INTELLIGENS RENDSZEREK I. LABORATÓRIUM

5. laborgyakorlat:

Tanuló ágensek

Készítette:

Valyon József
valyon@mit.bme.hu

Méréstechnika és Információs Rendszerek Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

2009, április.

Tartalomjegyzék

1 Követelmények

- A laborgyakorlat teljesítésére 4 órányi tiszta munkaidő áll rendelkezésre.
- Időre történő, maradéktalan teljesítéshez a mérési segédlet hiánytalan ismeretét feltételezzük.
- A sikeres teljesítéshez a gyakorlat során végzett munka – a futási eredményeket is beleértve – jegyzőkönyv formájában történő dokumentációja és határidőre történő leadása szükséges.

JEGYZŐKÖNYV:

A jegyzőkönyvnek – a megoldók nevén, neptun-kódján, és e-mail címén kívül – a következőket kell tartalmaznia **minden egyes részfeladatra vonatkozóan**:

- KI-MIT oldott meg (feladat felosztása és értelmezése).
- HOGYAN oldották meg (megoldás menetének leírása).
- MIÉRT úgy oldották meg (választott megoldás indoklása).
- EREDMÉNYEK összefoglalása (értelmezés, értékelés, screenshot-ok, magyarázat).

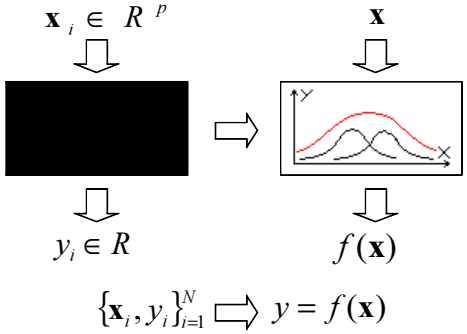
LEADÁS:

A jegyzőkönyvet (**PDF** formátumban), és a forráskódokat (azaz a `\jade\src\milab\lab06` könyvtár teljes tartalmát **ZIP**-be csomagolva) legkésőbb a **laborgyakorlati héten péntek délig** e-mail-ben, fájlként csatolva kell a **dkovacs@mit.bme.hu** címre elküldeni. A levél subject mezejében „**`[IR1 SVM]`**” szerepeljen (idézőjelek nélkül), míg a levél szövege rendre a *megoldók neve* és *neptunkódja* legyen. Több megoldó esetén csak egyetlen alkalommal, az egyik megoldónak kell elküldenie a megoldást, amit a többieknek is CC-ézzen.

2 Bevezetés

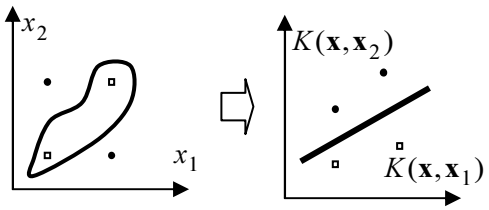
A laborgyakorlaton most tanuló ágensek építése lesz a feladat, amelyeknek egy függvény approximációs, és egy osztályozási problémát kell megoldaniuk egy speciális fajta Neurális Hálózattal, a Szupport Vektor Géppel (**S**upport **V**ector **M**achine – **SVM**). Az alábbiakban összefoglaljuk az SVM-el és annak gyakorlati alkalmazásával kapcsolatos legfontosabb tudnivalókat. A témáról részletesebb leírás található az alábbi könyvben [1]:
Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, **Valyon József**, „Neurális hálózatok”, Panem, 2006. (6. fejezet)

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), illetve ha valamennyire ismert a rendszer, „szürke doboz” modellezésnek nevezzük.



1. ábra: A fekete doboz modellezés.

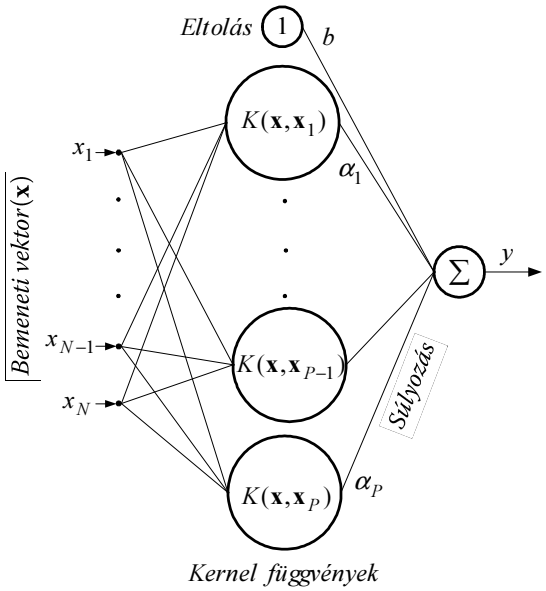
A modellezés eszközei között megtaláljuk a gépi tanulást, különféle „soft computing” technikákat és ezen belül a neurális hálózatokat is [1], [2]. A neurális hálózatok többek között általános eszközt adnak a (nemlineáris) regressziós feladat megoldására. A mérés során ezek egy speciális fajtájával a Szupport Vektor Gépekkel (SVM) foglalkozunk [3][4], melyek a gépek rejtett rétegében elhelyezkedő neuronjaiban található bázisfüggvényekből állítják elő a kívánt függvényt. A Szupport Vektor Gépek 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. Ehhez a leképezéshez a speciálisan származtatott $K(\mathbf{x}_i, \mathbf{x}_j)$ magfüggvényt alkalmazza (lásd később).



