7. fejezet

Kernel módszerek

Ebben a fejezetben olyan tanuló rendszerekkel foglalkozunk, amelyek a válaszokat ún. kernel függvények (vagy magfüggvények) súlyozott összegeként állítják elő. A megközelítés bizonyos rokonságot mutat a bázisfüggvényes hálózatokkal, hiszen ott is függvények súlyozott összegeként kapjuk a megoldást, azonban a származtatás a kétféle megközelítésnél jelentősen eltér. A bázisfüggvényes hálóknál a bázisfüggvények által megvalósított nemlineáris leképezés a mintapontokat a bemeneti térből egy ún. jellemzőtérbe képezi le, és a megoldást a jellemzőtérbeli reprezentáció felhasználásával nyerjük. A jellemzőtérre való leképezés célja, hogy egy eredetileg csak nemlineáris eszközzel megoldható problémát lineáris géppel megoldhatóvá transzformáljunk: egy nemlineárisan szeparálható problémát lineárisan szeparálhatóvá tegyünk, vagy egy nemlineáris regressziós feladatot lineáris regressziós feladattá transzformáljunk.

A bázisfüggvényes hálózatok és ezen belül is elsősorban az RBF hálók konstrukciójánál az egyik leginkább megoldatlan kérdés a jellemzőtér dimenziójának, vagyis a bázisfüggvények számának a meghatározása. Annyit tudunk csak biztosan, hogy ha a jellemzőtér dimenziója „kellően nagy", akkor pl. a lineáris szeparálás lehetősége garantált. Azonban az általában nem definiált, hogy mit tekinthetünk „kellően nagy"-nak. Ha a jellemzőtér dimenzióját „túl nagy"-ra választjuk, akkor egyrészt feleslegesen növeljük a tanuló rendszer komplexitását, másrészt a szabad paraméterek száma túl nagy lesz, aminek túltanulás és a teljesítőképesség romlása is lehet a következménye. Olyan megoldásra volna szükségünk, ahol a jellemzőtér dimenziója „automatikusan kiadódik" nemcsak azt biztosítva, hogy lineáris megoldást kapunk, hanem azt is, hogy a megoldás hibája, a 3. fejezetben definiált valódi kockázat, a lehető legkisebb lesz – függetlenül attól, hogy hány dimenziós a jellemzőtér. Az ún. kernel gépek ezt a célt próbálják elérni.

A kernel gépeknél is alkalmazzuk a jellemzőtérbe való transzformációt, a feladatot azonban nem a jellemzőtérbeli reprezentáció felhasználásával oldjuk meg, hanem erről áttérünk egy ún. kernel reprezentációra és a kernel térben kapjuk a megoldást. A jellemzőtérbeli és a kernel térbeli reprezentáció közötti kapcsolat egy újabb transzformációként is értelmezhető. Míg a jellemzőtérben valójában

bázisfüggvényes megoldással dolgozunk, addig a kernel térbeli reprezentáció a jellemzőtérbeli reprezentációból belső szorzattal nyerhető.

A kernel térbeli reprezentáció azzal az előnnyel jár, hogy itt a szabad paraméterek száma független a jellemzőtérbeli reprezentáció szabad paramétereinek számától. Ezt az előnyt ki is tudjuk használni, ha a belső szorzatot egy megfelelően megválasztott kernel függvénnyel helyettesítjük, vagyis ahelyett, hogy adott bázisfüggvényeket definiálnánk és ezek skalár szorzatát felhasználva állítanánk elő a kernel reprezentációt, közvetlenül a kernel függvényt választjuk meg. A kernel függvény megválasztása implicit módon meghatározza a jellemzőtérbeli leképezés bázisfüggvényeit és így a jellemzőtérbeli reprezentációt is, anélkül, hogy a bázisfüggvényeket közvetlenül kellene definiálni és felhasználni.

A fejezet célja, hogy bemutassa a kernel módszerek alapgondolatát és a legfontosabb kernel gépek konstrukciójának a lépéseit. A bemutatásnál a legegyszerűbb, lineáris feladatból kiindulva jutunk el a nemlineáris problémák kernel géppel történő megoldásáig. A fejezet azt is bemutatja, hogy kernel reprezentációs megoldásra több, különböző megközelítés vezethet. Ezek közül talán a legfontosabb az elsősorban Vladimir Vapnik nevéhez köthető szupport vektor gép *(support vector machine, SVM)*. A szupport vektor gépek azon túl, hogy a kernel térben szolgáltatják a megoldást, további fontos tulajdonságokkal is rendelkeznek. Az SVM-nél – az eddig bemutatott hálóktól eltérően – a megoldás komplexitása és teljesítőképessége kézben tartható. A fejezet ugyanakkor azt is bemutatja, hogy kernel gépek származtathatók a klasszikus négyzetes hibaminimalizáló eljárással is, tehát bizonyos értelemben az eddig tárgyalt neuronhálóknak is létezhetnek kernel változatai. Ehhez kapcsolódóan a fejezet tárgyalja az LS-SVM hálózatokat és az ún. ridge regressziós *(ridge regression)* megoldást is.

A kernel gépek elméleti háttere szorosan kapcsolódik a belső szorzat terek *(inner product spaces, dot product spaces)*, a Hilbert terek *(Hilbert spaces, reproducing kernel Hilbert spaces)* témakörökhöz. A fejezet ezzel a kapcsolattal csak utalás szinten foglalkozik, a témakör iránt mélyebben érdeklődők a bőséges irodalomban, pl. [Sai88, Sch02, Vap98, Sző72] tájékozódhatnak.

A fejezet felépítése a következő: Először bemutatjuk az irodalomban gyakran kernel trükknek nevezett módszert, ami lehetőséget ad a kernel függvények közvetlen felírására, alkalmazására. Ezt követően bemutatjuk a legismertebb és legelterjedtebb kernel módszert alkalmazó eljárást, a szupport vektor gépeket *(Support Vector Machines – SVM)*, valamint ennek néhány változatát. A szupport vektor gépek jelentős számításigényű eljárások. A számítási igények csökkentésére ezért az SVM megközelítésétől néhány ponton eltérő kernel megoldások is születtek. Ezek között a legfontosabbak az LS-SVM és az ezzel matematikailag azonos, de más indítékból származó ridge regresszió. A fejezetben ezeknek a megoldásoknak a származtatását is összefoglaljuk, és arra is kitérünk, hogy mi az ára az egyszerűbb számításoknak. Az LS-SVM a szupport vektor gépek egyik fontos tulajdonságát, nevezetesen, hogy ritka (sparse) megoldást elveszíti. Elsősorban e hiányosság kiküszöbölését célozza az LS^2-SVM, valamint a ridge regresszió egy redukált változata, melyeket szintén összefoglalunk. A fejezet végén egy konkrét hálózat kapcsán megmutatjuk, hogy a bázisfüggvényes

hálózatok kernel gép formában is megfogalmazhatók, biztosítva így a kernel megközelítés előnyeit.

7.1. Egy egyszerű kernel gép

A kernel reprezentációt a legegyszerűbben talán egy lineáris regressziós feladat kapcsán mutathatjuk be. Adott egy lineáris gép, amely a következő be-kimeneti leképezést valósítja meg:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b\,, \tag{7.1}$$

Itt az eddigieknek megfelelően $\mathbf{w}$ a lineáris gép súlyvektora, b pedig az eltolásérték. Amennyiben rendelkezésünkre áll egy $\{d_i, \mathbf{x}_i\}_{i=1}^P$ tanítópont készlet olyan súlyvektort keresünk, ami mellett a

$$C(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{P}\left(d_i - \mathbf{w}^T\mathbf{x}_i - b\right) \tag{7.2}$$

kritériumfüggvény minimumot vesz fel. A minimális négyzetes hibát biztosító súlyvektor – ahogy ezt az előző fejezetekben már láttuk – a kritériumfüggvény deriváltja alapján határozható meg. A b eltolásértéket a súlyvektor nulladik komponenseként kezelve és bevezetve a kibővített súlyvektort $\hat{\mathbf{w}} = \left[b, \mathbf{w}^T\right]^T$, valamint a kibővített bemeneti vektort $\hat{\mathbf{x}} = \left[1, \mathbf{x}^T\right]^T$ is, a lineáris gép válasza $\mathbf{x}$ bemenet mellett

$$y(\mathbf{x}) = \hat{\mathbf{w}}^T\hat{\mathbf{x}} = \sum_{i=0}^{N}\hat{w}_i\hat{x}_i \tag{7.3}$$

formába írható, ahol N a bemeneti $\mathbf{x}$ vektorok dimenziója. A tanítópontokban a lineáris gép válaszai és a kívánt válaszok közötti eltérésekből képezhetünk egy hibavektort

$$\boldsymbol{\varepsilon}(\mathbf{w}) = (\mathbf{d} - \hat{\mathbf{X}}\mathbf{w})\,, \tag{7.4}$$

melynek segítségével az eredő négyzetes hiba a

$$C(\mathbf{w}) = \frac{1}{2}\boldsymbol{\varepsilon}^T(\mathbf{w})\boldsymbol{\varepsilon}(\mathbf{w}) = \frac{1}{2}(\mathbf{d} - \hat{\mathbf{X}}\mathbf{w})^T(\mathbf{d} - \hat{\mathbf{X}}\mathbf{w}) \tag{7.5}$$

összefüggéssel adható meg. Itt $\mathbf{d}$ a tanítópontokban a kívánt válaszok vektora, $\hat{\mathbf{X}}$ pedig a tanítópontok kibővített bemeneti vektoraiból képezett mátrix:

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{\mathbf{x}}_1^T \\ \hat{\mathbf{x}}_2^T \\ \vdots \\ \hat{\mathbf{x}}_P^T \end{bmatrix}. \tag{7.6}$$

Elvégezve a gradiens számítást és a gradienst nullává téve

$$\frac{\partial C(\mathbf{w})}{\partial \mathbf{w}} = -\hat{\mathbf{X}}\mathbf{d} + \hat{\mathbf{X}}^T\hat{\mathbf{X}}\mathbf{w} = \mathbf{0} \tag{7.7}$$

a súlyvektor legkisebb négyzetes hibájú (LS) becslését most is a pszeudo-inverz segítségével kapjuk meg:

$$\hat{\mathbf{w}}^* = \hat{\mathbf{X}}^T \left(\hat{\mathbf{X}} \hat{\mathbf{X}}^T \right)^{-1} \mathbf{d} \,. \tag{7.8}$$

Ha a kapott súlyvektort behelyettesítjük a (7.3) összefüggésbe, akkor adott x-re a lineáris gép válasza a következő lesz:

$$y(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \hat{\mathbf{w}}^* = \hat{\mathbf{x}}^T \hat{\mathbf{X}}^T \left(\hat{\mathbf{X}} \hat{\mathbf{X}}^T \right)^{-1} \mathbf{d} \,. \tag{7.9}$$

