Least squares szupport vektor gépek adatbányászati alkalmazása

VALYON JÓZSEF, HORVÁTH GÁBOR

Budapesti Műszaki és Gazdaságtudományi Egyetem, Méréstechnika és Információs Rendszerek Tanszék
{valyon, horvath}@mit.bme.hu

Kulcsszavak: adatbányászat, Least Squares szupport vektor gépek, függvény approximáció, idősor előrejelzés

Napjainkban a számítástechnika, ezen belül is az adatbázisok és adattárházak elterjedt alkalmazásának köszönhetően az élet számos területén könnyedén nagy mennyiségű adat halmozható fel, melyek megfelelő feldolgozása, elemzése hasznosítható eredményekre vezethet. Az adatbányászat olyan eljárások és módszerek összessége, melyek segítségével feltárhatók felhalmozódott adatok közt rejlő, korábban ismeretlen összefüggések, rejtett trendek, szabályszerűségek. A klasszikus adatbányászat számos tudományterület, mint például a lineáris algebra, a gráfelmélet, az adatbázis elmélet, az algoritmus elmélet, a gépi tanulás, illetve a mesterséges intelligencia eszközeit felhasználja.

Cikkünkben a rendszerek, bemeneti és kimeneti adatai ismeretében történő fekete doboz modellezésével foglalkozunk, ahol a rendszer modelljét a rendelkezésre álló adatok alapján, ezek elemzésével kell meghatározni. Ezek általában nehezen algoritmizálható, specifikus megoldásokat igénylő feladatok, így gyakran intelligens rendszerek, illetve a „lágy" számítási módszerek (soft computing) kerülnek alkalmazásra. Kiemelten foglalkozunk a neurális hálózatok (Neural Networks – NN) egy különleges típusával, a szupport vektor gépek (Support Vector Machines – SVM) Least Squares változatával, az LS-SVM-mel, valamint bemutatjuk ennek egy módosított verzióját az LS^2-SVM-et. Az előbbi módszerek felhasználásával cikkünkben a függvény approximációra és idősor adatok elemzésére, illetve előjelzésére is mutatunk megoldásokat.

1. Bevezetés

Az élet számos területén találkozunk olyan komplex rendszerekkel, melyek működése, jellemzői, belső összefüggései nem ismertek. Ilyen problémákkal találkozhatunk például az orvosi diagnosztika, az ipari és a gazdasági folyamatok területén. Ezen rendszerek működése során azonban általában rengeteg bemeneti és kimeneti adat gyűjthető, melyek feldolgozásával információkat nyerhetünk, vagy megalkothatjuk a rendszer modelljét. Ezt fekete doboz (black-box) modellezésnek nevezzük.

Az adatbányászat feladata, hogy a leggyakrabban adatbázisokban, illetve adattárházakban felhalmozott adatok alapján olyan összefüggéseket, információkat tárjon fel, melyek közvetlenül nem állnak rendelkezésre.

Az adatbányászat eszközei között megtaláljuk a gépi tanulást, a különféle „soft computing" technikákat és ezen belül a neurális hálózatokat is [1,2]. A neurális hálózatok többek között általános eszközt adnak a (nemlineáris) regressziós feladat megoldására. Cikkünkben ezek egy speciális fajtájával a szupport vektor gépekkel (SVM) foglalkozunk [3,4], melyek a gépek rejtett rétegében elhelyezkedő neuronjaiban található bázisfüggvényekből állítják elő a kívánt függvényt. Az SVM-ek működésének lényege, hogy az eredeti megfogalmazásában még komplex nemlineáris megoldást igénylő feladatot, azaz a feladatból származó mintákat, nemlineáris transzformációk segítségével egy a bemeneti mintatér dimenziójánál több dimenziós térbe transzformálja, ahol az már lineárisan megoldható.

A módszer egyik legnagyobb előnye, hogy egy garantált felső korlátot ad az approximáció általánosítási hibájára. Egy másik fontos jellemzője, hogy a tanulási algoritmus törekszik a modell méretének minimalizálására (ritka modellt alkot), ami a hiba rovására történik, de mértéke egy paraméterrel szabályozható [9].

A hagyományos SVM alkalmazásának legnagyobb akadálya a módszer nagy algoritmikus komplexitása és a nagy memóriaigény, ami tipikusan a nagy adatmennyiség kezelését teszi lehetetlenné. A probléma megoldására számos megoldás született. Ezek az algoritmusok többnyire iteratív megoldások, melyek a nagy optimalizálási feladatot kisebb feladatok sorozatára bontják. Az egyes szeletelési „chunking" algoritmusok főként a feladat dekomponálás módjában – a részfeladatok meghatározásában – különböznek [5-7].

Az algoritmikus komplexitás legyőzésének egy másik lehetséges módja az LS-SVM alkalmazása, amikor is a számítási nehézségeket jelentő kvadratikus programozás helyett egy egyszerű mátrix inverziót használunk.

Az adatbányászati feladatokban általában nagy mennyiségű adattal kell dolgozni, ezért különösen fontos, hogy – ellentétben a hagyományos LS-SVM-el – a kiszámított modell komplexitása független legyen a tanítópontok számától. A következőkben az LS-SVM olyan kiterjesztésére adunk eljárásokat, amelyek lehetővé teszik, hogy egyszerűen, hatékonyan és kontrollálható módon hozhassunk létre ilyen modelleket.

A fekete doboz modellezési problémák körében általános feladatként két feladatcsoport jelölhető meg: (a) függvény approximáció (regresszió), illetve dinamikus rendszerek esetén (b) az idősor elemzése, előrejelzése.