2. ábra: A nemlineáris leképezés, a probléma linearizálása.

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 gyakorlatban a szupport vektor gépeket ritkán formalizálják, mint neurális hálózatot, ugyanakkor ez az értelmezés nagyon hasznos, mert egyszerűbb tárgyalásmódot tesz lehetővé a hagyományos matematikai megközelítésénél. A szupport vektor gép pontosan megfeleltethető egy egy rejtett rétegű neurális hálózatnak. A rejtett réteg tipikusan nemlineáris neuronokat tartalmaz. Az **Hiba! A hivatkozási forrás nem található.**ábra egy szupport vektor gépnek megfeleltethető neurális hálózatot mutat be.



3. 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 (P). 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.

A hagyományos SVM alkalmazásának legnagyobb akadálya a módszer nagy algoritmikus komplexitása és a nagy memóriaigény, ami tipikusan a nagy adatmennyiség kezelését teszi lehetetlenné. Ezen 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]. Egy ilyen megoldást valósítja meg a mérésben is használt libsvm implementáció.

A fekete doboz modellezési problémák körében a két legáltalánosabb feladat a következő:

1. Függvény approximáció (avagy regresszió)
2. Osztályozás

Függvény approximáció esetén a rendelkezésre álló adatok közötti függvénykapcsolat elemzésére van szükség, 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.

Az **osztályozás** esetén a legegyszerűbb feladat a két osztályos szeparálás. Ebben az esetben a mintahalmazban szereplő vektorokhoz egy-egy osztály címke van rendelve. A modellezés feladata, hogy a bemenet és az osztálycímke közötti kapcsolatot megtalálja. Tulajdonképpen ez a feladat felfogható egy olyan függvény approximációs feladatnak mely az egyik osztályba tartozó vektorokhoz az egyik, a másikhoz a másik osztálycímkét (pl. +1,-1) rendeli.

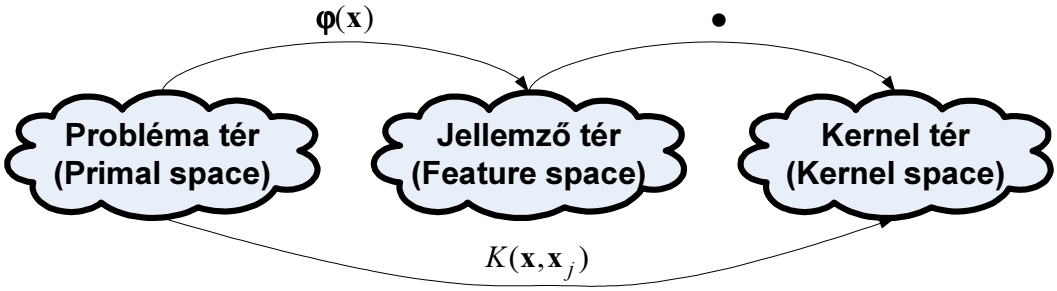
Az alábbiakban bemutatjuk az SVM származtatását mind az osztályozós, mind pedig a regressziós esetre, majd pedig részletezzük a laborgyakorlat során megoldandó – ehhez kapcsolódó – feladatokat.

3 Szupport Vektor Gépek

Az alábbiakban az SVM alapelveit, alapelemeit mutatjuk be gyakorlati megközelítésben. A részletes matematikai levezetéseket a mellékelt PDF tartalmazza.

3.1 Kernel függvények

Ahogy korábban már említettük, nemlineáris transzformációk alkalmazásával az eredetileg komplex nemlineáris problémák egyszerűbb lineáris feladatra vezethetők vissza. Ahhoz hogy lineáris problémát kapjunk a leképezésnek egy magasabb –esetenként akár végtelen– dimenziós térbe kell transzformálni a feladatot. Ahogy azt majd később látni fogjuk, az SVM megfogalmazásából eredő kernel trükknek köszönhetően nincs szükség a $\varphi(\mathbf{x})$ leképezés megadására, sőt a jellemző tér is elkerülhető.



4. ábra: A nemlineáris leképezések az eredeti probléma tértől a kernel térig.

