú, aby funkcia h čo najlepšie aproximovala funkciu f. Algoritmus učenia má pritom k dispozícii iba vzorku dát $S \subset X$ (nazývanú trénovacia množina), spolu so správnou hodnotou cieľovej funkcie $f(x)$ pre každý prvok $x \in S$.

Funkcia h môže byť reprezentovaná ako funkcia dvoch premenných $h(x) = h(x, \theta)$, kde parameter θ je výsledkom učenia na vzorke $S \subset X$. Takže množina H funkcií, v ktorej sa hľadá funkcia h je

$$H = \{h(x, \theta) \mid \theta \in \Theta\} ,$$

kde Θ je množina všetkých možných hodnôt parametru θ pre funkciu h.

Učiaci algoritmus v MATLABe potom môže byť funkcia `learn(M,c)`, kde `M` je matica obsahujúca k stĺpcových d-rozmerných vektorov a `c` je riadkový vektor s k zložkami taký, že `c(i)` $= f$ `(M(:,i))`. Pre danú trénovaciu množinu `M` a požadované výstupy `c`, funkcia `learn` spočíta parameter `theta`, ktorý potom bude použitý pri volaní funkcie `h` realizujúcej naučenú funkciu.

Triviálny algoritmus učenia `triv_learn`, ktorý si iba zapamätá trénovaciu množinu môžeme implementovať ako

```
function theta = triv_learn(M,c)
    theta.M = M;
    theta.c = c;
```

Naprogramujte funkciu `triv_h(x,theta)`, ktorá realizuje nasledujúcu naučenú funkciu $h(x, \theta)$. Pre každý stĺpcový vektor `s` obsiahnutý v matici `x` vráti funkcia `triv_h(x,theta)` hodnotu 0 alebo 1 podľa nasledujúceho pravidla: Ak sa `s` zhoduje s `i`-tym vektorom trénovacej množiny (t.j. `theta.M(:,i)`), tak vráti odpoveď podľa trénovacieho vzoru (t.j. `theta.c(i)`), inak si náhodne zvolí odpoveď 0 alebo 1. Keď `x` obsahuje viac než jeden vektor, tak `triv_h(x,theta)` vráti riadkový vektor s odpoveďami odpovedajúcimi jednotlivým vektorom.

5 Perceptrón

Naprogramujte funkcie potrebné na simulovanie perceptrónu:

- `p = perc_create(n)` vytvorí a náhodne inicializuje perceptrón s `n` vstupmi. Perceptrón bude reprezentovaný rozšíreným váhovým vektorom — riadkovým vektorom dĺžky `n+1`, ktorého posledná zložka je prah s obráteným znamienkom.
- `c = perc_recall(p,x)` vráti výstup perceptrónu `p` pre vstupný (stĺpcový) vektor `x`.
- `pn=perc_update(p,x,c,lam)` urobí jeden krok učenia perceptrónu `p`, kde

  `x` je vstupný (stĺpcový) vektor,

  `c` je požadovaný výstup perceptrónu,

  `lam` je parameter učenia.

Funkciu `perc_update` naprogramujte tak, aby fungovala aj pre viacej vstupných vektorov naraz. Tj. ak `x` je matica obsahujúca k stĺpcových vektorov a `c` riadkový vektor s k prvkami, tak funkcia vykoná jeden krok učenia pre každý stĺpec matice `x`.

- `c = perc_err(p,x,c)` vráti chybu perceptrónu `p` pre vstupné (stĺpcové) vektory `x`, keď požadované výstupy sú `c` (riadkový vektor).
- `pn=perc_learn(p,x,c,lam,maxit)` urobí maximálne `maxit` epoch učenia perceptrónu `p`, kde

