Vektorová kvantizace učením

Jedná se o **hybridní neuronovou síť** v tom smyslu, že **kombinuje učení bez učitele a učení s učitelem.**

Řeší klasifikační úlohy, ale může být použito i pro jednoduché rozpoznání.

Vektorová kvantizace učením (Learning Vector Quantization - LVQ) je společný název pro skupinu algoritmů LVQ1, OLVQ1, LVQ2, LVQ3. Toto učení je založeno na definici kvantizační oblasti mezi sousedními vektory kódové knihy. Je to obdoba **Voronoiových množin** u klasické vektorové kvantizace VQ. Hranice tříd jsou definovány jako úseky po částech lineární.

Úkolem je určení optimálního rozhodnutí o rozmístění tříd, o hranicích mezi nimi. Přístup k určení rozdělovacích (hraničních) ploch je u LVQ jiný, než u klasického přístupu v Bayesově teorii pravděpodobnosti. **Optimální hranice se určí odklonem všech váhových vektorů sítě, které leží mezi dvěma třídami a jejich přesunem blíž k jedné z nich.**
Není tedy nutné znát resp. počítat rozložení pravděpodobnosti.

Pro modelování fyziologických procesů se používá funkce tzv.**mexického klobouku (Mexican hat).**
Udává aktivitu neuronů v síti vůči vítěznému neuronu.

Než přistoupíme k podrobnějšímu vysvětlení jednotlivých variant učení LVQ, je vhodné se zmínit, v čem je výhoda LVQ a kdy je dobré dát v aplikacích přednost právě tomuto učení.

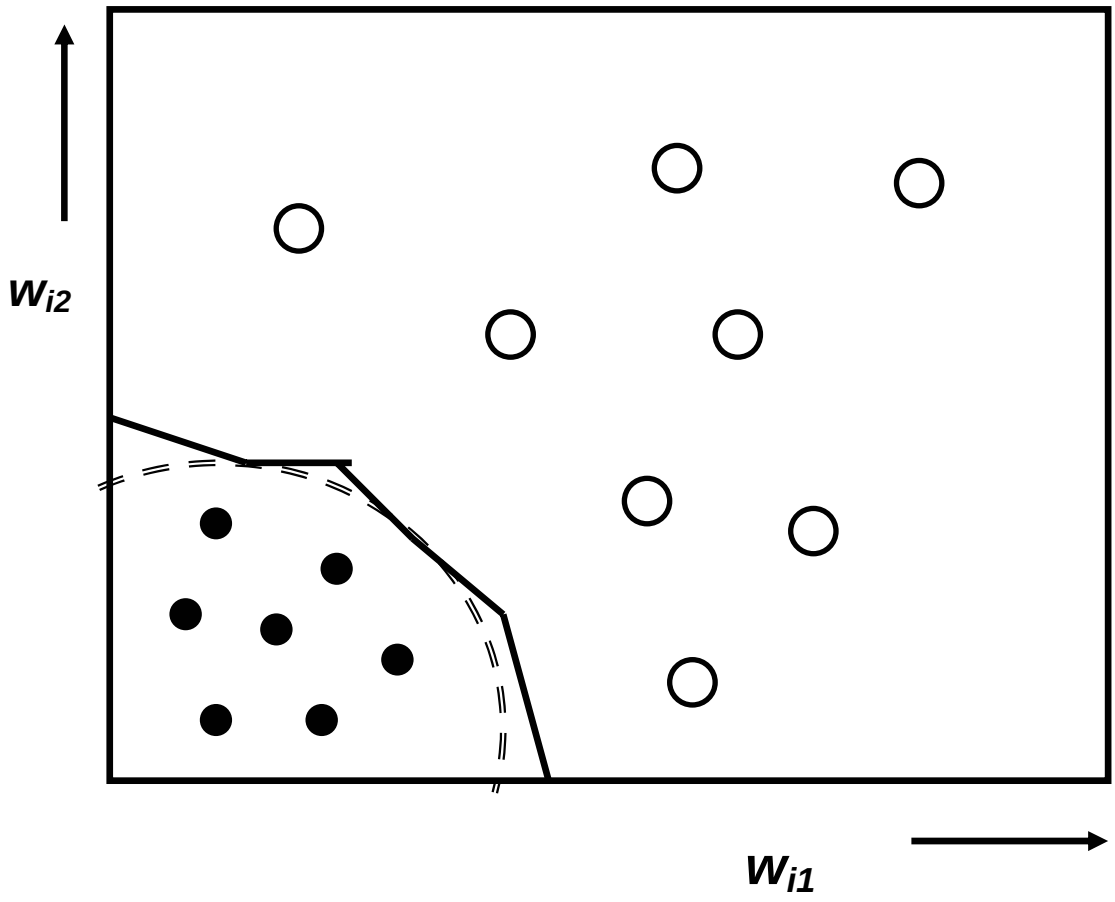
Pomocí jednotlivých variant LVQ je možné doladění váhových vektorů tak, aby se minimalizoval počet chybných klasifikací vzniklých v důsledku překrytí tříd. Jednotlivé varianty algoritmu LVQ se liší způsobem, jak nalézt co nejlepší hranice mezi třídami.