Függvény approximáció esetén a rendelkezésre álló adatok közötti függvénykapcsolat elemzésére van lehetőség, azaz *N* adatból álló *p*-dimenzós mintakészlet esetén elemezhető, hogy a változók egyike (ez a rendszer kimenete, ami gyakran ismert) milyen függvénykapcsolatban áll a többi *p*-1 adattal (vagy annak bármilyen részhalmazával). Ebben az esetben a minták sorrendjének nem tulajdonítunk jelentőséget, feltételezzük, hogy a rendszer statikus, azaz kimenete csak a pillanatnyi bemenettől függ.

Idősor előrejelzés esetén feltételezzük, hogy a rendszernek van emlékezete (pl. visszacsatolás, memória), azaz dinamikus rendszer. Az előrejelzési feladatban a bemeneti és/vagy kimeneti adatok egymást követő értékei alapján próbálunk egy olyan modellt konstruálni, ami jól reprezentálja a folyamat dinamikáját, így képes folytatni az idősort (jósolni a későbbi kimeneti értékeket). Ez a feladat is általánosítható, azaz az előrejelzés nem csak az időtengely, hanem tetszőleges változó mentén lehetséges (például annak monoton növekvő értékeire). A rendszerek jelentős része valamilyen dinamikát mutat, azaz a folyamat pillanatnyi értéke nem csupán az adott bemeneti jeltől, hanem az előzményektől – a rendszer állapotától – is függ. Ebben az esetben is valójában egy regressziós feladatot (függvény approximációt) kell megoldani, de úgy, hogy a függvény bemeneti változói a korábbi értékekkel kibővülnek.

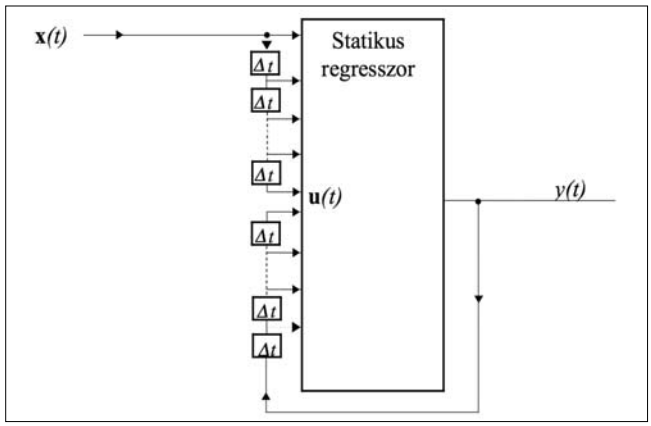
A következő rész bemutatja, hogy az idősor előrejelzés hogyan vezethető vissza a függvény approximációra. A 3. rész az LS-SVM regressziót és annak javasolt módosítását az LS-LS-SVM (LS^2-SVM) regressziót mutatja be, majd a cikk a 4. részben szereplő alkalmazási példák utáni áttekintéssel zárul.

2. Bemeneti mintahalmaz előállítása dinamikus rendszerekhez

Az előrejelzési feladat során, egy időben változó értéksorozat valahány elmúlt értékének (a folyamat régebbi mintavételi értékeinek) ismeretében a következő értéket vagy értékeket kell előre jelezni, jósolni (a későbbi mintavételi értékekre becslést adni).

Természetesen ilyen esetekben a feldolgozó algoritmusnak vagy modellező eljárásnak is a megfelelő időfüggést kell mutatnia. Ezt legegyszerűbben úgy érjük el, hogy a statikus feladatok megoldására kialakított (rendszerint nemlineáris) modellt egészítjük ki dinamikus (rendszerint lineáris) komponensekkel (legegyszerűbb esetben egyszerű késleltetéssel, visszacsatolással).

Leggyakrabban a tisztán statikus modellt „kívülről” egészítik ki dinamikus komponensekkel. Erre számos lehetőség van, de talán a legegyszerűbb és legelterjedtebb az *1. ábrán* látható.



1. ábra
Késleltető sorokkal dinamikussá tett statikus rendszer

Az ábrán látható statikus modell úgy tehető dinamikussá, hogy a bemenő adathalmazt késleltetett értékekkel bővítjük. Ehhez a bemeneti *N* dimenziós **x** vektorokból *N*+*K* dimenziós vektorokat állít elő úgy, hogy az *N* dimenzió mellé egyes bemenetek, illetve kimenet késleltetett értékét (értékeit) is felveszi.

$$\mathbf{u}^T(t) = [\, x_1(t), x_1(t-T_{11}), \ldots, x_1(t-T_{K_1 1}), \ldots, x_i(t), x_i(t-T_{1i}), \ldots, x_i(t-T_{K_i i}), \ldots, y(t-T_{1Y}), \ldots, y(t-T_{K_Y Y})\,] \quad (1)$$

ahol:

$x_i(t)$ az i-edik bemenet a t időpillanatban,

$x_i(t-T_{1i})$ az i-edik bemenet a t-T_{1i} időpillanatban,

$[T_{1i} \ldots T_{K_i i}]$ az i-edik bemenet késleltetéseinek halmaza (i=1...N)

K_i az i-edik bemenet késleltetéseinek száma.

$\mathbf{u}^T(t)$ az előállított N+K dimenziós vektor a t időpillanatban.

$[T_{1Y} \ldots T_{K_Y Y}]$ az y kimenet késleltetéseinek halmaza,

K_i az Y kimenet késleltetéseinek száma,

y a kimenet.