  `x` je matica vstupných (stĺpcových) vektorov,

  `c` je riadkový vektor požadovaných výstupov perceptrónu,

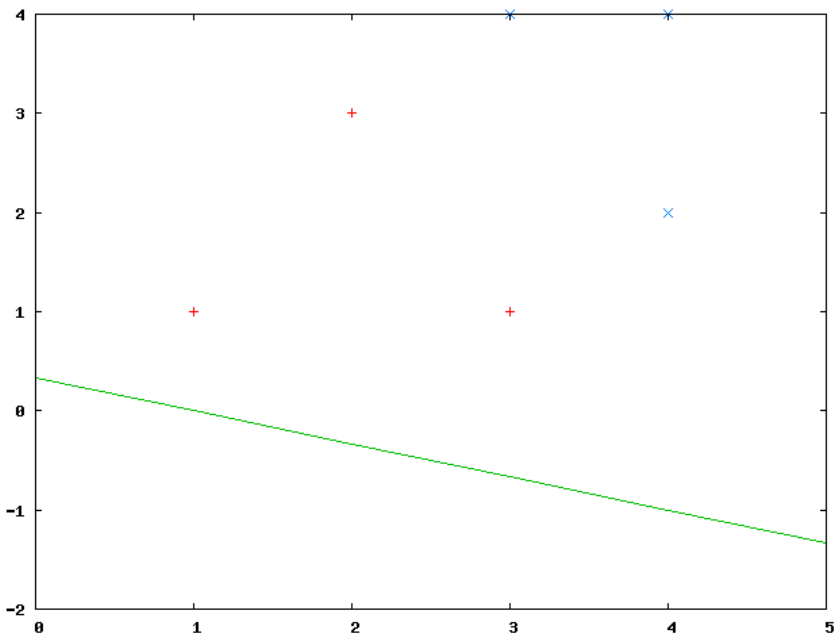
  `lam` je parameter učenia.

  Učenie končí, ak má perceptrón nulovú chybu alebo počet epoch učenia dosiahne `maxit`. Jedna epocha učenia spočíva v jednom vykonaní jedného kroku učenia pre každý vstupný vektor z `x`.

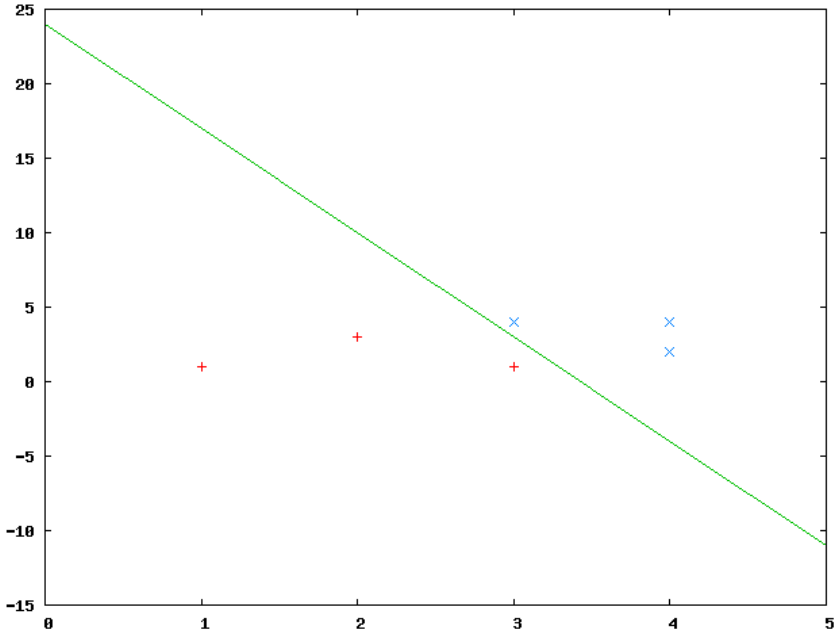
Vyskúšajte svoju implementáciu na nasledujúcom príklade:

```
p=[1 3 -1];                    %rucne inicializovany perceptron
xpos=[1 2 3; 1 3 1];           %trenovacie vstupy, pozadujeme odpoved 1
xneg=[3 4 4; 4 2 4];           %trenovacie vstupy, pozadujeme odpoved 0
x=[xpos xneg];
perm=[1 4 6 2 3 5];
x=x(:,perm)                    %trenovacie vzory zamiesame
c=[ones(1,3) zeros(1,3)];
c=c(perm)                      %pozadovane vystupy tiez zamiesame
xr=0:5;                        %vzory a perceptron zobrazime
plot(xpos(1,:),xpos(2,:),'or',xneg(1,:),xneg(2,:),'+b',xr,-(p(1)*xr+p(3))
  /p(2),'g')
pn=perc_learn(p,x,c,0.5,100)  %perceptron naucime a vysedok zobrazime
plot(xpos(1,:),xpos(2,:),'or',xneg(1,:),xneg(2,:),'+b',xr,-(pn(1)*xr+pn(3))
  /pn(2),'g')
```

Prvý príkaz `plot` nakreslí počiatočný stav perceptrónu.



Druhý príkaz `plot` nakreslí stav perceptrónu po naučení.