Vezessük be az

$$\boldsymbol{\alpha} = \left(\hat{\mathbf{X}} \hat{\mathbf{X}}^T \right)^{-1} \mathbf{d} \tag{7.10}$$

jelölést. Ekkor a kimenet az alábbi formában is felírható:

$$y(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \hat{\mathbf{X}}^T \boldsymbol{\alpha} = \sum_{i=1}^{P} \alpha_i \left(\hat{\mathbf{x}}^T \hat{\mathbf{x}}_i \right) = \sum_{i=1}^{P} \alpha_i K_i(\hat{\mathbf{x}}) \,. \tag{7.11}$$

ahol α_i az $\boldsymbol{\alpha}$ vektor i-edik komponense. A (7.11) összefüggés származtatásnál figyelembe vettük a tanítópontokhoz tartozó bemeneti vektorok (7.6) összefüggését. Így $\hat{\mathbf{x}}^T \hat{\mathbf{X}}^T$ egy olyan vektor, melynek elemei az $\hat{\mathbf{x}}$ bemenet és az $\hat{\mathbf{x}}_i$ tanítópont-bemenetek skalár szorzataiként állnak elő:

$$\hat{\mathbf{x}}^T \hat{\mathbf{X}}^T = \begin{bmatrix} \hat{\mathbf{x}}^T \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}^T \hat{\mathbf{x}}_2 & \dots & \hat{\mathbf{x}}^T \hat{\mathbf{x}}_i & \dots & \hat{\mathbf{x}}^T \hat{\mathbf{x}}_P \end{bmatrix} . \tag{7.12}$$

A (7.11) összefüggés érdekessége, hogy a lineáris gép egy $\mathbf{x}$ bemenetre adott válasza a skalár szorzattal definiált $K_i(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \hat{\mathbf{x}}_i$ függvények súlyozott összegeként határozható meg. Egy lineáris gépnél ezeket a függvényeket nevezzük kernel függvényeknek vagy magfüggvényeknek.

A (7.11) összefüggés a lineáris regressziós leképezés alternatív felírását jelenti. Míg az eredeti (7.3) összefüggés a bemenetek súlyozott összegeként adja meg a választ, ahol a súlyokat a $\hat{\mathbf{w}}$ súlyvektor képviseli, addig a kerneles reprezentációban a kimenet a $K_i(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \hat{\mathbf{x}}_i$ függvények súlyozott összegeként áll elő, ahol a súlyokat az α_i együtthatók jelentik.

A fenti (7.3) és (7.11) összefüggések ugyanannak a feladatnak két eltérő reprezentációját képviselik. Nyilvánvaló, hogy a két reprezentáció ekvivalens. A (7.3) összefüggés a megoldást a „bemeneti térben”, közvetlenül $\mathbf{x}$ függvényeként adja meg, míg a (7.11) összefüggés ugyanezt a „kernel térben” a $K_i(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \hat{\mathbf{x}}_i$ kernel értékek függvényeként teszi meg.

A most vizsgált lineáris leképezésnél a kernel reprezentáció különösebb előnnyel nem jár, mindössze egy alternatív megadási formát jelent. A különbség a kétféle reprezentáció között csupán annyi, hogy a két szummás kifejezésben a tagok száma eltérő. A bemeneti térben történő összegzés $N + 1$ tagból, míg a kernel térbeli P tagból áll.

Más a helyzet akkor, ha nemlineáris leképezést akarunk megvalósítani. A nemlineáris feladatok egy lehetséges megoldása a bázisfüggvényes hálók alkalmazása, vagyis, ha a kimenetet a (6.1) összefüggéssel megadott

$$y = \sum_{i=0}^{M} w_i \varphi_i(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\varphi}(\mathbf{x}) \tag{7.13}$$

alakban állítjuk elő.

A bázisfüggvényes leképezés a bemeneti térből az ún. jellemzőtérbe transzformál. A jellemzőtérbeli megoldás alternatívájaként viszont most is alkalmazható a kernel térre való áttérés. A lineáris esetre vonatkozó fenti gondolatmenetet követve a (7.13)-mal megadott leképezés kerneles változata úgy nyerhető, hogy a (7.11) összefüggésben minden $\hat{\mathbf{x}}$ helyére $\boldsymbol{\varphi}(\mathbf{x})$ kerül. Ezzel:

$$y(\mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\Phi}(\mathbf{x})^T \boldsymbol{\alpha} = \sum_{i=1}^{P} \alpha_i \left(\boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}_i) \right) = \sum_{i=1}^{P} \alpha_i K_i \left(\boldsymbol{\varphi}(\mathbf{x}) \right) , \tag{7.14}$$

ami azt jelenti, hogy a kimenetet most is a skalár szorzattal definiált kernel értékek súlyozott összegeként állítjuk elő. Itt a

$$\boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}_i) = K_i \left(\boldsymbol{\varphi}(\mathbf{x}) \right) = K(\mathbf{x}, \mathbf{x}_i) . \tag{7.15}$$

függvény a kernel függvény, amit tehát a bázisfüggvények skalár szorzatával nyerhetünk, míg a

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}(\mathbf{x}_1)^T \\ \boldsymbol{\varphi}(\mathbf{x}_2)^T \\ \vdots \\ \boldsymbol{\varphi}(\mathbf{x}_P)^T \end{bmatrix} \tag{7.16}$$

mátrix a tanítópontok jellemzőtérbeli reprezentációiból felépülő mátrix.

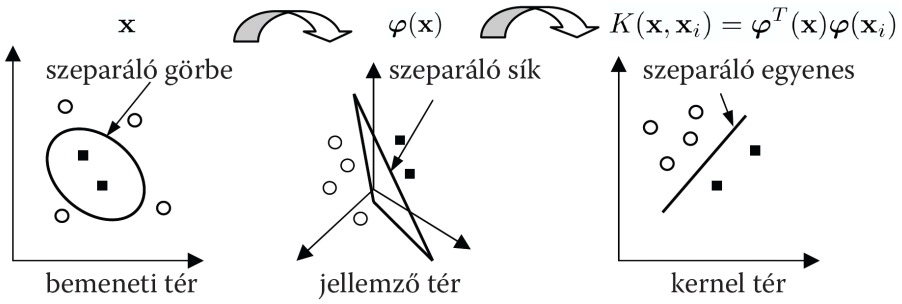
A (7.15) összefüggéssel megadott kernel reprezentáció itt is ekvivalens az eredeti reprezentációval ((7.13) összefüggés), és alkalmazása önmagában különösebb előnnyel itt sem jár. A kernel reprezentáció előnyei akkor látszanak, ha a kernel függvényeket nem a bázisfüggvények skalár szorzataként határozzuk meg, hanem közvetlenül felvesszük, ha tehát nem a bázisfüggvényekből, hanem a kernel függvényből indulunk ki. Kernel függvényként azonban bármilyen függvény nem használható, hiszen a kernel függvény még akkor is bázisfüggvények skalár szorzatával származtatható függvény kell legyen, ha a származtatásnál nem ezt az utat választjuk. Érvényes kernel függvénynek bizonyos feltételeket teljesítenie kell. A kernel függvény megválasztása implicit módon meghatározza a jellemzőtérre való leképezést biztosító bázisfüggvényeket is. Ez viszont már magában rejti a kernel megközelítés egy nagyon lényeges előnyét.

Amint az a (7.14) összefüggésből látszik a kerneles reprezentáció a tanítópontoknak megfelelő számú (P) kernel függvény-érték súlyozott összegeként áll elő, függetlenül attól, hogy az implicit módon definiált jellemzőtér dimenziója (M) mekkora. A kernel függvény megválasztásától függően a jellemzőtér dimenziója nagyon nagy, akár végtelen is lehet, ami a (7.13) szerinti kimenet előállítást

nagyon megnehezítené, sőt akár lehetetlenné is tenné, miközben a kernel reprezentáció komplexitása a tanítópontok száma által mindenképpen korlátozott. Minthogy a kernel térbeli megoldás ekvivalens a jellemzőtérbeli megoldással, a kernel módszerekkel azt tudjuk elérni, hogy a megoldás komplexitását akkor is korlátozni tudjuk, ha egyébként a megfelelő jellemzőtérbeli megoldás extrém módon komplex lenne. A kernel függvények bevezetésének ezt a hatását kernel trükknek *(kernel trick)* nevezzük.

Nemlineáris esetben a kernel reprezentációt – legalábbis koncepcionálisan – két lépésben érjük el. Először a bemeneti térről megfelelő bázisfüggvények alkalmazásával nemlineáris dimenzionövelő transzformációt hajtunk végre. Ennek eredménye a jellemzőtérbeli reprezentáció. A feladatot azonban nem itt, hanem egy újabb transzformáció után a kernel térben oldjuk meg. A jellemzőtér és a kernel tér között a skalár szorzat teremti meg a kapcsolatot, és mindkét származtatott reprezentációra igaz, hogy az eredetileg nemlineáris probléma a származtatott reprezentációkban már lineárisan megoldható. A reprezentációk kapcsolatát mutatja a 7.1. ábra.

A koncepcionális származtatás mellett azonban fontos hangsúlyozni, hogy a kernel reprezentáció alkalmazásánál nem a 7.1. ábrán bemutatott utat követjük, a kernel térbe nem a jellemzőtéren keresztül jutunk el, hanem épp ellenkezőleg a kernel térből indulunk ki, és ez automatikusan definiálja a jellemzőteret. A fordított út előnye – ami a kernel trükk következménye –, hogy nem kell a jellemzőteret definiálnunk, ami egyébként sok esetben komoly nehézséget jelentene, dolgoznunk sem kell a praktikusan nehezen kezelhető jellemzőtérben, hanem közvetlenül a kernel teret definiáljuk és a megoldást is itt nyerjük. Mindehhez az egyik legfontosabb követelmény a megfelelő kernel függvény megválasztása.



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

7.2. Kernel függvények

Az előzőekből nyilvánvaló, hogy kernel függvényként csak olyan függvény használható, amely belső szorzat segítségével is származtatható. A származtatás módjából következik, hogy a kernel függvények bizonyos tulajdonságokkal kell

rendelkezzenek.