A Mercer feltételek figyelembevételével levezethető, hogy a kernel függvény az n_h dimenziós jellemző vektorok direkt szorzata:

$$K(\mathbf{x}_i,\mathbf{x}_j)=\sum_{k=0}^{n_h}\varphi_k^T(\mathbf{x}_i)\varphi_k(\mathbf{x}_j)=\boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j) \qquad (3.1)$$

Látható, hogy kernel függvény egy i,j tanító minta páron értelmezett, így a kernel értékekből rendszerint egy kernel mátrix kerül kialakításra, ami tartalmazza az N tanítópont összes kombinációját.

$$\boldsymbol{\Omega}=\begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & \cdots & K(\mathbf{x}_i,\mathbf{x}_1) & \cdots & K(\mathbf{x}_N,\mathbf{x}_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_1,\mathbf{x}_i) & \cdots & K(\mathbf{x}_i,\mathbf{x}_i) & \cdots & K(\mathbf{x}_N,\mathbf{x}_i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_1,\mathbf{x}_N) & \cdots & K(\mathbf{x}_i,\mathbf{x}_N) & \cdots & K(\mathbf{x}_N,\mathbf{x}_N) \end{bmatrix} \qquad (3.2)$$

A kernel mátrix fontos jellemzője, hogy egy $N\times N$ -es szimmetrikus mátrix.

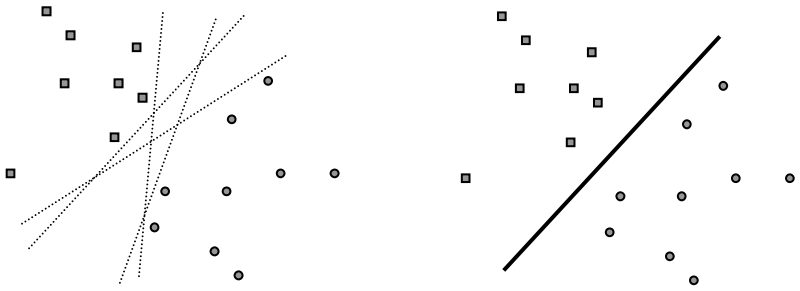
A mérés során az egyik leggyakrabban használt kernel függvényt az RBF magfüggvény használjuk:

$K(\mathbf{x},\mathbf{x}_k)=\exp\left\{-\|x-x_k\|^2/\sigma^2\right\}$, ahol σ konstans. (3.3)

Az SVM tehát lényegében az $\boldsymbol{\Omega}$ mátrix által reprezentált kernel térben keres egy lineáris megoldást.

3.2 Osztályozás Szupport Vektor Géppel

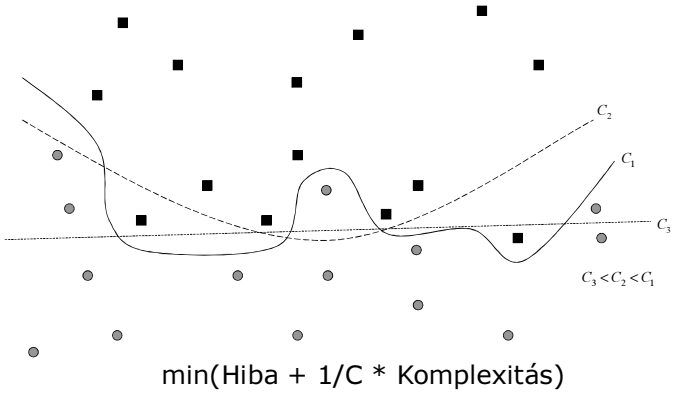
A szupport vektor gépekkel történő osztályozás (pl. kétosztályos szeparálás) alapgondolata, hogy a lehetséges megoldások közül azt az elválasztó felületet válassza ki, amely valamilyen értelemben optimális. Ez az optimum jelen esetben az a hipersík, amely a két osztály között középen –azaz mindkét osztálytól maximális távolságra – helyezkedik el.



5. ábra: A szeparálási feladat néhány lehetséges, valamint az optimális megoldása.

A VC dimenzió minimalizálásával bevezetett extra feltételnek köszönhetően a szupport vektor gép úgy garantál jobb megoldást, mint a pusztán tanítópontok alapján dolgozó eljárások, hogy közben nem használ fel a problémára jellemző apriori információt.

A szupport vektor gépek ezt a megoldást a statisztikus tanuláselmélet eredményeire alapozva, a strukturális hiba minimalizálás felhasználásával érik el. A megoldás lényege az a tény, hogy a modell általánosítási hibájára (azaz a tanítás során nem használt mintapontokra számított hiba) felső korlátot ad a tanítópontokra számított hiba és egy VC dimenziótól (az SVM modell kapacitásától) függő tag összege.



6. ábra: Az osztályozási hiba és a szeparáló felület komplexitása közötti kapcsolat.

Lineárisan szeparálható probléma esetén az első tag nulla (hiszen itt van megfelelő szeparálás), így jó látható, hogy a második tag minimalizálása adja a legjobb általánosítást. Ez megfelel a legnagyobb távolságot (margót) jelentő megoldásnak.