LVQ se používá pro kompresi dat pro přenos dat v digitálním kanálu, pro snížení počtu stavů obecně nebo pro možnost adaptivního rozšiřování počtu tříd.

V následujících řádcích si vysvětlíme **postup učení LVQ.**

1) Nejdříve vypočteme centroidy pomocí samoorganizace, např. pomocí KSOM. Tyto centroidy charakterizují pravděpodobné třídy.
2) Pak následuje krok, ve kterém jsou síti opětovně předloženy trénovací vzory s informací o jejich příslušnosti k třídě.
3) Třetím krokem je určení četnosti, s jakou je každý vektor sítě nejblíže k trénovacím vektorům každé třídy.
4) Nakonec je přiřazena třída, která se vyskytuje nejčastěji. Pokud vzor nelze zařadit do již existující třídy, vytvoří se třída nová.

Na obrázku klasifikace představují černé tečky referenční vektory z třídy S_1
prázdná kolečka referenční vektory z třídy S_2.
Plná čára je rozdělovací hranice určená podle LVQ, čárkovaná čára je Bayesova hranice.



Obr. Klasifikace do dvou tříd

Varianta LVQ1

Podstata učení zůstává stejná jako u základního LVQ. Jestliže W_i jsou kódové vektory označující jednotlivé třídy, pak vzorek x se umístí do stejné třídy, bude- li platit:

$$c = \arg\min_i \| \mathbf{X} - \mathbf{W_i} \|$$

Index pro nejbližší $\boldsymbol{W_i}$ k $\boldsymbol{X}$ je **index vítěze, centroidu** mezi $\boldsymbol{X}$ a všemi $\boldsymbol{W_i}$.

Je důležité si uvědomit, že je-li $\boldsymbol{X}$ přirozené číslo, náhodná nebo spojitá vektorová proměnná, nemusíme hledat další minima.

Pravděpodobnost pro $\|\boldsymbol{X} - \boldsymbol{W_i}\| = \|\boldsymbol{X} - \boldsymbol{W_j}\|$ pro $i \neq j$ je nulová.

$\boldsymbol{X}(t)$ je vstupní vzorek, $\boldsymbol{W_i(t)}$ je sekvence hodnot $\boldsymbol{W_i}$ v čase t = 0,1,2...Pak hodnoty $\boldsymbol{W_i}$ jsou asymptotické hodnoty v učebním procesu (minimalizují stupeň chybné klasifikace).

Pro LVQ1 definujeme následující algoritmus učení:

$\boldsymbol{W_i}(t+1) = \boldsymbol{W_c}(t)+g(t)[\boldsymbol{X}(t)-\boldsymbol{W_c}(t)]$, $\boldsymbol{X}$ a $\boldsymbol{W_c}$ patří do stejné třídy
$\boldsymbol{W_c}(t+1) = \boldsymbol{W_c}(t)-g(t)[\boldsymbol{X}(t)-\boldsymbol{W_c}(t)]$, $\boldsymbol{X}$ a $\boldsymbol{W_c}$ nepatří do stejné třídy
$\boldsymbol{W_i}(t+1) = \boldsymbol{W_i}(t)$, $i \neq c$

V těchto rovnicích je g(t) rychlost učení, pro níž platí $0< g(t)< 1$. Obvykle se g(t) volí menší než 0.1. Je třeba si uvědomit, že neexistuje pravidlo pro určení g = g(t).