Egy kernel függvénynek mindig két argumentuma van és ezekre nézve a függvény szimmetrikus kell legyen:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i)^T \boldsymbol{\varphi}(\mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_j)^T \boldsymbol{\varphi}(\mathbf{x}_i) = K(\mathbf{x}_j, \mathbf{x}_i)\,. \tag{7.17}$$

A kernel függvények általában kielégítik a következő követelményeket is:

$$\begin{aligned} &K(\mathbf{x}_i, \mathbf{x}_j) \geq 0\,, \\ &K(\mathbf{x}_i, \mathbf{x}_j) = K\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right)\,, \\ &K(\mathbf{x}, \mathbf{x}) = \max K(\mathbf{x}_i, \mathbf{x}_j)\,, \\ &\lim_{t\to\infty} K(t) = 0\,, \quad \text{ha} \quad t = \|\mathbf{x}_i - \mathbf{x}_j\|\,. \end{aligned} \tag{7.18}$$

A fentiek közül az első a nemnegativitást, a második a radiálisan szimmetrikus tulajdonságot jelenti. A harmadik feltételnek eleget tevő függvény maximumértéket vesz fel, ha mindkét argumentuma azonos, míg az utolsó azt fogalmazza meg, hogy a függvény a két argumentum távolságának monoton csökkenő függvénye.

A kernel függvényekkel támasztott követelmények precízebben is megfogalmazhatók: $K(\mathbf{x}_i, \mathbf{x}_j)$ lehet bármely olyan szimmetrikus függvény, amely kielégíti a Mercer tétel feltételeit [Vap98]:

$$\iint K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0 \;\; \forall f \neq 0 \text{ függvényre, ahol } \int f^2(\mathbf{x}) d\mathbf{x} < \infty\,, \tag{7.19}$$

ugyanis a Mercer tételt kielégítő függvények előállíthatók valamilyen jellemzőtérbeli $\varphi_i(\mathbf{x})$ függvények skalár szorzataként:

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \varphi_j(\mathbf{x}) \varphi_j(\mathbf{z})\,. \tag{7.20}$$

Néhány gyakran alkalmazott kernel függvényt az alábbi táblázatban foglalunk össze: A táblázat azt is jelzi, hogy egyes kernel függvények alkalmazásával olyan kernel térbeli hálóstruktúrát kapunk, amely bizonyos klasszikus neuronháló struktúrákhoz hasonló. Így pl. a Gauss kernel függvények az RBF hálóval rokon struktúrát, a tangens hiperbolikusz kernel függvény pedig olyan MLP hálózattal rokon struktúrát eredményez, melynek a kimeneti rétege lineáris.

A fenti, gyakran alkalmazott függvények mellett számos további függvényt használnak kernel függvényként. Megválasztásuk az egyes kernel gépeknél alapvetően fontos és sokszor igen nehéz kérdés. A választás nehézsége akkor jelentkezik, ha adott feladathoz „optimális” kernelt szeretnénk választani. Általában a gyakorlati feladatok megoldásánál nem is törekszünk „optimális” kernel megválasztására, hanem az előbb felsoroltak, vagy egyéb, szintén bevált függvények (pl. B-spline) közül választunk. A kernel függvények megválasztásának illetve konstrukciójának részleteivel itt nem foglalkozunk, csak az igen széleskörű irodalomra utalunk ld. pl. [Sch02], stb.

Lineáris	$K(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}_i^T \mathbf{x}$
Polinomiális (d fokszámú)	$K(\mathbf{x}, \mathbf{x}_i) = \left(\mathbf{x}_i^T \mathbf{x} + 1\right)^d$
Gauss (RBF)	$K(\mathbf{x}, \mathbf{x}_i) = \exp\{-\|\mathbf{x} - \mathbf{x}_i\|^2/\sigma^2\}$, ahol σ konstans.
Tangens hiperbolikusz (MLP)	$K(\mathbf{x}, \mathbf{x}_i) = \tanh\left(k\mathbf{x}_i^T \mathbf{x} + \theta\right)$, ahol k és θ megfelelően választott konstansok, mert nem minden kombináció eredményez magfüggvényt.

7.1. táblázat. A legelterjedtebben használt magfüggvények (kernel függvények)

Annyit azonban megjegyzünk, hogy megfelelő kernelek konstrukciójánál fontos szerepet játszhat, hogy különböző műveletekkel érvényes kernelekből további kernelek képezhetők. Ha pl. mind K_1, mind K_2 érvényes kernel, továbbá, ha a tetszőleges pozitív konstans, akkor a következők szerint további érvényes kernelek konstruálhatók:

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= K_1(\mathbf{x}, \mathbf{z}) + K_2(\mathbf{x}, \mathbf{z}) \\ K(\mathbf{x}, \mathbf{z}) &= aK_1(\mathbf{x}, \mathbf{z}) \\ K(\mathbf{x}, \mathbf{z}) &= K_1(\mathbf{x}, \mathbf{z}) K_2(\mathbf{x}, \mathbf{z}) \end{aligned} \tag{7.21}$$

Fentiekből az is következik, hogy kernelek lineáris kombinációja szintén kernel, ha a lineáris kombináció együtthatói nemnegatív számok

A kernel függvények előre történő megválasztása mellett az utóbbi időben számos olyan eredmény született, mely adatfüggő vagy adaptív kernelekkel oldja meg a feladatokat. Erre vonatkozó eredményeket mutat be többek között pl. [Cri98, Ama99, Vin00, Xio05, Mer06].

A későbbiekben látni fogjuk, hogy a kernel gépek konstrukciójánál fontos szerepet tölt be a kernel függvény $(\mathbf{x}_i, \mathbf{x}_j)$ $(i, j = 1, \ldots, P)$ tanító minta-páron értelmezett értékeiből képezett $\mathbf{K} = K(\mathbf{x}_i, \mathbf{x}_j)$ kernel mátrix, ami tartalmazza a P tanítópont összes lehetséges mintapárjánál a kernel függvény értékét.

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \cdots & K(\mathbf{x}_i, \mathbf{x}_1) & \cdots & K(\mathbf{x}_P, \mathbf{x}_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_1, \mathbf{x}_i) & \cdots & K(\mathbf{x}_i, \mathbf{x}_i) & \cdots & K(\mathbf{x}_P, \mathbf{x}_i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_1, \mathbf{x}_P) & \cdots & K(\mathbf{x}_i, \mathbf{x}_P) & \cdots & K(\mathbf{x}_P, \mathbf{x}_P) \end{bmatrix} \tag{7.22}$$

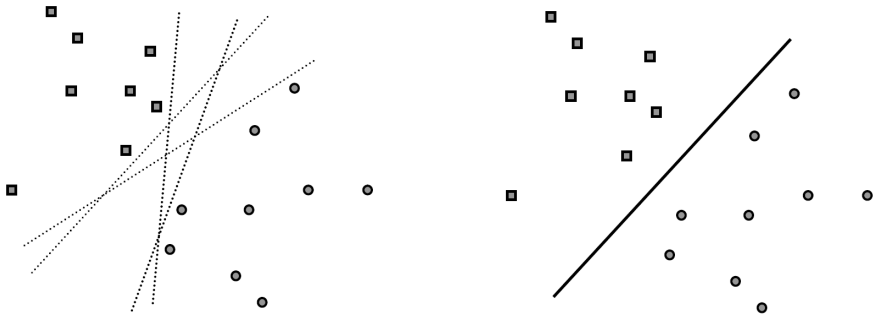
A kernel mátrix fontos jellemzője, hogy egy $(P \times P)$-es szimmetrikus mátrix. Az ilyen mátrixot szokás Gram-mátrixnak is nevezni.

7.3. Szupport Vektor Gépek

Az előzőekben bemutatott egyszerű kernel gép arra adott példát, hogy a 6. fejezetben bemutatott bázisfüggvényes hálók kernel gépként is értelmezhetők. Ennél az értelmezésnél a probléma megfogalmazása nem változott az eredeti bázisfüggvényes megfogalmazáshoz képest: mindkét esetben az átlagos négyzetes hiba minimumát biztosító megoldásra törekedtünk.

Kernel gépek azonban más megközelítésből is konstruálhatók. A Boser, Guyon és Vapnik által javasolt szupport vektor gép [Bos92] talán a legfontosabb, több új szempontot is hozó kernel gépnek tekinthető. A szupport vektor gépek olyan kernel gépek, melyek a statisztikus tanuláselmélet eredményeit is hasznosítják. Alapváltozatuk lineáris szeparálásra képes, amely azonban kiterjeszthető nemlineáris szeparálásra és nemlineáris regressziós feladatokra is.

Amint azt korábban láttuk, a lineáris szeparálási feladat megoldható egy egyszerű perceptronnal. A Rosenblatt perceptron azonban egy lineárisan szeparálható feladat egy lehetséges megoldását adja, miközben tudjuk, hogy lineárisan szeparálható mintapontok esetében általában végtelen sok különböző megoldásunk lehet. Ezek a megoldások egyenértékűek abból a szempontból, hogy mindegyikük egy olyan lineáris elválasztó felületet határoz meg, amely a tanítópontokat hibátlanul szétválasztja. A szeparálás minősége azonban az egyes megoldásokban mégis jelentősen különbözhet. Ezt illusztrálja a 7.2. ábra.



7.2. ábra. Egy lineárisan szeparálható feladat különböző megoldásai

Bár az ábra bal oldali részén látható összes egyenes hibátlanul szétválasztja a tanítópontokat, az a sejtésünk, hogy a jobb oldalon látható folytonos vonallal rajzolt egyenes kitüntetett a lehetséges megoldások között. A mintapontokból történő tanulásnál ugyanis nemcsak a tanítópontok megtanulására törekszünk, hanem megfelelő általánosítóképesség elérésére is. A mintapontok valamilyen – számunkra általában ismeretlen – eloszlású mintahalmazokból származnak, és a feladat nemcsak a véges számú ismert válaszú tanítópont helyes osztályozása, hanem az egész osztályozási feladat minél jobb megoldása is. Feltételezve, hogy a tanítópontok kellően reprezentálják a megoldandó problémát, azt várjuk, hogy az az elválasztás eredményez jobb általánosítóképességet, amely a két osztályba

tartozó tanítópontok között, a tanítópontoktól a lehető legnagyobb távolságra helyezkedik el. A lineáris kétosztályos osztályozási feladat megoldását adó szupport vektor gép ezt az „optimális" elválasztó felületet határozza meg.

A következőkben előbb ezt a legegyszerűbb esetet és a megfelelő megoldást mutatjuk be, majd a szupport vektor gépeket kiterjesztjük lineárisan nem megoldható osztályozási, végül nemlineáris regressziós feladatok megoldására is.

7.3.1. Lineárisan szeparálható feladat lineáris megoldása

Fentiek értelmében optimális lineáris szeparálásnak azt a megoldást tekintjük, amikor az elválasztó egyenes (sík, hipersík) a két osztályba tartozó tanítópontok között a pontoktól a lehető legnagyobb távolságra helyezkedik el. A pontok között középre elhelyezett szeparáló felületet a tanítópontoktól egy margó, azaz egy biztonsági sáv választja el, ezért az így megoldható feladatokat maximális tartalékot vagy maximális margót biztosító lineárisan szeparálható osztályozási feladatoknak is nevezzük. A következőkben ennek a feladatnak a megoldását mutatjuk be.