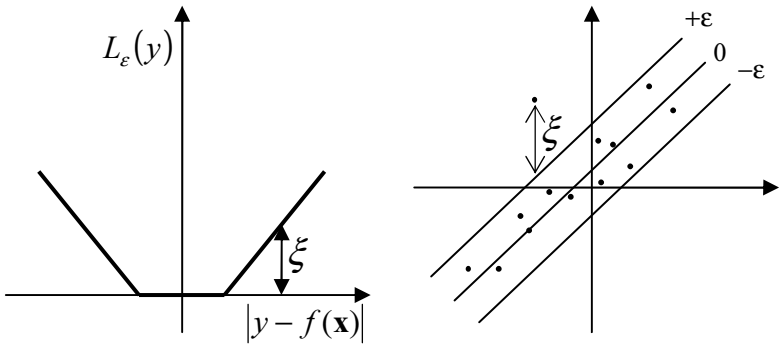
A szupport vektor gép a megoldás során kiválasztja a tanító vektorok egy részhalmazát, amelyek a modell alapját adják. Ezek a vektorok az úgynevezett szupport vektorok (tartó vektorok).

3.3 Regresszió Szupport Vektor Géppel

Az osztályozásra kidolgozott SVM általánosítható a regresszió esetére is. A függvény approximációs feladatok esetében azonban nem a távolság maximalizálás fogalma. Ez szükségessé tesz egy olyan értelmezést, felírására, ami lehetővé teszi az SVM kiegészítő feltételének (komplexitás) felhasználását ebben az problémakörben is.

Függvényapproximációs feladatnál az átlagos négyzetes hiba helyett az ε érzéketlenségi sávval rendelkező abszolútérték függvényt (ε-insensitive loss function) alkalmazzák az eltérés mérésére.

$$L_\varepsilon(y) = \begin{cases} 0 & for\ |f(\mathbf{x}) - y| < \varepsilon \\ |f(\mathbf{x}) - y| - \varepsilon & egyébként \end{cases} \qquad (3.4)$$

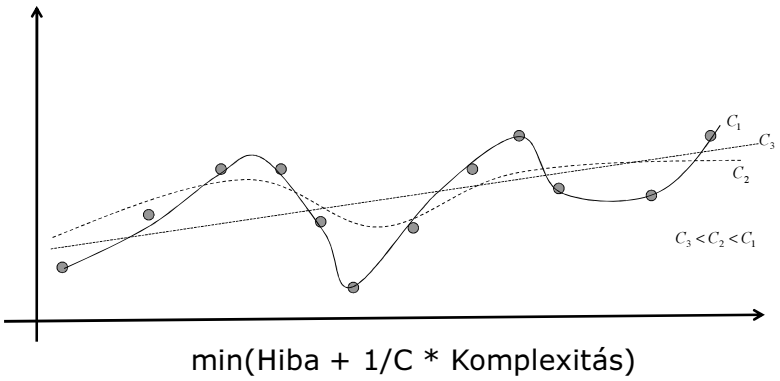


a.) b.)

7. ábra: Az ε érzéketlenségi sávval rendelkező abszolút érték függvény alkalmazása.

Az ε érzéketlenségű veszteség függvény felhasználásával tulajdonképpen az osztályozási esethez nagyon hasonló probléma áll elő. Azokra a mintákra, melyek az érzéketlenségi sávon belül esnek, a hiba értéke 0, míg a kívül eső pontok hibáját büntetjük. Ez megfeleltethető az osztályozási feladatnak, ahol a helyes osztályozás hibája 0, míg biztonsági sávba eső vagy a rosszul osztályozott pontokat pedig gyengítő (ξ) változók bevezetésével büntetjük.

Az ε érzéketlenségű veszteség függvény alkalmazásával az osztályozós problémához hasonló optimalizálási feladatra jutunk.



8. ábra: Az approximáció hibája és a szeparáló felület komplexitása közötti kapcsolat

3.4 Az SVM működése

A nemlineáris problémák lineáris kezelésének kulcsa, hogy az eredeti feladatot egy olyan véges dimenziójú térbe transzformáljuk ahol az már lineárisan megoldható. A szupport vektor gépek ezt kiegészítik azzal, hogy a lineáris megoldás kiszámításához olyan módszert adnak, ami alapján a lehetséges megoldások közül, a legjobb számítható ki.

A kernel módszerekben alkalmazott nemlineáris transzformáció valójában két egymást követő transzformáció eredménye. Az első transzformáció a tanító adatokat egy magasabb –esetleg végtelen – dimenziós térbe transzformálja, míg a második a kernel trükk révén ismét egy véges kernel térbeli reprezentációját adja a problémának. A transzformációk részleteit korábban már bemutattuk, így itt a működés mechanizmusának megértéséhez a kernel tér alábbi jellemzőit hangsúlyozzuk:

- A kernel tér a jellemző tér, illetve az ehhez tartozó leképezés konkrét ismerete nélkül is elérhető.
- A kernel tér véges dimenziójú.
- A kernel függvény és a kernel középpontok (szupport vektorok) ismeretében minden bemeneti vektor leképezhető a kernel térbe.