Aký je minimálny počet epoch potrebných na naučenie perceptrónu v tomto príklade tak, aby nerobil chyby?

6 Porovnávanie algoritmov učenia

Na porovnanie dvoch algoritmov učenia sa dajú použiť metódy porovnávania hypotéz, ale lepšie výsledky dáva párový test a ešte lepšie využíva trénovacie dáta tzv. k-násobná krížová validácia.

6.1 Porovnanie dvoch hypotéz

Hypotéza h_1 je testovaná na množine S_1 obsahujúcej n_1 prvkov, hypotéza h_2 je testovaná na množine S_2 obsahujúcej n_2 prvkov. Chceme odhadnúť rozdiel $d = \mathsf{Error}_D(h_1) - \mathsf{Error}_D(h_2)$ medzi skutočnými chybami hypotéz pri pravdepodobnostnom rozložení D. Podobne ako v sekcii 3 predpokladajme, že cieľová funkcia má diskrétny obor hodnôt a chyba hypotézy sa počíta ako podiel počtu chybne klasifikovaných vzorov a celkového počtu vzorov. Nestranný odhad rozdielu chýb je

$$\widehat{d} = \mathsf{Error}_{S_1}(h_1) - \mathsf{Error}_{S_2}(h_2) \ .$$

Problém je, že chyby hypotéz majú obecne nielen rôzne stredné hodnoty ale i rozptyly. Preto sa odhad rozdielu chýb aproximuje normálnym rozdelením so strednou hodnotou $\widehat{d}$ a rozptylom

$$\sigma^2_{\widehat{d}} = \frac{\mathsf{Error}_{S_1}(h_1) \cdot (1 - \mathsf{Error}_{S_1}(h_1))}{n_1} + \frac{\mathsf{Error}_{S_2}(h_2) \cdot (1 - \mathsf{Error}_{S_2}(h_2))}{n_2} \ .$$

Takže N %-ný interval spoľahlivosti odhadu $\widehat{d}$ je:

$$\widehat{d} \pm z_N \sqrt{\sigma^2_{\widehat{d}}} \ .$$

6.2 Párový test

Keď porovnávame dva algoritmy učenia, tak môže zmenšiť neurčitosť ich porovnania spôsobenú porovnaním výsledkov na rôznych trénovacích a testovacích množinách tým, že ich porovnáme pri učení z tej istej trénovacej množiny. Párové testy sa robia na identických testovacích množinách. Výsledné intervaly spoľahlivosti sú užšie, pretože rozdiely v pozorovaných chybách vznikajú kvôli rozdielom v algoritmoch, a nie kvôli rozdielom v použitých dátach.

Nech L_A a L_B sú dva algoritmy učenia tej istej cieľovej funkcie f. Učenie sa robí vždy z konečnej množiny trénovacích vzorov, preto by sme mali L_A a L_B porovnať pri učení na všetkých možných n-prvkových podmnožinách X pri pevnom pravdepodobnostnom rozložení D výberu prvkov. Stredná hodnota rozdielu chýb týchto dvoch algoritmov učených na takýchto n-prvkových podmnožinách X je

$$E_{S \subset D}[\mathsf{Error}_D(L_A(S)) - \mathsf{Error}_D(L_B(S))] \ ,$$

kde $L_\alpha(S)$ je hypotéza získaná pomocou algoritmu L_α učením na trénovacej množine S ($\alpha \in \{A, B\}$). Stredná hodnota sa počíta cez všetky n-prvkové podmnožiny vybrané z X podľa rozdelenia D.

V praxi je pre porovnanie metód učenia k dispozícii obmedzené množstvo trénovacích dát D_0. Preto sa D_0 náhodne rozdelí na trénovaciu množinu S_0 a trénovaciu množinu T_0, ktoré sú disjunktné. Trénovacie vzory sa použijú na učenie podľa L_A

a L_B. Testovacie vzory sa použijú na vyhodnotenie správnosti naučených hypotéz a dosiahnuté chyby sa odčítajú:

$$\mathsf{Error}_{T_0}(L_A(S_0)) - \mathsf{Error}_{T_0}(L_B(S_0)) \ .$$

Potom chyba $\mathsf{Error}_{T_0}(h)$ aproximuje chybu $\mathsf{Error}_D(h)$. Teda chyba sa počíta na jednej trénovacej množine S_0 a nie ako stredná hodnota rozdielu cez všetky možné vzorky S vybrané podľa rozdelenia D.