Legyenek adottak az $\{\mathbf{x}_i, d_i\}_{i=1}^{P}$ tanítópontok, ahol az $\mathbf{x}_i \in \mathsf{R}^N$ N-dimenziós bemeneti pontok két osztály valamelyikéből származhatnak, vagyis a kívánt válaszok a két lehetséges érték egyikét veszik fel: $d_i \in \{-1,1\}$. Egy olyan $g(\mathbf{x}) = = \mathbf{w}^T\mathbf{x} + b$ szeparáló függvényt keresünk, amely a tanítópontokat hiba nélkül osztályozza:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq a > 0 \quad &\text{ha} \quad d_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -a < 0 \quad &\text{ha} \quad d_i = -1 \end{aligned} \qquad (i = 1,2,\ldots,P) \tag{7.23}$$

továbbá, ahol a hipersíkhoz legközelebb álló tanítópontoknak a síktól vett távolsága maximális. A (7.23) egyenlőtlenségben a valamilyen kis pozitív konstans, d_i pedig az $\mathbf{x}_i$-hez tartozó kívánt válasz.

A hipersík paramétereinek megfelelő skálázásával biztosítható, hogy

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq +1 \quad &\text{ha} \quad d_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad &\text{ha} \quad d_i = -1 \end{aligned} \tag{7.24}$$

legyen, vagyis a síkhoz legközelebb eső pontokban az elválasztó függvény értéke 1 legyen. Így a feladat az alábbi tömörebb formában is felírható:

$$d_i \left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad (i = 1,2,\ldots,P)\,. \tag{7.25}$$

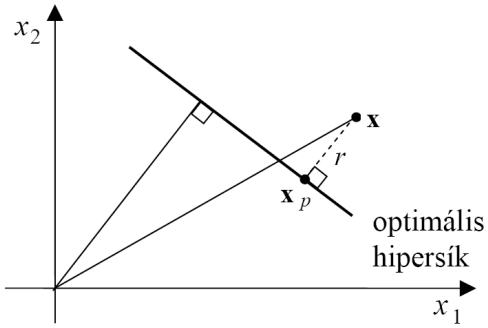
Jelölje az optimális szeparáló lineáris felület paramétereit $\mathbf{w}^*$ és b^* , és határozzuk meg egy $\mathbf{x}$ pontnak ettől a hipersíktól való távolságát. A 7.3. ábrát követve jelölje $\mathbf{x}_p$ az $\mathbf{x}$ pont merőleges vetületét a szeparáló felületre. Ekkor a pont felírható a vetület és a szeparáló felület normálvektora segítségével:

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}\,. \tag{7.26}$$

Mivel $g(\mathbf{x}_p) = 0$, $g(\mathbf{x}) = r\|\mathbf{w}^*\|$, vagyis az $\mathbf{x}$ pontnak a $g(\mathbf{x})$ szeparáló függvény által megadott hipersíktól való távolságára

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}^*\|} \tag{7.27}$$

adódik, ahol $\|\mathbf{w}^*\|$ a $\mathbf{w}^*$ vektor euklideszi normáját jelöli. Az optimális hipersík a síkhoz legközelebbi tanítópontok távolságának maximumát biztosítja.



7.3. ábra. A margó geometriai értelmezése

Az egységnyi távolságúra skálázott feladatban a határoló felülethez legközelebb eső pontokra $g(\mathbf{x}_s^+) = 1$ vagy $g(\mathbf{x}_s^-) = -1$. Ekkor a szeparáló hipersíkhoz legközelebb eső, különböző osztályba tartozó bemeneti vektorok közötti, a síkra merőlegesen mért távolság

$$\rho = \frac{g(\mathbf{x}_s^+)}{\|\mathbf{w}^*\|} - \frac{g(\mathbf{x}_s^-)}{\|\mathbf{w}^*\|} = \frac{1}{\|\mathbf{w}^*\|} - \frac{-1}{\|\mathbf{w}^*\|} = \frac{2}{\|\mathbf{w}^*\|}\,. \tag{7.28}$$

Az optimális hipersík által biztosított margó tehát:

$$r = \frac{\rho}{2} = \frac{1}{\|\mathbf{w}^*\|}\,. \tag{7.29}$$

Az egyenlet szerint az osztályozási tartalék akkor lesz maximális, ha $\|\mathbf{w}^*\|$ minimális értékű.

A megoldandó feladat ezek után a következőképpen fogalmazható meg: adott egy lineárisan szeparálható mintapont készlet

$$S = ((\mathbf{x}_1, d_1), \ldots, (\mathbf{x}_P, d_P))\ , \tag{7.30}$$

és keressük azt a minimális normájú $\mathbf{w}^*$ vektort és azt a b^* skalár értéket, melyekkel a tanítópontok mindegyikét helyesen osztályozzuk, vagyis keressük a

$$\mathbf{w}^* = \arg\min_{\mathbf{w}}(\mathbf{w}^T\mathbf{w}) \tag{7.31}$$

megoldását, azzal a feltétellel, hogy

$$d_i \left(\mathbf{w}^{*T} \mathbf{x}_i + b^* \right) \geq 1 \quad (i = 1,2,\ldots,P) \,. \tag{7.32}$$

A feladatot tehát feltételes szélsőérték-keresési problémaként tudjuk megfogalmazni, ahol a feltételek egyenlőtlenségek formájába vannak megadva. A feltételes szélsőérték-keresési feladat megoldását egy Lagrange egyenlet (ld. Függelék) megoldásával kereshetjük:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[d_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right] \,, \tag{7.33}$$

ahol az $\alpha_i \geq 0$ együtthatók a Lagrange multiplikátorok. Az optimalizálási feladatnak ezt a felírását elsődleges alaknak *(primal problem)* nevezzük.

A optimalizálási feladat megoldásához a Karush-Kuhn-Tucker (KKT) elmélet (ld. Függelék) szerint a fenti Lagrange egyenletet kell minimalizálni $\mathbf{w}$ és b szerint és maximalizálni α_i szerint, vagyis a Lagrange egyenlet által definiált kritériumfelület nyeregpontját *(saddle point)* kell meghatározni. A megoldás két lépesben érhető el:

- először $\mathbf{w}$ és b szerinti szélsőértéket keresünk a megfelelő deriváltak számításával,
- majd ezek eredményét behelyettesítve a Lagrange egyenletbe kapjuk az ún. másodlagos feladatot *(dual problem)*, melynek megoldásai az α_i Lagrange multiplikátorok.

Az $L(\mathbf{w}, b, \boldsymbol{\alpha})$ Lagrange egyenlet $\mathbf{w}$ és b szerinti parciális deriválásából az optimumra az alábbi két feltételt kapjuk:

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \quad \longrightarrow \quad \mathbf{w} = \sum_{i=1}^{P} \alpha_i d_i \mathbf{x}_i \,, \tag{7.34}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0 \quad \longrightarrow \quad \sum_{i=1}^{P} \alpha_i d_i = 0 \quad (\alpha_i \geq 0) \,. \tag{7.35}$$

Látható, hogy ha a Lagrange multiplikárokat ismerjük, a feladatot megoldottuk, hiszen $\mathbf{w}$ csak az α_i Lagrange multiplikátoroktól és a tanítópont-értékektől függ.

A feltételes optimalizálási feladat megoldásához felírható annak *duális alakja*, melyben már csak az α_i Lagrange multiplikátorok az ismeretlenek. Az elsődleges és a duális probléma megoldásai ugyanazon optimumra vezetnek.
Mivel az

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^{P} \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^{P} \alpha_i d_i - \sum_{i=1}^{P} \alpha_i \tag{7.36}$$

kifejtésében a harmadik tag (7.35) alapján nulla, és mivel (7.34) alapján

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{P} \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^{P}\sum_{j=1}^{P} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \tag{7.37}$$

az $L(\mathbf{w}, b, \boldsymbol{\alpha}) = Q(\boldsymbol{\alpha})$ duális optimalizálási feladat az alábbi:

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \tag{7.38}$$

a

$$\sum_{i=1}^{P} \alpha_i d_i = 0 \tag{7.39}$$

és a

$$\alpha_i \geq 0 \quad (i = 1,2,\ldots,P) \tag{7.40}$$

feltételekkel. A duális probléma egy olyan, a keresett α_i értékekben másodfokú szélsőérték-keresési feladat, mely mellett a megoldásnak egy egyenlőség, (7.39) és egy egyenlőtlenség, (7.40) típusú mellékfeltételt is ki kell elégítenie. Az ilyen típusú optimalizálási feladatok kvadratikus programozással *(Quadratic Programing)* oldhatók meg. Az optimumhoz tartozó Lagrange multiplikátorok – az α_i^* értékek – ismeretében, felhasználva az optimális súlyvektorra vonatkozó (7.34) összefüggést, a lineáris kétosztályos osztályozó szupport vektor gép válasza felírható, mint

$$y(\mathbf{x}) = \text{sign}\left[\sum_{i=1}^{P} \alpha_i^* d_i \mathbf{x}_i^T \mathbf{x} + b^*\right] . \tag{7.41}$$

Vegyük észre, hogy a válasz összefüggésében a bemeneti mintavektorok nem közvetlenül, hanem egy skalár szorzat részeként szerepelnek, tehát a (7.11) összefüggéshez hasonlóan itt is kernel megoldást kaptunk. A megoldás érdekessége, hogy az α_i-k nagyrésze általában 0, így mind a súlyvektor kifejezésében, mind a szupport vektor gép válaszában a tanítópontoknak csak egy része vesz részt.

Azokat a tanítópontokat, amelyek résztvesznek a megoldás kialakításában, amelyekhez tartozó Lagrange multiplikátorok értéke nem nulla, szupport vektoroknak *(support vectors)* nevezzük.[1] A szupport vektor gépek tehát olyan kernel gépek, ahol a kernel tér tényleges dimenziója nem a tanítópontok számával (P), hanem a szupport vektorok számával (P_s) egyezik meg. Ez jelentős egyszerűsítést jelenthet a válasz számításában, különösen, ha $P_s \ll P$. A szupport vektor gépeknek ezt a tulajdonságát ritkasági *(sparseness)* tulajdonságnak nevezzük.

Azt is észrevehetjük, hogy azon pontokra, ahol $\alpha_i > 0$, teljesül a

$$d_i \left(\mathbf{w}^{*T}\mathbf{x}_i + b^*\right) = 1 \tag{7.42}$$

[1] Úgy is mondhatjuk, hogy ezek a mintapontok „tartják" a megoldást ezért ezek a mintapontok a tartó vektorok vagy támasztó vektorok *(support vectors)*. A tartó vektor elnevezés azonban a magyar nyelvű szakirodalomban egyelőre nem terjedt el, ezért a könyvben a szupport vektor elnevezést használjuk.