Az itt bemutatott megadás meglehetősen általános, hiszen minden bemenet esetén egyedileg és tetszőlegesen lehet megadni a késleltetéseket. Ezt a gyakorlati megoldásokban gyakran korlátozzák:

– nem bemenetenként szabályozható késleltetések,
– nem tetszőleges késleltetések, hanem
  például ablakméret megadás (utolsó *n* minta) stb.

Az így átalakított, kibővített adatsorra már a statikus regresszió alkalmazható, melynek megvalósítására a következő fejezetben mutatunk megoldást.

A tanítás után a becslés ugyanígy a késleltetett értékekkel kibővített bemenetre történik. Előrejelzés esetén iteratívan időlépésenként kell kiszámítani az eredményt, hiszen minden kimenetre szükség lehet a további kimenetek számításához.

3. Függvény approximáció

A regresszió, illetve a függvény approximáció megvalósítására számos különböző módszer áll rendelkezésre.

A módszerek több szempont szerint is rendszerezhetők, de a megoldható feladatok köre, illetve a megvalósítás szempontjából a következőket emeljük ki:

- Lineáris regresszió
- Nemlineáris regresszió

Ez a csoportosítás azért fontos, mert a későbbiekben bemutatásra kerülő bázisfüggvényes módszerek a nemlineáris regressziót a jóval egyszerűbb – hatékonyabban számítható – lineáris regresszióra vezetik visszsza. A feladat mindegyik esetben a következő:

Adott $\{\mathbf{x}_k, d_k\}_{k=1}^N$ tanító ponthalmaz, ahol $\mathbf{x}_k \in \Re^p$ egy *p*-dimenziós bemeneti vektor, illetve a $d_k \in \Re$ a kívánt kimenet. A cél egy $y = f(\mathbf{x})$ függvény megadása, ami jól reprezentálja a tanítópontok által leírt kapcsolatot.

Az ismert analitikus megoldások (lineáris/polinomiális regresszió, spline-ok) mellett a függvény approximációs feladat neurális hálózatokkal is megoldható [1,2]. Az alábbiakban bemutatott szupport vektor módszer a Least Squares SVM, az LS-SVM is ide sorolható [8].

Mint említettük az SVM-ek működésének lényege, hogy az elsődleges térben még komplex nemlineáris megoldást igénylő feladatot, azaz az ezt leíró mintákat, nemlineáris transzformációk segítségével egy magasabb dimenziós térbe transzformálják, ahol az már lineárisan megoldható. További tulajdonságuk, hogy bizonyos válogatást végeznek a rendelkezésre álló be-kimeneti mintapárok között. Olyan mintákat választanak ki – ezek a szupport vektorok – melyek a megoldás szempontjából relevánsak, míg a többit eldobják. Így valójában egy ritka (sparse) megoldást hoznak létre [9]. A least-squares megoldás (az LS-SVM) a tanulás gyorsítása érdekében éppen ezt a ritkasági tulajdonságot áldozza fel. A cikkben javasolt részleges redukciós eljárás (az LS^2-SVM) lehetővé teszi, hogy hasonlóan a hagyományos SVM megoldáshoz a modell mérete, illetve a hiba mértéke hangolható legyen, miközben az algoritmikus komplexitás tovább csökken [10,11].

3.1. LS-SVM regresszió

Az LS-SVM a hagyományos, Vapnik által bevezetett szupport vektor gépek (SVM) [3] egy módosított verziója, ami egy lineáris egyenletrendszer megoldására vezet, ellentétben a hagyományos módszerben alkalmazott idő és erőforrás-igényes kvadratikus programozással. A fő előnye ennek a módszernek, hogy a számítási komplexitás ezáltal csökken.

Az SVM-eknél a megoldást $y(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + b$ alakban keressük, ahol a $\varphi(.): \Re^n \to \Re^{n_h}$ egy nemlineáris leképezés egy többdimenziós térbe, ahol $n>n_h$, sőt sokszor $n>>n_h$. A bemenetet tehát egy magasabb dimenziós térbe transzformáljuk, majd itt a tanítás során kiszámított lineáris megoldás együtthatói alapján számítjuk az eredeti feladat megoldását.

A szupport vektor gépeknél ehhez egy további feltételt fogalmazunk meg ($\mathbf{w}$ hosszának, $\mathbf{w}^T\mathbf{w}$-nek minimalizálása), ami azt biztosítja, hogy a tanítópontokhoz minél jobban illeszkedő, de egyben minél simább megoldást nyerjünk, így az általánosítási hibát minimalizáljuk. A regresszió számításához az alábbi optimálási feladat írható fel (2):

$$\min_{w,b,e} J_p(\mathbf{w},e) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2 ,$$

a $d_k = \mathbf{w}^T\varphi(\mathbf{x}_k) + b + e_k$, ahol $k = 1,...,N$ feltételekkel.

A fenti egyenletekből az alábbi Lagrange multiplikátoros egyenlet írható fel (3):

$$L(\mathbf{w},b,e;\alpha) = J_p(\mathbf{w},e) - \sum_{k=1}^{N}\alpha_k\left\{\mathbf{w}^T\varphi(\mathbf{x}_k) + b + e_k - d_k\right\},$$

ahol az α_k értékek a Lagrange multiplikátorok.