6.3 k-násobná krížová validácia

Pri dostatočnom počte trénovacích vzorov táto metóda umožňuje lepšie využitie týchto vzorov. Postupuje sa nasledovne:

1. Rozdelíme trénovacie dáta D_0 do k navzájom disjunktných množín $T_1, T_2, \ldots, T_k$ rovnakej veľkosti. Veľkosť T_i by mala byť aspoň 30.

2. Pre $i = 1, \ldots, k$ opakujeme:
   Vezmeme T_i ako testovaciu množinu, zvyšok dát dáme do trénovacej množiny
   $$\begin{aligned} S_i &:= D_0 \setminus T_i \\ h_A &:= L_A(S_i) \\ h_B &:= L_B(S_i) \\ \delta_i &:= \mathsf{Error}_{T_i}(h_A) - \mathsf{Error}_{T_i}(h_B) \end{aligned}$$

3. Výsledná hodnota rozdielu medzi chybami algoritmov L_A a L_B je

   $$\overline{\delta} = \frac{1}{k} \sum_{i=1}^{k} \delta_i \ .$$

   N %-ný interval spoľahlivosti je $\langle \overline{\delta} - t_{N,k-1} \cdot s_{\overline{\delta}},\ \ \overline{\delta} + t_{N,k-1} \cdot s_{\overline{\delta}} \rangle$, kde $s_{\overline{\delta}}$ je odhad smerodajnej odchýlky

   $$s_{\overline{\delta}} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \overline{\delta})^2}$$

   a $t_{N,k-1}$ je $\frac{N+100}{2}$ %-ný kvantil t-rozdelenia s $k-1$ stupňami voľnosti. V Matlabe sa $t_{N,k}$ dá spočítať ako `tinv((N+100)/200, k)`[2].

Navrhnite funkciu, ktorá porovná dva algoritmy učenia pomocou k-násobnej krížovej validácie.

[2] V Octave sa ekvivalentná funkcia nazýva `t_inv`.

Predpokladajte, že učiaci algoritmus je funkcia tvaru

LPar = *meno*(Tr,DTr,Par) ,

kde *meno* je meno funkcie, Tr sú trénovacie vstupy (stĺpcové vektory), DTr sú požadované výstupy (riadkový vektor) a Par sú parametre algoritmu učenia napríklad v tvare bunkového poľa. Takáto funkcia vráti naučené parametre LPar (napríklad tiež vo forme bunkového poľa). Ďalej budeme potrebovať funkciu, ktorá realizuje naučenú funkciu

Out = *menoL*(Par,In) ,

ktorá počíta naučenú funkciu s parametrami LPar pre vstupné vektory z matice In (každý stĺpec je jeden vstupný vektor). Out je riadkový vektor výsledkov.

Potom je možné urobiť funkciu Err(*Meno*,*MenoL*,Par,Tr,DTr,Ts,DTs), ktorá počíta chybu naučenej funkcie menoL učenej algoritmom Meno s parametrami Par z trénovacích dát Tr s požadovanými výstupmi DTr na testovacích vzoroch Ts s požadovanými výstupmi DTs. *Meno* a *MenoL* sú reťazce s názvami príslušných funkcií. Tie sa dajú vyhodnotiť pomocou funkcie feval.

Potom už je jednoduché naprogramovať funkciu

CrossVal(Meno1,Meno1L,Par1,Meno2,Meno2L,Par2,Pat,DOut,k),

ktorá spočíta rozdiel medzi chybou učiaceho algoritmu Meno1 s parametrami Par1 a chybou algoritmu Meno2 s parametrami Par2 pomocou k-násobnej krížovej validácie na vzoroch Pat s požadovanými výstupmi DOut.

Napríklad by sme mohli porovnať perceptrónový algoritmus, ktorý používa len 10 iterácií s perceptrónovým algoritmom, ktorý používa maximálne 1000 iterácií. Musíme upraviť učiacu a vybavovaciu funkciu perceptrónu:

```
function p = PLearn(x,c,Par)

   p = perc_learn(Par{1},x,c,Par{2},Par{3})

end

function Out = PRecall(Par,x)

   p = perc_recall(Par{1},x)

end
```

Potom už ich môžeme použiť vo funkcii CrossVall.

```
In = randn(2,200);
c = In(1,:)-3*In(2,:) >= 0.5
k = 5
Par1 = {[1 1 -1], 1, 10}
Par2 = {[1 1 -1], 1, 1000}

CrossVal('PLearn','PRecall',Par1,'PLearn','PRecall',Par2,In,c,5)
```