A probléma a kernel térben már lineáris így itt az SVM-nek megfelelő optimális hipersíkot kell meghatározni. Ez osztályozás esetén megfelel a leképezet két ponthalmazt maximális margóval elválasztó felületnek, regresszió esetén pedig a leképezett pontokra illesztett hipersíknak. A tanítási lépésnek tehát a hipersíkot leíró egyenlet szabad paramétereit az α_i-ket és a b-t kell meghatározni.

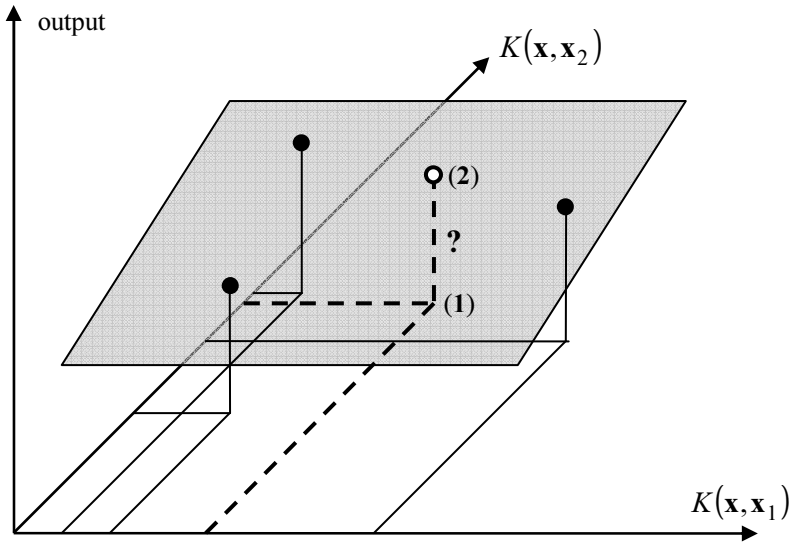
Amennyiben a szupport vektorok száma M, a kernel tér $M+1$ dimenziós, ahol M koordinátát a $K(\mathbf{x}_i,\mathbf{x}_j)$ kernel leképezéssel számítunk ki, míg a további egy dimenzió a kimenetnek felel meg. A keresett hipersík a leképezett tanító minták és a hozzájuk tartozó kimenet közötti összerendelést definiálja. A működés szemléltetéséhez a regressziós esetet mutatjuk be. A megtanított SVM működését az 9. ábra szemlélteti. Az SVM modell eredményét leíró

$$y(\mathbf{x}) = sign\left[\sum_{i=1}^{N_S} \alpha_i d_i K(\mathbf{x}_i,\mathbf{x}) + b\right] \qquad (3.5)$$

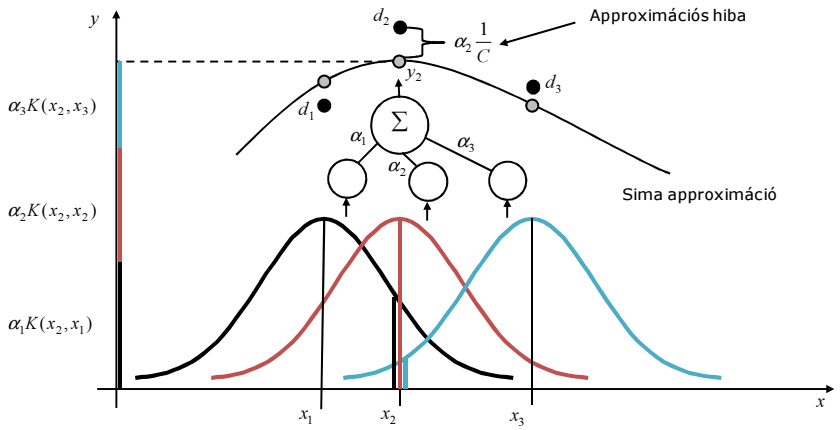
$$y(\mathbf{x}) = \sum_{i=1}^{N_S} (\alpha_i - \alpha_i')K(\mathbf{x},\mathbf{x}_i) + b \qquad (3.6)$$

egyenletek alapján a működés a következő:

- A vizsgált $\mathbf{x}$ bemeneti vektort leképezzük a kernel térbe. Ez a szupport vektoroknak megfelelő számú $K(\mathbf{x}_i,\mathbf{x}_j)$ kernel kiszámítását jelenti.
- Az eredmény a hipersík által definiált pont a leképezéssel kapott pozícióban. Ennek kiszámítása a hipersík egyenletéből, azaz a modell eredményeként megadott összefüggésből könnyen számítható.



9. ábra: A kernel térbeli lineáris megoldás illusztrációja két bemeneti és egy kimeneti dimenzió esetén. A fekete pontok a tanító mintákat illusztrálják, míg a fehér pont egy kiszámítandó kimenetet illusztrál.