A **w** és *e* kifejezése után a következő lineáris egyenletrendszer írható fel:

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^{\mathbf{T}} \\ \vec{\mathbf{1}} & \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix}\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix} \tag{4}$$

ahol:

$$\mathbf{d}^{\mathbf{T}} = [d_0, d_1, ..., d_N],\ \vec{\mathbf{1}}^{\mathbf{T}} = [1,...,1],\ \boldsymbol{\alpha}^{\mathbf{T}} = |$$
$$= [\alpha_0, \alpha_1, .., \alpha_N],\ \Omega_{i,j} = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j),$$
$$i, j = 1,..., N .$$

Az eredmény, hasonlóan a hagyományos SVM-hez a következő alakban írható fel:

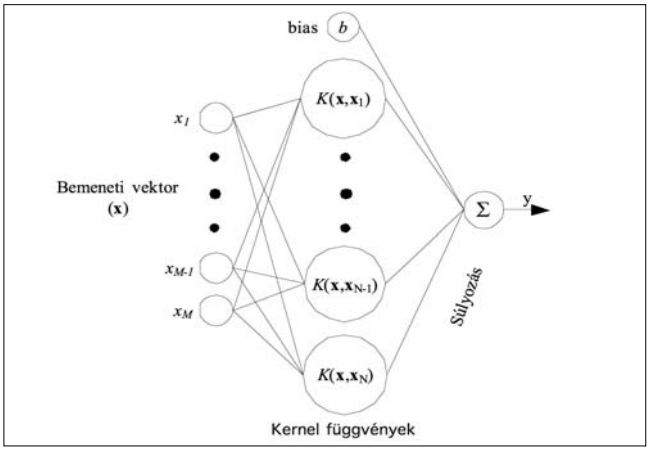
$$y(\mathbf{x}) = \sum_{k=1}^{N}\alpha_k K(\mathbf{x}, \mathbf{x}_k) + b \tag{5}$$

ahol az α_k és *b* a fenti egyenletrendszer megoldása.

Látható, hogy az eredeti $y(\mathbf{x}) = \sum_{i=1}^{N} w_i\varphi_i(\mathbf{x}) + b$ megoldáshoz hasonlóan egy lineáris megoldásra jutottunk, ahol a nemlineáris leképezést egy új, az eredeti φ(.) leképezések szorzatként előálló $K(.,\mathbf{x}_i)$ kernel függvénye váltja fel.

Bár a gyakorlatban az SVM-eket ritkán alkalmazzuk neurális hálózat alakjában, ez az értelmezés nagyon hasznos, mert egyszerűbb tárgyalásmódot tesz lehetővé a pusztán matematikai megközelítésénél. A szupport vektor gépek tanítása, és használata is matematikai számítások sorozata, a válasz számításának képlete pontosan megfeleltethető egy rejtett rétegű neurális hálózatnak *(2. ábra)*. A rejtett réteg tipikusan nemlineáris neuronokat tartalmaz.

2. ábra
A szupport vektor gépnek megfeleltethető neurális hálózat



A bemenet egy *M*-dimenziós vektor. A nemlineáris kernel függvényeket a rejtett réteg neuronjai tartalmazzák. Ezen neuronok száma megegyezik a szupport vektorok számával (LS-SVM esetén a ritkaság hiányában *N*, azaz a minták száma). A hálózat válasza (y) a rejtett réteg neuronjai kimenetének súlyozott összege. Az α_k súlyok a tanítás során kiszámított Lagrange multiplikátor értékek. Ennek megfelelően, minél kisebb a hálózat, annál kevesebb számításra van szükség a válasz megadásához, így a cél a lehető legkisebb hálózat elérése.

3.2. LS^2-SVM regresszió

Az LS2-SVM fő célja, hogy csökkentse a modell komplexitását, azaz a rejtett rétegbeli nemlineáris neuronok számát. A javasolt módszer egy járulékos előnye, hogy a modell számításának algoritmikus komplexitása is csökken. Az új megközelítés kiindulópontja a (4) képlet lineáris egyenletrendszere.

A javasolt módszer két fontos lépést tartalmaz:

(i) Az *első lépés* átalakítja az LS-SVM megoldást úgy, hogy az a tanítópontoknak csak egy részhalmazát használja „szupport vektornak'. Megmutatjuk, hogy az így kapott túlhatározott egyenletrendszer is megoldható.

(ii) A *második lépésben* a „szupport vektorok" (a kernel függvényeket meghatározó vektorok) automatikus meghatározására adunk módszert.

Túlhatározott egyenletrendszer

Ha a tanító készlet *N* mintapontot tartalmaz, akkor az egyenletrendszer (*N*+1) ismeretlent, az α-kat és a *b*-t, (*N*+1) egyenletet és (*N*+1)^2 együtthatót tartalmaz. Az együtthatók a $K(\mathbf{x}_i,\mathbf{x}_j)$ $i,j = 1,\ldots,N$ kernel függvény értékei. A mintapontok száma tehát meghatározza az egyenletrendszer méretét, ami egyúttal a megoldás komplexitását, a hálózat méretét. Hogy ritka megoldást, azaz egy kisebb modellt kapjunk, az egyenletrendszert, illetve az együttható-mátrix méretét redukálni kell.

Vizsgáljuk meg közelebbről az LS-SVM regressziós problémát leíró egyenletrendszert és jelentését.

Az első sor jelentése:

$$\sum_{k=1}^{N}\alpha_k = 0\,, \tag{6}$$

míg a *j*. sor a

$$b+\alpha_1 K\left(\mathbf{x}_j,\mathbf{x}_1\right)+\ldots+\alpha_k\left[K\left(\mathbf{x}_j,\mathbf{x}_k\right)+C^{-1}\mathbf{I}\right]+\ldots$$
$$\ldots+\alpha_N K\left(\mathbf{x}_j,\mathbf{x}_N\right)=d_j \tag{7}$$

feltételt, megkötést tartalmazza.