összefüggés, vagyis ezek a pontok – tehát a szupport vektorok – lesznek az elválasztó hipersíkhoz legközelebb lévő mintapontok. Itt $\mathbf{w}^*$ már az optimális Lagrange együtthatókkal meghatározható súlyvektor:

$$\mathbf{w}^* = \sum_{i=1}^{P} \alpha_i^* d_i \mathbf{x}_i \tag{7.43}$$

A szupport vektoroknál (7.42) alapján a (7.24) és a (7.25) egyenlőtlenségek egyenlőségként állnak fenn.

A maximális margójú lineáris szeparálás általánosító képessége

Az előzőekben láttuk, hogy egy tanuló rendszer létrehozásánál a konstrukciós lépések meghatározásán túl fontos kérdés, hogy a létrehozott rendszert minősíteni is tudjuk. A 3. fejezetben részletesebben tárgyaltuk a minősítés lehetőségeit és bemutattuk, hogy a minősítéshez valójában az ún. valódi kockázat meghatározására lenne szükség. Azt is láttuk, hogy a valódi kockázatot – a mintapont-eloszlás ismeretének hiányában – nem tudjuk meghatározni. A statisztikus tanuláselmélet azonban megadja azokat a feltételeket, melyek fennállta esetén a valódi kockázatnak legalább egy felső korlátja megadható.

A (3.31) összefüggés osztályozási feladatra adott felső korlátot. A felső korlát két tagból állt: egyrészt a tanítópontokban meghatározható eredő hibából, a tapasztalati kockázatból, és egy ún. konfidencia tagból, ami a megoldás komplexitásától függ. A statisztikus tanuláselmélet a komplexitás mértékeként a VC-dimenziót használja. Ha tehát egy megoldás VC-dimenziójára tudunk mondani valamit, a valódi kockázat egy felső korlátja megadható. A felső korlát (3.31) összefüggése alapján azt is láttuk, hogy olyan megoldásra van szükségünk, amely egyidejűleg mindkét tagot minimalizálja, tehát a tapasztalati hibát úgy próbálja minél kisebbre leszorítani, hogy közben a VC-dimenzió értéke is minél kisebb legyen. A megoldás elve a strukturális kockázatminimalizálás (SRM elv) volt. A strukturális kockázat minimalizálás elve szerint azt a legkevésbé komplex megoldást keressük, melynél a két tag összege minimumot ad.

A lineáris osztályozó szupport vektor gép, azon túl, hogy a tanítópontok hibátlan osztályozását kívánja elérni, még azt is célnak tekinti, hogy a szeparálás maximális biztonsággal, maximális margó mellett történjen meg. A maximális margó azt az intuitív benyomást kelti, hogy a szupport vektor gép által szolgáltatott megoldás a lineáris szeparálások között kitüntettet, netán optimális. Az alábbiakban bemutatjuk, hogy ez nemcsak intuitív benyomás. A maximális margó biztosítása mellett a megoldás komplexitásának mértékére, a VC-dimenzióra is adható egy felső korlát. Ennek segítségével a lineáris szupport vektor gép általánosítóképességéről is tudunk állítani valamit.

A statisztikus tanuláselmélet eredményei szerint [Vap95, Vap98] a

$$\mathbf{w}^{*T}\mathbf{x}_i + b^* = 0 \tag{7.44}$$

által megadott optimális szeparáló hipersík VC-dimenziójára az alábbi felső ko-

lát adható:

$$h \leq \min\left(\left\lceil \frac{R^2}{r^2} \right\rceil, N\right) + 1\,, \tag{7.45}$$

ahol N a bemeneti tér dimenziója, r a szeparációs tartalék, a margó, R pedig az összes bementi vektort tartalmazó legkisebb átmérőjű gömb sugara. Az összefüggésben a $\lceil . \rceil$ művelet azt a legkisebb egészet jelöli, ami nagyobb vagy egyenlő, mint a művelet argumentuma.

Bár egy függvényosztály VC-dimenzióját általános esetben nehéz meghatározni, N-dimenziós szeparáló hipersík esetén könnyen megmutatható, hogy a VC-dimenzió értéke $h = N + 1$. A (7.45) összefüggéssel megadott felső korlát ugyanakkor azt mondja, hogy maximális margójú szeparáló hipersík mellett – ha az r margó értéke elegendően nagy – a VC-dimenzió értéke lehet $(N + 1)$-nél kisebb is. Ennek különösen nemlineáris esetben van jelentősége, ahogy ezt a későbbiekben látni fogjuk. Az a tény, hogy egy felső korlát adható a VC-dimenzióra azt jelenti, hogy a megoldás általánosítóképessége a bemeneti tér dimenziójától függetlenül kontrol alatt tartható a szeparálás tartalékának megfelelő megválasztásával.

A tartalék (7.29) összefüggése szerint a margó fordítottan arányos a súlyvektor normájával. A (7.33) Lagrange egyenlet olyan optimalizálási feladatot fogalmaz meg, ahol a tanítópontok megfelelő osztályozásán túl a súlyvektor normanégyzetének minimumát is biztosítani szeretnénk. A szupport vektor gépeknél az osztályozási hiba minimuma mellett a maximális tartalék, vagyis a minimális VC-dimenzió elérésére törekszünk, összhangban a strukturális kockázat minimalizálási elvvel. A szupport vektor gépekre ezért a statisztikus tanuláselméletnek a 3. fejezetben összefoglalt eredményei alkalmazhatók. Ez lényeges különbség, ha a szupport vektor gépeket az előzőekben bemutatott ún. klasszikus hálókkal, de elsősorban a többrétegű perceptronnal hasonlítjuk össze.

Az MLP hálózatoknál a komplexitás a rejtett réteg méretétől függ. A háló struktúrájának rögzítése a rejtett réteg méretét is meghatározza. A tanítás során ezen már nem változtatunk. Ezzel szemben a szupport vektor gépeknél nem a szabad paraméterek száma határozza meg a komplexitás mértékét, hanem a VC-dimenzió, melynek felső korlátja a maximális margó értékével befolyásolható. A kétféle megközelítés közötti alapvető különbség tehát, hogy míg az MLP-nél csak a tapasztalati hiba, addig a szupport vektor gépeknél mind a tapasztalati hiba, mind az általánosítóképeséget befolyásoló komplexitásmérték, a VC-dimenzió minimalizálása megtörténik.

7.3.2. A lineárisan nem szeparálható feladat lineáris megoldása

Az előzőekben tárgyalt maximális margójú lineáris szeparálásra képes megoldás az alapelvek bemutatására alkalmas. A gyakorlatban azonban ritkán találkozunk olyan feladatokkal, melyek úgy szeparálhatók lineárisan, hogy még osztályozási tartalék is biztosítható. Általában vagy csak olyan lineáris szeparálás lehetséges, ahol a két osztály közötti biztonsági sávban is lehetnek tanítópontok, vagy

lineáris szeparálás hibamentesen nem is lehetséges. A következőkben olyan lineáris megoldást mutatunk be, ahol megfelelő kompromisszum alapján dől el a biztonsági sáv mérete és a hibás osztályozás valószínűsége.

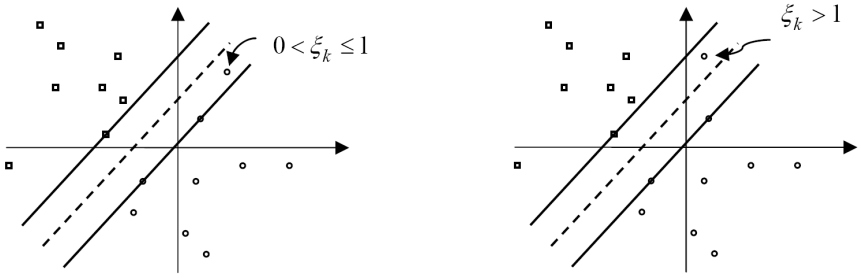
Ha megengedjük, hogy a biztonsági sávban is legyenek tanítópontok, miközben továbbra is cél a lehető legnagyobb margó biztosítása, ún. lágy vagy szoft margójú megoldásról beszélünk. Azoknál a pontoknál, amelyek a biztonsági sávon belül helyezkednek el a maximális margójú osztályozást biztosító

$$d_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 \tag{7.46}$$

egyenlőtlenség nem áll fenn. Az ilyen mintapontokra vonatkozó, az előző egyenlőtlenségnek megfelelő formális matematikai kapcsolat ún. gyengítő *(slack)* ξ_i változók bevezetésével lehetséges. A gyengítő változók bevezetése lehetővé teszi, hogy a (7.46) összefüggés az egyes tanítópontoknál különböző mértékben gyengítve érvényesüljön. Ennek megfelelően az összes pontra most a következő egyenlőtlenség írható fel:

$$d_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \xi_i \quad (i = 1,2,\ldots,P) \,. \tag{7.47}$$

Azon tanítópontoknál, ahol $\xi_i = 0$, visszakapjuk az alapfeladatot. Ha $0 < \xi_i < 1$, az adott tanítópont a hipersík megfelelő oldalán, de a biztonsági sávban helyezkedik el, ha pedig $\xi_i > 1$ az adott tanítópont a sík ellenkező oldalán (a hibás oldalon) van. Ezt illusztrálja a 7.4. ábra.



(a) A mintapont osztályozása helyes, de a mintapont a biztonsági sávba esik

(b) A mintapont osztályozása hibás

7.4. ábra. A gyengítő változók használata

Az optimális hipersíkot úgy kell meghatározni, hogy a hibás osztályozások száma minimális legyen, miközben továbbra is törekszünk a lehető legnagyobb margó elérésére. A minimalizálandó kifejezés – amit a továbbiakban $J(\mathbf{w})$-vel jelölünk – ennek megfelelően két tagból áll:

$$J(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{P} \xi_i \,, \tag{7.48}$$

ahol C a két tag közötti kompromisszumot beállító együttható. Ha $C = 0$, visszakapjuk az előző, gyengítő változó nélküli esetet. Ha C értéke kicsi, a minimalizálandó összefüggésben a második tag súlya kicsi, vagyis nem nagyon

büntetjük, ha egy tanítópont a margón belülre, netán a rossz oldalra kerül. Ez azt jelenti, hogy a biztonsági sáv szélesebb lehet, de több pont kerül a sávon belülre vagy a rossz oldalra. Ha C értékét növeljük, ezeket az eseteket jobban büntetjük, ami keskenyebb biztonsági sávot eredményez, hiszen ekkor nyilván kevesebb pont eshet a sávon belülre.