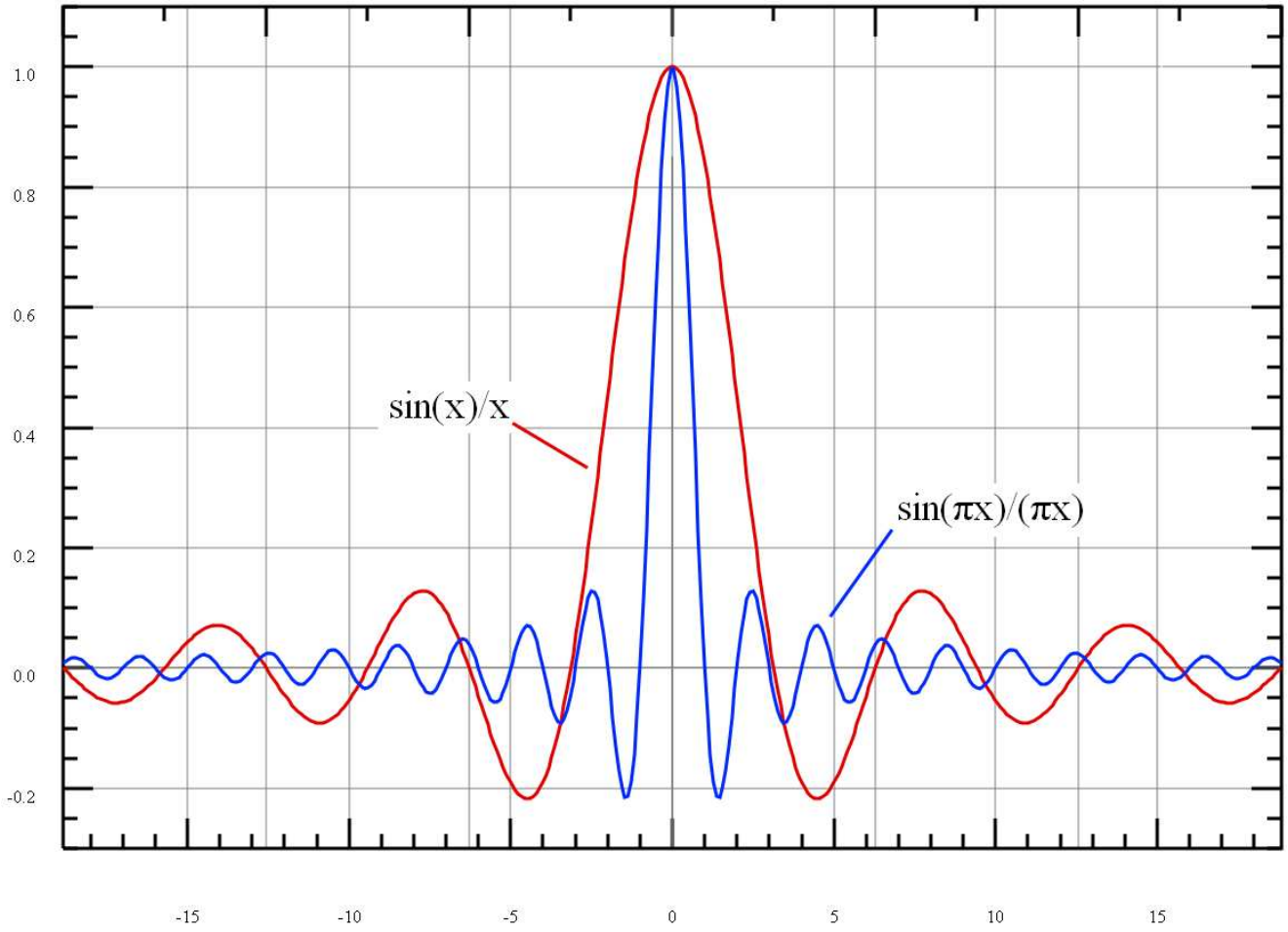
10. ábra: Az SVM regresszió működése az eredeti (probléma) térben.

A regressziós feladat megoldása során az SVM megkonstruált optimális hipersík – amennyire lehetséges – jó választ ad a tanító pontokra azaz illeszkedik rájuk. A modell akkor működik megfelelően, ha nem csak a tanítópontokra, hanem később a teszteléshez használt pontok esetén is a hipersíkra esik a kimenet, vagy másképpen a hipersík értéke jól közelíti a valós kimenet értéket.

4 Feladatok

Regresszió

A regressziós feladatban 2 ágens típus szerepel. A **tanár ágensek** (`milab.lab05.teacherAgent.TeacherAgent`), akik a $\mathrm{sinc}(x) = \sin(x)/x$ függvényt (lásd. 11. ábra) paraméterezhetően megadható zajjal tudják mintavételezni, valamint **tanuló ágensek** (`milab.lab05.studentAgent.StudentAgent`), akik a tanár ágens(ek)től kérdezik le a függvény értékeit és ez alapján (ha elegendő információ összegyűlt), approximálják a függvényt.



11. ábra: The normalizált (kék) és nem normalizált (piros) sinc függvény a [−6π, 6π] tartományon.