Az egyenletrendszer redukálásánál sorokat, illetve oszlopokat hagyhatunk el.

1) Ha a *j.* **oszlopot** hagyjuk el, akkor az ennek megfelelő α_j szintén „eltűnik", így az eredményként megkapott hálózat mérete csökken. Az első sor által támasztott feltétel azonban automatikusan alkalmazkodik, hisz a megmaradó α_k-k összege 0 marad.

2) Ha a *j.* **sort** töröljük, akkor az $\left(\mathbf{x}_j,d_j\right)$ tanító pontnak megfelelő információ elvész, hiszen a *j.* megkötést (7) elveszítjük.

Ezek alapján az egyenletrendszert leíró mátrix legfontosabb része $\mathbf{\Omega}+C^{-1}\mathbf{I}$ részmátrix, ahol $\mathbf{\Omega}$ az összes lehetséges tanító vektor kombinációját tartalmazza ($\Omega_{i,j}=K(\mathbf{x}_i,\mathbf{x}_j)$). A mátrix redukálása során abból sorokat, oszlopokat, illetve mindkettőt (sort a hozzá tartozó oszloppal) törölhetünk.

Minden oszlop egy-egy neuronnak felel meg, míg a sorok bemenet-kimenet relációkat, azaz a megoldás által teljesítendő feltételeket fogalmaznak meg. A mátrix az alábbi két módon redukálható:

Hagyományos teljes redukció – a $\left(\mathbf{x}_j,d_j\right)$ tanító pontot teljesen elhagyjuk, azaz mind a hozzá tartozó sort, mind pedig az oszlopot töröljük. Az alábbi egyenlet a tanító minták teljes elhagyásának hatását mutatja. A törölt elemek szürkével jelöltek.

$$\begin{bmatrix} 0 & \vec{1} \\ \vec{1} & \Omega+C^{-1}\mathbf{I} \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix}=\begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix} \tag{8}$$

A hagyományos metszési (pruning) technika esetében pontosan ez történik, hisz a metszési algoritmus iteratívan kihagyja a tanító pontok egy részét. Az elhagyott tanító pontok által hordozott információ ezért teljesen elvész. A tanítópontok által hordozott információ megőrzésére a részleges redukciót javasoljuk.

Részleges redukció – a $\left(\mathbf{x}_j,d_j\right)$ tanító mintát csak részben hagyjuk el, úgy, hogy töröljük a ponthoz tartozó oszlopot, de megtartjuk a hozzá tartozó sort. Így a tanító minta által hordozott megkötés továbbra is érvényben marad, hisz az adott sor súlyozott összegének amennyire csak lehet egyezni kell a d_j kívánt kimenettel.

Ha kiválasztunk *M*(*M*< *N*) *„szupport vektort"*, az egyenletrendszer túlhatározottá válik. A részleges redukció hatását a mutatja a (9) egyenlet, ahol az eltávolított részek szürkével jelöltek.

$$\begin{bmatrix} 0 & \vec{1} \\ \vec{1} & \Omega+C^{-1}\mathbf{I} \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix}=\begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix} \tag{9}$$

A fentiek alapján egy túlhatározott egyenlethez jutunk, amit négyzetes hiba-minimalizálással oldhatunk meg. Egyszerűsítsük jelöléseinket az alábbiak szerint:

$$\mathbf{A}=\begin{bmatrix} 0 & \vec{\mathbf{1}}^{\mathbf{T}} \\ \vec{\mathbf{1}} & \mathbf{\Omega}+C^{-1}\mathbf{I} \end{bmatrix},\quad \mathbf{u}=\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix},\quad \mathbf{v}=\begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix}. \tag{10}$$

A négyzetes hiba minimalizáló megoldás:

$$\mathbf{u}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{v}\,. \tag{11}$$

Az oszlopok törlése a sorok megtartása mellett biztosítja, hogy a neuronok száma csökkenjen, míg az összes ismert megkötést (mintapontot) figyelembe veszszük. Ez kulcsa annak, hogy a megoldás komplexitása a pontosság megtartása mellett csökkenthető legyen.

A módosított, részlegesen redukált egyenletrendszert least-squares értelemben oldjuk meg, ezért nevezzük ezt a módszert Least Squares LS-SVM-nek vagy röviden LS^2-SVM-nek.

Az itt bemutatott részleges redukció hasonlít a hagyományos SVM kiterjesztéseként bevezetett redukált szupport vektor gép (Reduced Support Vector Machine – RSVM) alapelvéhez [12]. Az RSVM esetén azonban, mivel az SVM eleve kisebb (ritka – sparse) modellt eredményez, ennek célja az algoritmikus komplexitás csökkentése, míg az LS-SVM esetén célunk a modell komplexitásának, a hálózat méretének csökkentése, azaz a ritka LS-SVM.

A kiválasztási eljárás

A fenti részleges redukció alkalmazása esetén szükség van valamilyen módszerre, ami meghatározza a szükséges szupport vektorokat. A kernel mátrix oszlopaiból kiválasztható egy olyan lineárisan független részhalmaz, melyek lineáris kombinációival a többi előállítható. Ez a kernel mátrix „bázisának" meghatározásával érhető el, hiszen az oszlopvektorok terének bázisa az a legkisebb vektorkészlet, amellyel a feladat megoldható. Az itt említett bázis csak egy bizonyos tolerancia értelmében határozható meg, hiszen célunk, hogy az N darab N dimenziós oszlopvektorból $M<N$ bázisvektort határozzunk meg.