Varianta OLVQ1

V této variantě je navíc optimalizovaná rychlost učení g(t) (pro každý kódový vektor je individuálně modifikována $g_i(t)$. Tuto optimalizaci nelze uplatnit pro LVQ2, protože proces by nekonvergoval, $g_i(t)$ se zde musí zmenšovat.

Algoritmus učení OLVQ1 je dán jako

$\boldsymbol{W_c}(t+1) = \boldsymbol{W_c}(t) + g_c(t)[\boldsymbol{X}(t)-\boldsymbol{W_c}(t)]$, je-li $\boldsymbol{X}$ klasifikováno korektně
$\boldsymbol{W_c}(t+1) = \boldsymbol{W_c}(t) - g_c(t)[X(t)-\boldsymbol{W_c}(t)]$, je-li **X** klasifikováno nekorektně
$\boldsymbol{W_i}(t+1) = \boldsymbol{W_i}(t)$, pro $i \neq c$

Pro rychlou konvergenci se určí $g_c(t)$ takto:

$\boldsymbol{W_c}(t+1) = [1–s(t)\, g_c(t)]\boldsymbol{W_c}(t)+s(t)\, g_c(t)\boldsymbol{X(t)}$
s(t) = +1 pro korektní třídu
s(t) = -1 pro nekorektní třídu

Rekursívní tvar pro určení optimální hodnoty je vyjádřen pomocí vztahu:

$g_c(t) = [g_c(t-1)] / [1+s(t)\, g_c(t-1)]\}$

Pro inicicializační hodnotu g(0) je dobré volit 0.3.

Varianta Batch LVQ1

$\boldsymbol{W_c}(t+1) = \boldsymbol{W_c}(t)+g(t)\, s(t)\, \delta_{ci}\, [\boldsymbol{X}(t)-\boldsymbol{W_c}(t)]$
s(t) = +1 pro $\boldsymbol{X}$ a $\boldsymbol{W_c}$ ze stejné třídy
s(t) = -1 pro $\boldsymbol{X}$ a $\mathbf{W_c}$ z různých tříd

V tomto vztahu δ_{ci} Kroneckerovo delta, $\delta_{ci} = 1$ pro c=i,
$\delta_{ci} = 0$ pro $c \neq i$

Pro každé *i* se určí nový referenční vektor ve tvaru

$\boldsymbol{W_i}^* = \sum_{t'} s(t')\, \boldsymbol{X}(t') / = \sum_{t'} s(t')$

kde t' jsou vzorky v uzlu *i*.

Varianta LVQ2 a LVQ2.1

V této variantě učení dochází k redukci počtu bodů rozložení w_i v blízkosti hraničních ploch.

Rozdělení do tříd je stejné, jako u LVQ1, ale při učení existují 2 kódové knihy $\boldsymbol{W_i}$ a $\boldsymbol{W_j}$.

Tyto třídy se nacházejí ve vektorovém prostoru blízko sebe. Vektor X se musí klasifikovat nejen do správné třídy, ale současně musí patřit do oblasti hodnot označených okénkem tak, že musí platit

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s \qquad\qquad s = \frac{1 - win}{1 + win}$$

kde d_i resp. d_j jsou Euklideovské vzdálenosti $\boldsymbol{X}$ od $\boldsymbol{W_i}$ a $\boldsymbol{W_j}$, *win* je relativní šířka okénka. Obvykle bývá v rozmezí $0.2 < win < 0.3$ (určuje se experimentálně).