A megoldás most is a megfelelő Lagrange egyenlet felírását igényli, hiszen itt is feltételes szélsőérték-keresési feladattal állunk szemben. Egyfelől (7.48) minimumát kívánjuk elérni, de közben a (7.47) által a tanítópontokra megfogalmazott egyenlőtlenségeket is teljesíteni kívánjuk. Ennek megfelelően a Lagrange egyenlet:

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma},b)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\sum_{i=1}^{N}\xi_i-\sum_{i=1}^{N}\alpha_i\left\{d_i\left(\mathbf{w}^T\mathbf{x}+b\right)-1+\xi_i\right\}-\sum_{i=1}^{N}\gamma_i\xi_i\,. \tag{7.49}$$

Melyet $\mathbf{w}$, b és ξ_i-k szerint minimalizálni, míg α_i-k és γ_i-k szerint maximalizálni kell. A feladatot most is két lépésben oldhatjuk meg. Először a Lagrange egyenlet $\mathbf{w}$, b és $\boldsymbol{\xi}$ szerinti deriváltját írjuk fel:

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma},b)}{\partial\mathbf{w}}=0\quad\longrightarrow\quad\mathbf{w}=\sum_{i=1}^{P}\alpha_i d_i\mathbf{x}_i\,, \tag{7.50}$$

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma},b)}{\partial b}=0\quad\longrightarrow\quad\sum_{i=1}^{P}\alpha_i d_i=0\,, \tag{7.51}$$

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma},b)}{\partial\boldsymbol{\xi}}=0\quad\longrightarrow\quad\gamma_i+\alpha_i=C\,. \tag{7.52}$$

Ezt követően a gradiensek értékének nullává tételével kapott összefüggéseket behelyettesítve a Lagrange egyenletbe kapjuk a (7.38) összefüggéssel megegyező duális feladatot:

$$Q(\boldsymbol{\alpha})=\sum_{i=1}^{P}\alpha_i-\frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}\alpha_i\alpha_j d_i d_j\mathbf{x}_i^T\mathbf{x}_j\,, \tag{7.53}$$

ahol

$$\sum_{i=1}^{P}d_i\alpha_i=0\,,\quad 0\le\alpha_i\le C\quad(i=1,2,\ldots,P)\,. \tag{7.54}$$

és melyből az α_i-k most is kvadratikus programozással határozhatók meg. Figyeljük meg, hogy a gyengítő változók bevezetése az optimalizálási feladat duális alakját csak a Lagrange multiplikátorokra vonatkozó feltételekben módosítja. Az osztályozó eredménye is megegyezik a lineárisan szeparálható feladatra megadottakkal (7.41). Ennek megfelelően itt is lesznek nulla értékű α_i-k, melyekhez tartozó tanítópontok nem vesznek részt a kimenet előállításában. Ha pedig $\alpha_i\neq 0$, a megfelelő tanítópontok szupport vektorok lesznek, melyekre a

$$d_i\left(\mathbf{w}^{*T}\mathbf{x}_i+b^*\right)-1+\xi_i=0 \tag{7.55}$$

egyenlőség áll fenn. Ez azt jelenti, hogy most a szupport vektorok sem feltétlenül a margó határán helyezkednek el. A margóhoz viszonyított helyzetüket a ξ_i értékek befolyásolják.

A (7.48) minimalizálandó kritériumban szereplő C konstanst hiperparaméternek is szokás nevezni. Szerepe – ahogy az előbb említettük – az, hogy a súlyvektor hosszának, illetve a tanító mintákra számított osztályozási hibának a viszonyát beállítsa. C meghatározása nem könnyű feladat, általában kereszt kiértékeléssel történhet.

A szoft margójú lineáris szeparálás általánosító képessége

A gyengítő változó bevezetése azt eredményezte, hogy az összes pont már nem esik a biztonsági sávon kívülre. Ennek a megoldás általánosítóképességére is lesz hatása. A VC-dimenzió (7.45) összefüggése a szoft margójú megoldásra már nem lesz érvényes, így nem lesz érvényes a valódi kockázat felső korlátját megadó (3.31) összefüggés sem. Azonban a lágy margójú megoldás általánosítási hibájára is származtattak felső korlátokat, melyek figyelembe veszik a gyengítő változók értékeit is. Ezek bemutatásával itt nem foglakozunk. A származtatás és az eredmények az irodalomban [Cris00, Sha02] megtalálhatók.

7.3.3. Nemlineáris szeparálás

A valós osztályozási feladatok túlnyomó része nem szeparálható lineárisan. A lineárisan nem szeparálható feladatoknál – ahogy ezt már többször láttuk – egy megfelelő (és általában dimenzionövelő) $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$ nemlineáris transzformációval lineárisan szeparálhatóvá alakítjuk a feladatot. Az SVM előnyeinek kihasználására az így kapott jellemzőtérben az előzőeknek megfelelő optimális (maximális margójú) megoldást keressük.

A nemlineárisan szeparálható feladat SVM-el történő megoldásában tehát két lépés szerepel:

- nemlineáris transzformáció a bemeneti térből a jellemzőtérbe,
- a jellemző térben az optimális lineáris szeparáló hipersík meghatározása.

Ez utóbbi lépést azonban itt sem a jellemzőtérben, hanem az ebből származtatott kernel térben oldjuk meg.

A nemlineáris megoldás a lineáris eset megoldásából egyszerűen az $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$ helyettesítéssel nyerhető. Az optimális hipersíkot a következő alakban keressük:

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b = 0 \,. \tag{7.56}$$

Ennek alapján a korábbi lineárisan szeparálható esetre kapott eredményünknek megfelelően az optimális súlyvektor:

$$\mathbf{w}^* = \sum_{i=1}^{P} \alpha_i^* d_i \boldsymbol{\varphi}(\mathbf{x}_i) \,, \tag{7.57}$$

ahol $\boldsymbol{\varphi}(\mathbf{x}_i)$ az $\mathbf{x}_i$ mintához tartozó jellemző *(feature)* vektor. Az optimális eltolásérték *(bias)* (b^*) a $\mathbf{w}^{*T}\boldsymbol{\varphi}(\mathbf{x}_s) \pm b^* = \pm 1$ egyenletből számítható, ahol $\mathbf{x}_s$ egy szupport vektor. A jellemzőtérben megkonstruált szeparáló felületet tehát a következő egyenlet adja meg:

$$\sum_{i=1}^{P} \alpha_i^* d_i \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}) + b^* = 0\,. \tag{7.58}$$

Látható, hogy a megoldáshoz most is csak az α_i Lagrange multiplikátorok meghatározása szükséges, amit a (7.38) összefüggésnek megfelelő duális feladat kvadratikus programozással történő megoldásával nyerhetünk. A különbség mindössze annyi, hogy a duális feladatnál $\mathbf{x}$ helyére mindenhol $\boldsymbol{\varphi}(\mathbf{x})$ kell kerüljön. Ha a jellemzőtérben maximális margójú megoldás kapható, akkor a Lagrange multiplikátorokra az $0 \leq \alpha_i$ feltétel, ha csak a gyengített változat, akkor az $0 \leq \alpha_i \leq C$ feltétel teljesül.

A $\boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x})$ szorzatot a kernel trükknek megfelelően egy magfüggvényként írjuk fel, azaz

$$K(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x})\,. \tag{7.59}$$

A nemlineáris osztályozó tehát:

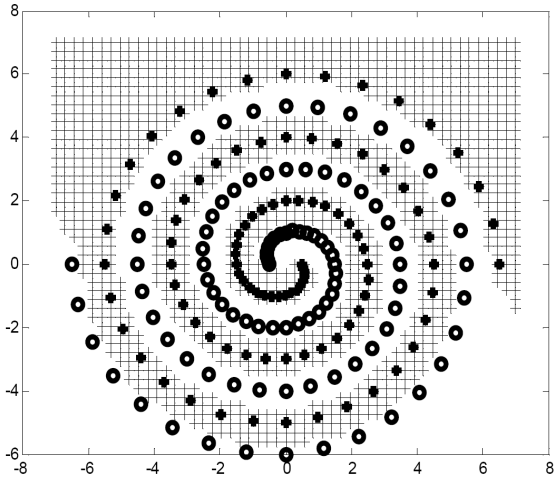
$$y(\mathbf{x}) = \operatorname{sign}\left[\sum_{i=1}^{P} \alpha_i^* d_i K(\mathbf{x}_i, \mathbf{x}) + b^*\right] \tag{7.60}$$

alakban adja meg egy $\mathbf{x}$ bemenetre a választ. Az összegzés az összes tanítópontra vonatkozik, valójában azonban csak a szupport vektorokra kell elvégezni, hiszen a többi esetben $\alpha_i = 0$. A szupport vektor gépek tehát nemlineáris esetben is ritka *(sparse)* megoldást eredményeznek.

A szupport vektor gépes megoldás előnye a kerneles megoldások előnyéhez hasonlóan itt is azáltal jelentkezik, hogy a feladatot nem a jellemzőtérben oldjuk meg, hanem a kernel térben, azáltal, hogy a $\boldsymbol{\varphi}(\mathbf{x})$ nemlineáris transzformációt nem közvetlenül, hanem csak a kernel függvényen keresztül, közvetve definiáljuk.

A 7.5. ábrán a nemlineáris SVM osztályozás illusztrálására gyakran használt kettős spirál feladat egy megoldását mutatjuk be. A feladatot, mint benchmark problémát gyakran használják osztályozó rendszerek, köztük neuronhálók teljesítőképességének összehasonlítására. Ezt az indokolja, hogy bár a feladat nagyon egyszerűnek tűnik, megoldása nem könnyű, pl. egy MLP számára különösen nehéz.