A kiválasztási eljárás tartalmaz egy paramétert, ami kontrollálja a szükséges szupport vektorok számát (M). A szupport vektorok száma valójában nem függ a mintapontok számától (N), csak a probléma nehézségétől, hiszen M a mátrix lineárisan független oszlopainak száma. A gyakorlatban ez azt jelenti, hogy ha egy probléma nehézsége M neuront igényel, akkor a tanításhoz felhasznált mintapontok számától függetlenül a modell mérete nem növekszik.

A szupport vektorok kiválasztása az $\mathbf{A}^T$ mátrix reduced row echelon alakra hozása során részleges pivotolással végrehajtott Gauss-Jordan eliminációval történik [13,14]. A tolerancia figyelembevétele a pivot elem (p) ellenőrzésével történik. Eredetileg a pivotolás lényege, hogy az elimináció végrehajtása során szükséges osztásban a lehető legnagyobb elemet használjuk fel a numerikus stabilitás érdekében. A kiválasztott pivot elem nagysága azonban információt hordoz arról is, hogy mennyire fontos a hozzá tartozó oszlop a pontos megoldáshoz.

Ha $p \le \varepsilon'$ (ahol ε' a tolerancia paraméter), akkor az adott oszlop elhagyható, ellenkező esetben az oszlopnak megfelelő bemenet egy szupport vektor. A módszer azon oszlop vektorok listáját adja meg, melyek az *ε' toleranciaérték értelmében* lineárisan függetlenek.

A megfelelő tolerancia helyes meghatározása hasonló a többi hyper paraméter (C, ε, kernel paraméterek) megválasztási feladathoz. Egy lehetséges megoldás a kereszt kiértékelés (cross-validation) alkalmazása. Minél nagyobb a tolerancia, annál kisebb a hálózat, ellenben az approximáció hibája nő. Az ε' helyes megválasztása egy döntési feladat ahol a pontos approximáció illetve a modell komplexitása között kell dönteni.

Fontos hangsúlyozni, hogy 0 tolerancia esetén az LS^2-SVM és az LS-SVM megoldás azonos, mivel ebben az esetben a kiválasztási módszer a mátrix minden oszlopát, azaz minden tanító mintát megtartja.

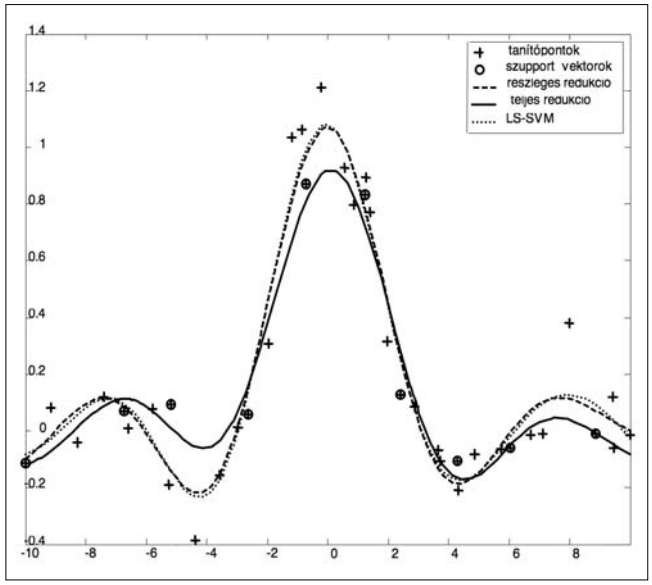
4. Vizsgálatok

A bemutatott algoritmusok értékeléséhez a szupport vektor regresszióval kapcsolatos irodalmakban legelterjedtebben használt benchmark feladatot a sinc(**x**) függvényt használjuk a [-10,10] tartományban. Kernel függvénynek a Gauss kernel kerül alkalmazásra $\sigma = \pi$ paraméterrel. A fent bemutatott ε' tolerancia értéke 0.2, valamint C = 100. A bementi mintákat 0.01 szórású normál eloszlású zaj terheli.

Elsőként a részleges redukció hatását kerül bemutatásra. Később ugyanezen problémában az automatikus szupport vektor kiválasztó algoritmus is felhasználásra kerül. Ugyanez a probléma kerül felhasználásra a hagyományos módszerekkel (LS-SVM, és Pruned LS-SVM) való összehasonlításhoz is, de egy összetettebb idősor előrejelzési feladat is bemutatásra kerül (Mackey-Glass kaotikus idősor előrejelzése).

A redukciós módszerek összehasonlításához először egy nagyon egyszerű szupport vektor kiválasztási módszert használunk. A 40 véletlenszerűen generált bemeneti mintapont közül minden negyediket szupport vektornak választunk. A *3. ábrán* a teljes és a részleges redukció eredménye együtt látható az eredeti, teljes LS-SVM-el.