A működés során a tanuló ágens tetszőleges helyeken lekérdezi a tanítókat (azaz mintavételezi a – tanítók által ismert – függvényt), majd meghatározza mely (paraméterben megadott számú) tanítómintákat használja fel a modell elkészítéséhez. A tanuló ágens a tanítóktól kapott adatok alapján egy SVM modellt épít, ahol a helyes eredményhez meg kell határozni a megfelelő hiperparamétereket. A tanuló ágens a kész modellt képes validálni, illetve a valós sinc(x)-hez képest ki tudja számítani az approximáció hibáját. A tanítás után a `\jade\src\milab\lab05\SampleFiles\sinc_test.txt` fájlban szereplő mintapontokra meghatározza a modell válaszát, amit egy `output.txt` fájlba kiment. A `sinc_test.txt` tulajdonképpen egyenletes eloszlásban tartalmaz bemeneti mintapontokat a [-10,+10] tartományból, amelyekre a modellel kiszámítandó a kimenet (ez kerül az `output.txt` fájlba). Ezen kimeneti fájl segítségével az SVM modell eredménye Excel-ben megtekinthető, ábrázolható. Az ábrázolás menete a következő:

1. Az `output.txt` fájl megnyitása Excel-ben (*File → Open → All files…*),
2. a kimeneti oszlop kijelölése,
3. és `Line` grafikon ábrázolása (*Insert → Line*).
   (Szükség esetén a bemeneti pontok – a `sinc_test.txt`-ben szereplő adatsor – a kimenet mellé másolható, hogy a grafikon precízebb legyen.)

A `TeacherAgent` indításkor egy paramétert (egy valós számot) vár, ami arányos a sinc(x)-re adott Gauss zaj mértékével (0 a zajmentes eset, míg a 0.3 érték már egy elég zajos mintát eredményez). Itt tulajdonképpen azt lehet megadni, hogy az egyes tanárok mennyire pontosan, illetve pontatlanul ismerik a megtanulandó függvényt.

A `StudentAgent` egyetlen indítási paramétere a tanuláshoz szükséges minták/mintapontok száma, azaz hogy hányszor, hány helyen fogja a diák lekérdezni a tanárok által ismert függvényt.

1. Próbálja ki a működést zajmentes esetben, miközben az ágensek közti üzenetváltást figyelje meg a `Sniffer` ágenssel. Végül ellenőrizze az `output.txt`-ből Excel segítségével a megtanult függvény alakját. *Megjegyzés: ebben az esetben a tanárok pontos választ adnak a lekérdezett mintapontokban, így megfelelő számú és eloszlású minta – pl. néhány 100 egyenletes eloszlás szerint vett minta – alapján, megfelelő hiperparaméterek (ez a kódban állítható) esetén a sinc(x) függvény nagyon jól, pontosan megtanulható.*

2. Növelje a zajt, valamint a mintapontok számát, illetve hangolja (kézzel) a hiperparamétereket és értékelje az eredményeket. Ellenőrizze az `output.txt`-ből Excel segítségével a megtanult függvény alakját.

3. Írja át a `TeacherAgent` ágens kódját úgy, hogy esetenként (kb. 100-ból egyszer) nagy hibát (outliert) adjon vissza. Értékelje ismét a működést. Ellenőrizze az `output.txt`-ből Excel segítségével a megtanult függvény alakját.

Amennyiben maradt még ideje, szorgalmi feladatként oldja meg a következőket (legfeljebb +1 jegy):

4. Indítson el több `TeacherAgent` ágenst, különböző zaj paraméterekkel, és módosítsa a `StudentAgent` ágens kódját úgy, hogy ki tudja választani, hogy mely `TeacherAgent` tanítópontjaiból tanul. A kiválasztást javasolt az alábbi módszerek egyikével megtenni.

   a. Kevésbé realisztikus megoldás: a `StudentAgent` ágens a sinc(x) függvényt 20 pontban ismerheti (pl. [-10,10] tartomány egész pontjain) és ennek fényében választja ki a legjobb tanárt. Itt a legegyszerűbb, ha az ismert pontokban lekérdezi az egyes tanítókat (többször is, hiszen a random zajnak csak a szórását állítottuk be), és azt választja, amelyik a legpontosabb válaszokat adja.

   b. Realisztikusabb megoldás: a `StudentAgent` ágens ugyanazon pontban többször is lekérdezi ugyanazon tanítót, és ennek alapján számítja ki a zaj szórását. Ezt minden tanítóval megismételve kiválasztható a legpontosabban válaszoló, „legmagabiztosabb” tanító. *Megj.: feltételezzük, hogy a tanítók helyesen tudják a megtanulandó függvényt.*

Osztályozás

A feladatban 2 fajta ágens kommunikál egymással: a **kamera ágens** (`milab.lab05.webcamAgent.WebcamAgent`), és az **osztályozó ágens** (`milab.lab05.classifierAgent.ClassifierAgent`). A `WebcamAgent` egy vektorizált képet küld az osztályozó `ClassifierAgent` ágensnek, amely ennek alapján tanul, és osztályozza a kapott mintákat.