7.1 példa. Példaként oldjuk meg a XOR problémát szupport vektor géppel. Minthogy a feladat lineárisan nem szeparálható, csak a nemlineáris változat használható. Első lépésként meg kell választani a kernel függvényt. A feladat viszonylagos egyszerűsége miatt az eddig említett kernel függvények bármelyike használható. Jelen esetben használjunk polinomiális kernelt: $K(\mathbf{x}, \mathbf{x}) = (\mathbf{x}^T\mathbf{x} + + 1)^2$. A polinomiális kernel előnye, hogy ebben az egyszerű esetben még a



7.5. ábra. A kettős spirál probléma megoldása SVM-el

jellemzőtérre való leképezést biztosító $\boldsymbol{\varphi}(\mathbf{x})$ függvények is meghatározhatók. A jellemzővektorra a következő adódik:

$$\boldsymbol{\varphi}(\mathbf{x}) = \begin{bmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1x_2 & x_1^2 & x_2^2 \end{bmatrix} . \tag{7.61}$$

Látható, hogy a jellemzőtér egy 6-dimenziós tér (5 dimezió + az eltolás). Az osztályozása feladat megoldásához fel kell írnunk és meg kell oldanunk a duális egyenletet.

$$Q(\boldsymbol{\alpha}) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_i\alpha_j d_i d_j K(\mathbf{x}_i^T, \mathbf{x}_j) , \tag{7.62}$$

a következő feltételek mellett:

$$\sum_{i=1}^{4} d_i\alpha_i = \alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0 \quad \text{és} \quad 0 \leq \alpha_i \quad (i = 1, \ldots, 4) . \tag{7.63}$$

A kvadratikus egyenlet felírásánál feltételeztük, hogy a bemenetek és a kimenetek nem bináris, hanem bipoláris megadásúak, vagyis a XOR probléma leképezése a következő:

A kvadratikus programozási feladat megoldásához fel kell írni a kernel mátrixot. Mivel a XOR problémánál összesen 4 pontunk van, melyek mind tanítópontok, a kernel mátrixra a következő adódik:

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix} \tag{7.64}$$

Tanítópont index i	$[\mathbf{x}_i, \mathbf{x}_2]$	d
1	[1,1]	1
2	$[1,-1]$	-1
3	$[-1,-1]$	1
4	$[-1,1]$	-1

A kvadratikus programozás eredményeképp azt kapjuk, hogy mind a négy a azonos értékű lesz, és mind különbözik nullától: $\alpha_1^* = \alpha_2^* = \alpha_3^* = \alpha_4^* = 1/8$. Ez azt jelenti, hogy az összes pont szupport vektor, ami a feladat ismeretében nem meglepő. A szupport vektoroknál a háló válasza alapján az eltolásérték is meghatározható, ami $b^* = 0$ -ra adódik. A háló válasza a kernel térben tehát:

$$y(\mathbf{x}) = \text{sign}\left[0{,}125 \sum_{i=1}^{4} d_i \left(\mathbf{x}_i^T \mathbf{x} + 1\right)^2\right] . \tag{7.65}$$

A szupport vektor gépeknél a jellemzőtérbeli reprezentáció és megoldás általában nem határozható meg, és nincs is rá szükség. Ebben az egyszerű példában azonban ismerjük a jellemzőtérre való leképezést biztosító $\boldsymbol{\varphi}(\mathbf{x})$ függvényeket, és a Lagrange multiplikátorok ismeretében a $\mathbf{w}^*$ jellemzőtérbeli súlyvektor is meghatározható:

$$\mathbf{w}^* = \sum_{i=1}^{4} \alpha_i^* d_i \mathbf{x}_i = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \end{bmatrix}^T \tag{7.66}$$

Ennek ismeretében a megoldás a jellemzőtérben is megkapható. A döntési függvény

$$g(\mathbf{x}) = x_1 x_2 . \tag{7.67}$$

Arra az érdekes eredményre jutottunk, hogy bár a választott kernel által meghatározott nemlineáris transzformáció hatdimenziós jellemzőteret definiál, a döntés egyetlen dimenzió alapján meghozható, melyet a két bementi komponens szorzata határoz meg. Ez egyben arra is példa, hogy előfordulhat olyan eset, amikor a „dimenzionövelő" nemlineáris transzformáció valójában kisebb dimenziójú térbe képez le, mert a lineáris szeparálás ott is lehetséges. Ez természetesen általánosan nem igaz, de feladattól függően, ha megfelelő nemlineáris transzformációt választunk, előfordulhat. A nemlineáris SVM a jellemzőtérben biztosítja a maximális margót, ami itt $\sqrt{2}$-re adódik.

7.3.4. Szupport vektor gépek regresszióra

A szupport vektor gépek az osztályozási feladatokon túl regressziós feladatokra is alkalmazhatók. A regressziós vagy függvényapproximációs feladatok esetén azonban nem definiálható az osztályozásnál értelmezett margó, azaz az osztályozási tartalék fogalma. Ennek megfelelően szükség van a regressziós feladat egy

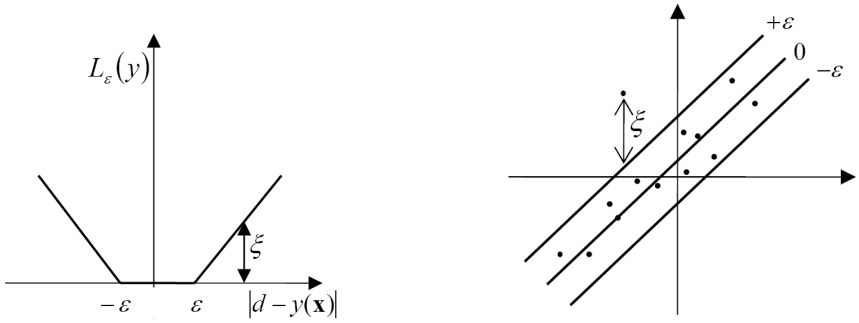
olyan értelmezésére, felírására, ami lehetővé teszi az SVM kiegészítő feltételének felhasználását.

Hasonlóan az osztályozónál bemutatottakhoz itt is lehetőség lenne először a lineáris esetet, majd annak nemlineáris kiterjesztését tárgyalni, de a tömörebb tárgyalás kedvéért itt közvetlenül a nemlineáris (azaz a jellemzőtérben lineáris) regressziót mutatjuk be.

Az osztályozós és a regressziós feladatok között a kapcsolat viszonylag könnyen megteremthető, ha a függvényapproximációs feladatoknál az átlagos négyzetes hiba helyett egy ún. ε érzéketlenségi sávval rendelkező abszolútérték hibafüggvényt *(ε-insensitive loss function)* alkalmazunk az eltérés mérésére.

$$L_\varepsilon(y) = \begin{cases} 0 & \text{ha } |d - y(\mathbf{x})| < \varepsilon\,, \\ |d - y(\mathbf{x})| - \varepsilon & \text{egyébként.} \end{cases} \tag{7.68}$$

Az ε érzéketlenségű veszteségfüggvény 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 egy olyan osztályozási feladatnak, ahol a helyes osztályozás hibája 0, míg biztonsági sávba eső vagy a rosszul osztályozott pontokat pedig gyengítő (ξ) változók bevezetésével büntetjük. Ennek megfelelően az alábbiakban bemutatott regressziós megoldást a lineárisan nem szeparálható osztályozási feladatra kapott eredményekkel célszerű összevetni. Hibafüggvényként az abszolútérték-függvény használatát az is indokolja, hogy a kiugró adatokra *(outlier)* való érzékenysége az abszolútérték-függvénynek kisebb, mint a gyakran alkalmazott négyzetes hibafüggvénynek. Ennek ellenére szokás ε érzéketlenségi sávú négyzetes hibafüggvényt is alkalmazni. Jelen tárgyalásnál azonban megmaradunk a Vapnik féle alapmegoldásnál, ahol az abszolútérték-függvény szerepel.



(a) (b)

7.6. ábra. Az ε-érzéketlenségi sávval rendelkező abszolút érték hibafüggvény és alkalmazása

Írjuk fel a háló kimenetét. Ez most is nemlineáris bázisfüggvények lineáris kombinációjaként áll elő, azaz a $\boldsymbol{\varphi}(\mathbf{x})$-ek által reprezentált ($M$-dimenziós)

jellemzőtérben lineáris megoldást keresünk:

$$\begin{aligned} &y(\mathbf{x}) = \textstyle\sum_{j=1}^{M} w_j \varphi(\mathbf{x}) + b = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b\,, \\ &\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_M \end{bmatrix}^T\,, \\ &\boldsymbol{\varphi}(\mathbf{x}) = \begin{bmatrix} \varphi_1(\mathbf{x}) & \varphi_2(\mathbf{x}) & \dots & \varphi_M(\mathbf{x}) \end{bmatrix}^T\,. \end{aligned} \tag{7.69}$$

A kielégítendő feltételek (a gyengítő változók bevezetésével) a következők:

$$\begin{aligned} y_i - \left(\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b\right) &\le \varepsilon + \xi_i\,, \\ \left(\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b\right) - y_i &\le \varepsilon + \xi_i'\,, \\ \xi_i &\ge 0\,, \\ \xi_i' &\ge 0\,. \end{aligned} \qquad (i = 1,2,\dots,P) \tag{7.70}$$

A minimalizálandó költségfüggvény:

$$J\left(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\xi}'\right) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_{i=1}^{P}\left(\xi_i + \xi_i'\right)\right) \qquad (i = 1,2,\dots,P) \tag{7.71}$$

a (7.70) egyenlőtlenségek által megfogalmazott mellékfeltételekkel.
A megfelelő Lagrange egyenlet az $\alpha_i, \alpha_i', \gamma_i, \gamma_i' \ge 0$ Lagrange multiplikátorok felhasználásával:

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}', \boldsymbol{\alpha}, \boldsymbol{\alpha}', \boldsymbol{\gamma}, \boldsymbol{\gamma}') = C\sum_{i=1}^{P}\left(\xi_i + \xi_i'\right) + \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ - \sum_{i=1}^{P}\alpha_i\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b - y_i + \varepsilon + \xi_i\right] \\ - \sum_{i=1}^{P}\alpha_i'\left[y_i - \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) - b + \varepsilon + \xi_i'\right] \\ - \sum_{i=1}^{P}\left(\gamma_i\xi_i + \gamma_i'\xi_i'\right)\,. \end{aligned} \tag{7.72}$$

A Lagrange függvényt kell $\mathbf{w}$, b és a $\boldsymbol{\xi}$ és $\boldsymbol{\xi}'$ gyengítő változók szerint minimalizálni, $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}'$ valamint $\boldsymbol{\gamma}$ és $\boldsymbol{\gamma}'$ szerint maximalizálni. Elvégezve a deriválást, és a deriváltak értékét nullává téve, a következő összefüggéseket kapjuk:

$$\begin{aligned} &\mathbf{w} = \textstyle\sum_{i=1}^{P}(\alpha_i - \alpha_i')\boldsymbol{\varphi}(\mathbf{x}_i)\,, \\ &\gamma_i = C - \alpha_i\,, \\ &\gamma_i' = C - \alpha_i'\,. \end{aligned} \tag{7.73}$$

Behelyettesítve ezeket a Lagrange egyenletbe jutunk a duális feladatra:

$$\begin{aligned} Q(\boldsymbol{\alpha}, \boldsymbol{\alpha}') = & \sum_{i=1}^{P} d_i(\alpha_i - \alpha_i') - \varepsilon \sum_{i=1}^{P} (\alpha_i + \alpha_i') \\ & - \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} (\alpha_i - \alpha_i')(\alpha_j - \alpha_j') K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{7.74}$$

a

$$\sum_{i=1}^{P} (\alpha_i - \alpha_i') = 0\,, \quad 0 \le \alpha_i \le C\,, \quad 0 \le \alpha_i' \le C \quad (i = 1,2,\ldots,P) \tag{7.75}$$

megkötésekkel. Az eredmény a kvadratikus programozással meghatározható α_i Lagrange multiplikátorok, a kiszámított b és a $K(\mathbf{x}, \mathbf{x}_i)$ kernel függvény alapján:

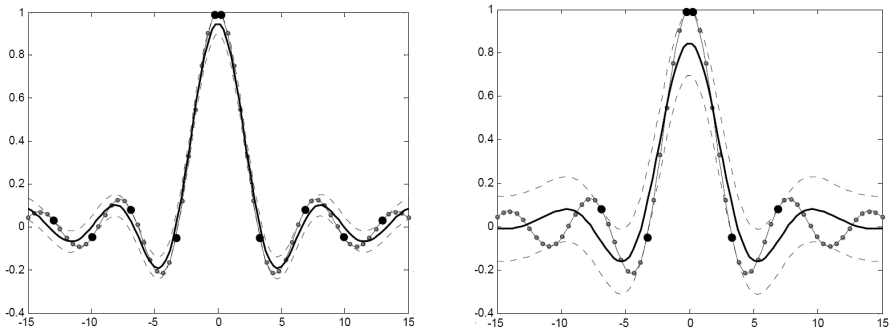
$$y(\mathbf{x}) = \sum_{i=1}^{P} (\alpha_i - \alpha_i') K(\mathbf{x}, \mathbf{x}_i) + b\,. \tag{7.76}$$

Csakúgy, mint az osztályozási esetben, az SVM modell most is ritka, ugyanis csak azok a tanítópontok vesznek részt a kimenet előállításában, melyekre $(\alpha_i - \alpha_i') \neq 0$. Fontos megjegyezni, hogy most is a kernel függvényből indulunk ki, így a jellemzőtér bázisfüggvényeit a kernel függvény implicit módon definiálja. Ez a jellemzőtér azonban nagyon nagy dimenziós, akár végtelen dimenziós is lehet, ezért míg lineáris esetben a probléma az eredeti térben is megoldható, nemlineáris esetben a jellemzőtérben már nem minden esetben lehetséges a megoldás.