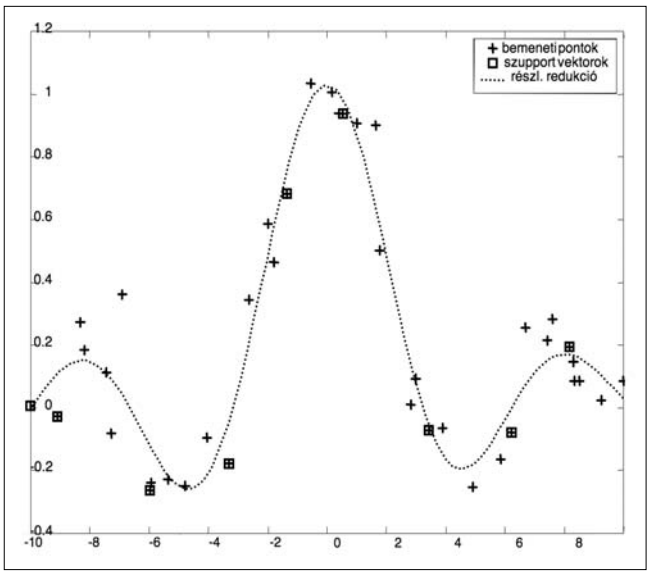
3. ábra
A különböző redukciós módszerek eredménye azonos szupport vektor készlet esetén



Látható, hogy a részleges redukció eredménye majdnem megegyezik az eredeti LS-SVM kimenetével, de a megoldás komplexitása a negyedére csökkent. Mivel a teljes redukció esetén az eredményt csak a szupport vektorok befolyásolják, ebben az esetben jóval rosszabb approximációt kapunk:

$$MSE_{partial\ red.}:1.04x10^{-3},$$
$$MSE_{full\ red.}:6.43x10^{-3},$$
$$MSE_{LS\text{-}SVM}:1.44x10^{-3}.$$

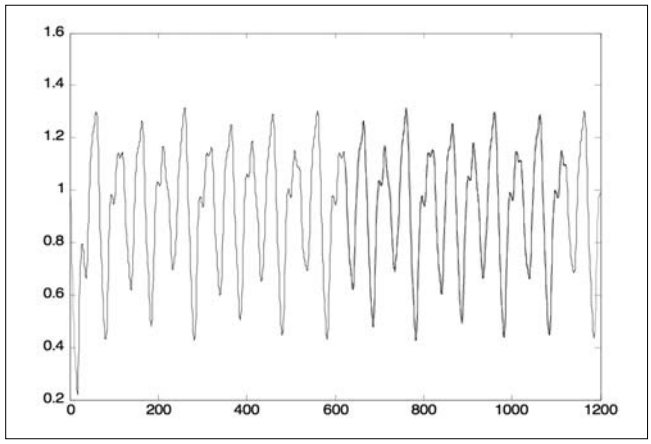
A bemutatott kiválasztási eljárással a szupport vektorok automatikusan is meghatározhatók. A *4.ábra* egy ilyen, a tanulás során automatikusan meghatározott szupport vektor készleten alapuló megoldást mutat be.



4. ábra
Egy részlegesen redukált LS-SVM, ahol a szupport vektorok a fent megadott kiválasztási algoritmussal kerültek meghatározásra (ε'= 0.2)

Az ábrán látható, hogy míg az eredeti hálózat 40 neuront tartalmazna, a redukált hálózat csak 9-et. A kiválasztási módszer még fontosabb tulajdonsága, hogy a meghatározott szupport vektor halmaz számossága független a tanító pontok számától. Ha a feladat *N* rejtett rétegbeli neuronnal megoldható, akkor a felhasznált tanító pontok számával a hálózat mérete nem kell hogy változzon.

A *táblázat (5. ábra)* ugyanazon feladat (sinc(**x**)) különböző méretű tanítópont készletei mellett kapott szupport vektorok számát, valamint az approximáció hibáját mutatja be. Az approximáció hibája egy 100 zajmentes mintából álló ellenőrző készleten számított négyzetes hiba. Látható, hogy a mintapontok számának növekedésével a hiba – a vártnak megfelelően – csökken, míg a hálózat mérete, azaz a megoldás komplexitása gyakorlatilag változatlan.



6. ábra
A Mackey-Glass idősor előrejelzési feladat megoldása LS-SVM-el

A *6. ábra* a széleskörűen elterjedt Mackey-Glass idősor előrejelzési probléma megoldását mutatja be. A Mackey-Glass idősor előrejelzésében az [-6,-12,-18,-24] késleltetéseket használtuk fel, azaz a t. időpillanat x(t) értékét négy korábbi érték alapján becsüljük (a Mackey-Glass folyamatban nincs külső bemenet, azaz a kimenet csak a korábbi értékektől függ). Az LS-SVM tanításához az idősor első feléből 500 mintát használtunk fel.

Az előrejelzés ellenőrzése során a következő értéket mindig a korábbi becsült idősor értékek alapján számítottuk. Egy másik megoldás, ha a későbbi becslések során a helyes korábbi értékeket használjuk fel. Az előbbi esetben a becslések hibája folyamatosan egyre nagyobb hibákat eredményez. Az utóbbi ellenőrzést akkor célszerű alkalmazni, ha a becslés során mindig csak a következő értéket kell meghatározni.

A fenti benchmark feladat alapján látható, hogy a bemutatott LS-SVM regresszión alapuló megoldás kiválóan alkalmas egy idősor előrejelzési probléma megoldására.