Verze LVQ2.1 se liší od LVQ2 tím, že dovoluje, aby buď W_i nebo $\boldsymbol{W_j}$ byly uzavřené kódové knihy (v LVQ2 to platilo pouze pro jednu z nich).Algoritmus učení popisují následující vztahy:

$\boldsymbol{W_c}(t+1) = \boldsymbol{W_c}(t) + g(t)[\boldsymbol{X}(t) - \boldsymbol{W_c}(t)],\ \boldsymbol{X}(t) \in B_k,\ \boldsymbol{X}(t) \in S_k$
$\boldsymbol{W_c}(t+1) = \boldsymbol{W_c}(t) - g(t)[X(t) - \boldsymbol{W_c}(t)],\ \boldsymbol{X}(t) \in B_k,\ \boldsymbol{X}(t) \in S_r$
$\boldsymbol{W_i}(t+1) = \boldsymbol{W_i}(t),\ i \neq c$

kde B_k představuje Bayesovskou třídu. Ke korekci dochází jen pro $\boldsymbol{X}(t)$ z okna na špatné straně poloroviny.

Varianta LVQ3

Tato varianta hledá optimální umístění kódového vektoru (na rozdíl od LVQ2, která vytváří pohyblivé hranice vzhledem k Bayesovským limitám).

$\boldsymbol{W_i}(t+1) = \boldsymbol{W_i}(t) + g(t)[\boldsymbol{X}(t) - \boldsymbol{W_j}(t)],\ \boldsymbol{X}(t) \in B_k,\ \boldsymbol{X}(t) \in S_k,\ \boldsymbol{X}(t) \in win$
$\boldsymbol{W_j}(t+1) = \boldsymbol{W_j}(t) - g(t)[X(t) - \boldsymbol{W_j}(t)],\ \boldsymbol{X}(t) \in B_k,\ \boldsymbol{X}(t) \in S_r,\ \boldsymbol{X}(t) \in win$
$\boldsymbol{W_k}(t+1) = \boldsymbol{W_k}(t) - \varepsilon g(t)[X(t) - \boldsymbol{W_k}(t)],\ k \in \{i, j\}$

kde $\boldsymbol{X}(t)$, $\boldsymbol{W_i}$, $\boldsymbol{W_j}$ patří dostejné třídy. V této rovnici představuje B_k Bayesovskou třídu a *win* je šířka okénka. Z experimentů plyne: $0.1 < \varepsilon < 0.5$ pro $win = 0.2$ resp. $win = 0.3$. Optimální hodnota ε závisí přímo úměrně na šířce okénka. Optimální umístění kódových vektorů se během trénování nemění.

Jednotlivé varianty LVQ1, LVQ2 a LVQ3 se mezi sebou liší nejenom v matematickém zápisu rovnic, ale i svými vlastnostmi. LVQ1 a LVQ3 jsou robustnější procesy, pro LVQ1 je možné optimalizovat $g(t)$, abychom docílili rychlejší konvergenci. LVQ 2 optimalizuje relativní vzdálenost kódových vektorů od hranic tříd, není však garantováno jejich optimální umístění.

Obecné podmínky

U všech variant LVQ se definují hranice tříd podle pravidla nejbližšího okolí (není třeba znát funkci rozložení vzorků jako u klasické vektorové kvantizace VQ). Přesnost klasifikace záleží na:

- přibližně optimálním počtu vektorů kódové knihy přiřazených k jednotlivým třídám
- na jejich inicializaci
- na použitém algoritmu
- na vhodném *g(t)*
- na vhodném kritériu ukončení učení

Inicializace kódové knihy

Důležitým krokem při učení je inicializace kódové knihy. Jsou-li vektory kódové knihy rozděleny do tříd symetricky, pak průměrná hodnota nejmenší vzdálenosti vektorů kódové knihy bude stejná pro všechny vektory v jednotlivých třídách. Je tu však jeden problém. Konečné rozdělení kódových vektorů je totiž známé až po skončení učení. Proto je vhodné provést inicializaci kódové knihy pomocí SOM.

Učení

Při učení se doporučuje začít variantou LVQ1 nebo OLVQ1. Konvergence bude dosažena pro počet iterací rovný 30ti – 50ti násobku počtu kódových vektorů. V praktických aplikacích se nejčastěji volí OLVQ1, protože zrychluje učení. Ostatní varianty je možné navázat na LVQ1 resp. OLVQ1. Učení se ukončuje experimentálně.

Pokud budeme přihlížet k požadavku generalizace, je třeba ukončit učení asi 50 – 200 krát později, než odpovídá počtu kódových vektorů.