Két részfeladat lesz:

a) **+/– osztályozás,**
b) **vidám/szomorú smiley osztályozás.**

1. táblázat: a mintahalmaz elemei.

a.)		b.)	
+	–	☺	☹

A tanításhoz több tanító készlet áll rendelkezésre, ami fájlból olvasható be (előre elkészített minták a `\jade\src\milab\lab05\SampleFiles` könyvtárban), illetve ezek után valós kamerakép alapján is megvizsgálhatja a működést.

A feladat során – csakúgy, mint a regressziós esetben – be kell állítani az SVM osztályozó hiperparamétereit. Ennek egyik módja lenne, hogy **(1)** ugyanúgy, mint eddig, ***kézzel*** próbálja meg behangolni a megfelelő paramétereket. Ez azonban lassú, nehézkes, valamint nemigen lehet – nagyon nehézkes – megfelelően pontosan meghatározni az optimális beállítást. Egy másik módszer **(2)** a ***keresztkiértékelés (cross validation)***. Ebben az utóbbi esetben készíteni kell egy ciklust, ami folyamatosan változtatja a hiperparaméterek értékét egy-egy tartományon, majd rendre elvégzi a modellezést (azaz felépíti az SVM-et), ellenőrzi az adott modell eredményét, és megjegyzi hibát. A ciklus lefutatása után a legkisebb hibát eredményező hiperparaméter-kombinációt kell kiválasztani. Az eredmény tovább pontosítható, ha az optimalizált hiperparaméterek szűkebb környezetében ismét elvégzünk egy keresést.

Amikor nem áll rendelkezésre egy ellenőrző mintakészlet, a gyakorlatban a tanítómintákat osztják fel két részre, hogy a tanításhoz és az ellenőrzéshez (hiba kiszámításához) is legyen egy-egy állomány. A kiadott implementáció

- a tanításhoz 2 fájlt használ (pl. `plussz1.txt`, és `minusz1.txt`),
- a validáláshoz pedig 1-et (pl. `minusz1_plussz1.txt`), ami már mindkét osztályból tartalmaz mintákat.

Az osztályokat reprezentáló mintafájlokat célszerű úgy szétszedni, hogy a minták egy része tanító állomány (pl. 80 db), egy másik része pedig teszt állomány (20 db) legyen. Ez azért fontos, mert alapértelmezésben kiadott két, egyenként 100 mintát tartalmazó tanító, illetve az egyesített, 200 mintát tartalmazó ellenőrző állomány tartalma azonos, azaz ezzel futtatva tulajdonképpen csak a tanítópontokban ellenőriznénk a működést. *Megjegyzés: ennek eredményeképp a modellnek nem lenne általánosító képessége, azaz nem működne jól olyan mintákra, amiket még nem látott (pl. a kameraképre).*

A mintaként kiadott állományokat tehát át kell átszerkeszteni úgy, hogy a tesztelésnél eddig nem látott mintákra tesztelje a modellt. Ezt legegyszerűbben úgy tehetjük meg, hogy az állományokat Excel-ben

megnyitjuk. Itt az egyes mintákhoz az Excel táblázat egyes sorai tartoznak majd. A sorok megfelelő törlésével, és átrendezésével könnyű a kívánt tanító és teszt állományokat előállítani. Arra viszont oda kell figyelni (**FONTOS!!!**), hogy mindkét mintatípus (a **+1**-el, és a **–1**-el címkézett osztályok) esetére is el kell végezni a tanító/teszt szétválasztást.

Végül, az eredményül kapott kimeneti fájl (`output.txt`) alapján megkereshető a hibásan osztályozott bemeneti minta(ák) képe is. Ez a fájl gyakorlatilag ugyanolyan, mint regressziós esetben, csak itt most a kimeneti értékek nem függvényértékek, hanem osztálycímkék (+1, –1).

Az előbbi feladatok elvégzését követően a kamerás teszteléshez készítsen sajátkezüleg rajzolt képet, és ezt a kamera elé tartva tesztelje a működést.

1. Készítse el a +/– osztályozásra képes `ClassifierAgent` ágenst. A `WebcamAgent` ágens segítségével tesztelje néhány (5-10) saját mintára, és dokumentálja a működés helyességét (hibás osztályozások számát, stb).

Amennyiben maradt még ideje, szorgalmi feladatként oldja meg a következőt (legfeljebb +1 jegy):

2. Készítse el a **vidám/szomorú** smiley-t osztályozó `ClassifierAgent` ágenst. A `WebcamAgent` ágens segítségével tesztelje néhány (5-10) saját mintára és dokumentálja a működés helyességét (hibás osztályozások számát, stb).

5 Irodalomjegyzék

[1] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, Valyon József, „*Neurális hálózatok*”, Panem, 2006. (6. fejezet)

[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, pages 276–285, New York. IEEE, 1997.

[7] Thorsten Joachims. “*Making Large–Scale SVM Learning Practical*”, Advances in Kernel Methods–Support Vector Learning’, MIT Press, Cambridge, USA, 1998.