5. Összefoglalás

Cikkünkben a Least-Squares szupport vektor gépek adatbányászati alkalmasainak lehetőségeit vizsgáltuk. Bemutattuk a hagyományos LS-SVM-et, majd javaslatot tettünk egy olyan kiterjesztésre, ami lehetővé teszi,

5. ábra
A javasolt módszerrel elért szupport vektorok száma és a négyzetes hiba ugyanazon feladatra különböző tanítóhalmaz készletekkel (toleranciaérték: 0.25)

Tanító minták száma (A hagyományos LS-SVM esetén ez megegyezik a neuronok számával)	**A "szupport vektorok" száma** (A neuronok száma a javasolt módszerrel)	**Az approximáció négyzetes hibája** (A javasolt módszer alkalmazásával elért hiba)
40	8	$1.890x10^{-3}$
80	9	$0.877x10^{-3}$
800	9	$0.155x10^{-3}$
1600	9	$0.029x10^{-3}$

hogy nagy mennyiségű adat esetén is a rendszer egy egyszerű modelljét alkothassunk meg. Tartalmazza, hogy egy előrejelzési feladat általános esetben hogyan vezethető vissza egy függvény regressziós problémára, melynek megoldására egy tisztán analitikus és egy neurális modell is bemutatásra került.

Ezek a módszerek és eljárások felhasználhatók különböző orvosi adatok elemzésére is, s így jól használható eszközt jelentenek, mint diagnosztikai, mind pedig egyéb kutatási célokhoz.

Irodalom

[1] Horváth G. (szerk.):
„Neurális hálózatok és műszaki alkalmazásaik”,
Műegyetem kiadó, Budapest, 1998.

[2] S. Haykin:
„Neural networks. A comprehensive foundation”,
Prentice Hall, N. J., 1999.

[3] V. Vapnik:
„The Nature of Statistical Learning Theory”,
New York, Springer-Verlag, 1995.

[4] E. Osuna, R. Freund, F. Girosi:
„Support vector machines: Training and applications”,
Technical Report AIM-1602, MIT A.I. Lab., 1996.

[5] C. J. C. Burges, B. Schölkopf:
„Improving the accuracy and speed of
support vector learning machines”
In M. Mozer, M. Jordan and T. Petsche, editors,
Advances in Neural Information Processing Systems 9,
pp.375–381, Cambridge, MA, MIT Press, 1997.

[6] E. Osuna, R. Freund, F. Girosi:
„An improved training algorithm for
support vector machines”
In J. Principe, L. Gile, N. Morgan and E. Wilson, editors,
Neural Networks for Signal Processing VII –
Proceedings of the 1997 IEEE Workshop,
pp.276–285., New York, IEEE, 1997.

[7] Thorsten Joachims:
„Making Large-Scale SVM Learning Practical”,
Advances in Kernel Methods-SV Learning’,
MIT Press, Cambridge, USA, 1998.

[8] J.A.K. Suykens, T. Van Gestel, J. De Brabanter,
B. De Moor, J. Vandewalle:
„Least Squares Support Vector Machines”,
World Scientific, Singapore, 2002.

[9] F. Girosi:
„An equivalence between sparse approximation
and support vector machines,”
Neural Computation, 10(6), pp.1455–1480., 1998.

[10] J. Valyon, G. Horváth,
„A generalized LS-SVM”,
SYSID’2003 Rotterdam, 2003, pp.827–832.

[11] J. Valyon, G. Horváth,
„A Sparse Least Squares Support
Vector Machine Classifier”,
Proceedings of the International Joint Conference
on Neural Networks IJCNN 2004, pp.543–548.

[12] Yuh-Jye Lee, Olvi L. Mangasarian:
„RSVM: Reduced support vector machines”,
Proc. of the First SIAM International Conference
on Data Mining, Chicago, April 5-7, 2001.

[13] W. H. Press, S. A. Teukolsky,
W. T. Wetterling, B. P. Flannery:
„Numerical Recipes in C”,
Cambridge University Press, Books On-Line, 2002.
www.nr.com

[14] G. H. Golub, Charles F. Van Loan:
„Matrix Computations”,
Gene Johns Hopkins University Press, 1989.

Hírek

A **Sun Microsystems** a nyár folyamán új konszolidált tárolómegoldást alakított ki az OTP Banknál, amellyel a pénzintézet több üzleti területhez tartozó rendszert képes kiszolgálni. A heterogén rendszerek ellátására alkalmas, folyamatos rendelkezésre állást biztosító berendezéssel egyszerűsíthető és javítható az adatok felügyelete és védelme, valamint általa mérsékelni lehet az adatközpontok költségeit és bonyolultságát. Ezen megoldás a konszolidált rendszerek számára dinamikus háttérkapacitás-bővítést biztosít a jövőben, továbbá új rendszerek háttértár-konszolidációjának alapjait teremti meg. Az OTP Bank és a Sun folyamatos együttműködésének köszönhetően a rendszerek migrációja zökkenőmentes volt: az átállást minimális leállással, alacsony kockázati szinten sikerült megoldani.

A most bevezetett, jelentős bővíthetőségi tartalékkal rendelkező, nagy megbízhatóságú és magas rendelkezésre állást nyújtó – 16 terabájtnyi adat megóvására képes – tárolórendszer amellett, hogy a korábbi megoldásokhoz képest lényegesen magasabb szolgáltatási szintet biztosít, kedvező feltételekkel került értékesítésre. A színvonal emeléséhez többek között a nagyobb fokú adathozzáférési lehetőség és adatbiztonság, az emelt szintű teljesítmény, valamint több intelligens tárolóoldali szolgáltatás léte – például a pillanatfelvételi lehetőség, a távoli adatreplikáció, vagy az újgenerációs menedzsment framework – is hozzájárul. A kiépített rendszer további előnye még, hogy a pénzintézetnél használt rendszerek teljes – a Sun által támogatott, StorEdge-t, tárolóhálózatot (SAN-t) és szervereket tartalmazó – infrastruktúrája így egy forrásból kap támogatást.