Figyeljük meg, hogy a regressziós megoldásnál az SVM kialakításában az osztályozós esethez képest eggyel bővül a hiperparaméterek száma, hiszen az ε érzéketlenségi paraméter megadására is szükség van. Az ε hiperparaméternek az approximációra gyakorolt hatása zajmentes tanítópontok esetén szemléletesen is értelmezhető (7.7. ábra).

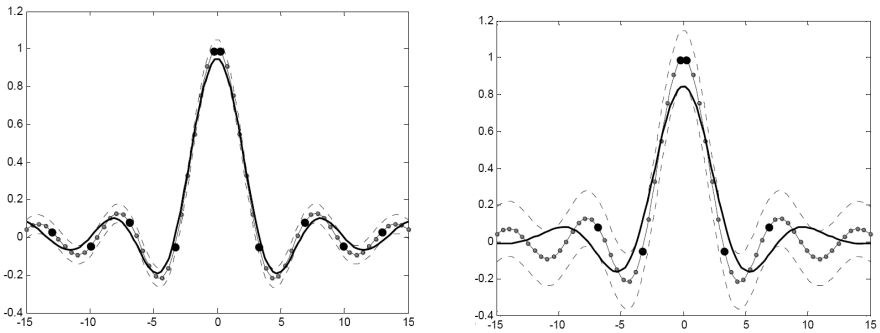
Az ábrán látható, hogy ha a megoldás köré egy ε szélességű sávot húzunk úgy, hogy az SVM válaszfüggvényét $\pm\varepsilon$-nal eltoljuk, akkor minden tanítópont ezen a sávon belül helyezkedik el. A sávhatárra eső pontok lesznek a szupport vektorok (az ábrán a nagyobb fekete pontok). Az ábra azt is mutatja, hogy kisebb szélességű sáv több, a nagyobb szélességű pedig kevesebb szupport vektort eredményez. Ezt úgy biztosítja az SVM, hogy minél nagyobb ε értéke, annál simább megoldást kapunk. Ezt a hatást illusztrálja a 7.8. ábra is. Ezen az ábrán is szerepel egy ε szélességű sáv, ami azonban most az approximálandó függvényt veszi közre, és annak $\pm\varepsilon$-nal történő eltolásával kapható. Az SVM zajmentes esetben mindig olyan megoldást ad, amely ebbe az ε szélességű sávba belefér, miközben a válasz a lehető legsimább. Látható, hogy a szélesebb sáv simább, a keskenyebb kevésbé sima megoldást eredményez.

Fentiek alapján ε hatása zajmentes esetben jól követhető, és ennek alapján a megfelelő ε érték megválasztható, hiszen minél kisebb ez az érzéketlenségi sáv,



(a) $\varepsilon = 0{,}05$ (b) $\varepsilon = 0{,}15$

7.7. ábra. Az ε-érzéketlenségi sáv szerepe a függvényapproximációban. A vastagabb folytonos vonal a háló válaszát, a szaggatott vonalak az ε szélességű sáv széleit mutatják. A vékonyabb folytonos vonal az approximálandó függvényt a rajta lévő tanítópontokkal (kisebb pontok) mutatja. A nagyobb fekete pontok a szupport vektorok.



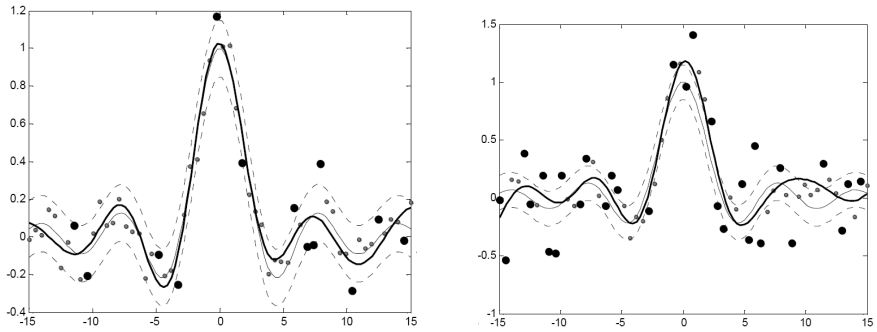
(a) $\varepsilon = 0{,}05$ (b) $\varepsilon = 0{,}15$

7.8. ábra. Az ε-érzéketlenségi sáv hatása a válasz simaságára függvényapproximációban. A vastagabb folytonos vonal a háló válaszát, a szaggatott vonalak az ε szélességű sáv széleit mutatják. A vékonyabb folytonos vonal az approximálandó függvényt a rajta lévő tanítópontokkal mutatja. A nagyobb fekete pontok a szupport vektorok.

annál pontosabban próbálja a modell a tanítópontokra illeszteni a megoldást. A valós feladatokban azonban szinte mindig zajos bemenetekkel van dolgunk. Ebben az esetben a kis ε arra késztetné a modellt, hogy minden pontot jól (kis hibával) közelítsen, ami túltanuláshoz vezethet, holott a cél itt éppen egy simább függvény illesztése lenne. Nagyobb zaj esetén tehát célszerű nagyobb ε-t, szélesebb érzéketlenségi sávot használni. Zajos tanítópontok mellett ε hatására a 7.9. ábra mutat példát.

Az SVM minél simább válasza annak a következménye, hogy a súlyvektor normáját minimalizáljuk. A költségfüggvényben szereplő $\mathbf{w}^T\mathbf{w}$ tag osztályozós

esetben a maximális margóért, regressziós esetben pedig a sima megoldásért felelős.



(a) $\sigma_n = 0{,}01$ (b) $\sigma_n = 0{,}04$

7.9. ábra. A sinc(x) regressziója zajos adatok alapján. (Gauss kernel; $\sigma = \pi$ $C = 10$, $\varepsilon = 0{,}15$. A Gauss zaj szórása σ_n.)

A megoldás hibája, valamint komplexitása közötti kompromisszumot regressziós esetben is a C paraméter állítja be, azaz az ε és a C hiperparaméterek értékét összehangoltan kell megválasztani. Általánosságban is elmondható, hogy a szupport vektor gépek hiperparamétereinek (az ε, a C, valamint a kernel függvény paramétereinek – Gauss kernel esetén a σ szélességparaméternek) a megválasztása igen összetett feladat, mert az egyes paraméterek nem függetlenek egymástól, így a lehetséges beállítások terében kell egy jó, esetleg optimális kombinációt megtalálni. A kernel paraméterek hatása – ha Gauss kernelt választunk – hasonló a Gauss bázisfüggvényes RBF hálók szélességparaméterének hatásához: a nagy σ simább, míg a kis σ változékonyabb (erősen lokalizált) közelítést eredményez, ami hatással van például a megfelelő C értékre is. C értékét az előbbi esetben kisebb, míg az utóbbi esetben várhatóan nagyobbra kell megválasztani. A hiperparaméterek megfelelő megválasztásával számos irodalom pl. [Sch02, Cha02, Kwo03] foglalkozik.

Az SVM működése

Láttuk, hogy a nemlineáris problémák lineáris kezelésének kulcsa, hogy az eredeti feladatot egy olyan többdimenziós térbe transzformáljuk, ahol az már lineárisan megoldható. A kernel gépek mindezt a kernel térben tudják megoldani. 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 működés mechanizmusának jobb megértéséhez a kernel tér alábbi jellemzőit hangsúlyozzuk:

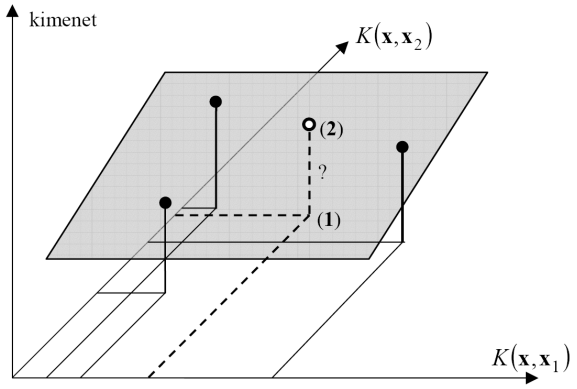
– A kernel tér a jellemzőtér, illetve az ehhez tartozó leképezés konkrét ismerete nélkül is elérhető.

– 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 kernel térbe leképezett két ponthalmazt maximális margóval elválasztó lineáris felületnek, regresszió esetén pedig a kernel térbeli pontokra illesztett hipersíknak. A tanítási lépés során tehát a hipersíkot leíró egyenlet szabad paramétereit, az α_i Lagrange multiplikátorokat és 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 tanítópontok és a hozzájuk tartozó kimenet közötti összerendelést definiálja. A működés szemléltetéséhez tekintsük a regressziós esetet. A megtanított SVM működését a 7.10. ábra szemlélteti.



7.10. ábra. A kernel térbeli lineáris megoldás illusztrációja két-dimenziós bemenet és egydimenziós kimenet esetén. A fekete pontok a tanító mintákat illusztrálják, míg a fehér pont egy kiszámítandó kimenetet illusztrál.

Az SVM modell eredményét leíró

$$y(\mathbf{x}) = \sum_{i=1}^{P} (\alpha_i - \alpha_i') K(\mathbf{x}, \mathbf{x}_i) + b \tag{7.77}$$

egyenlet 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 érték 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 lehetséges.

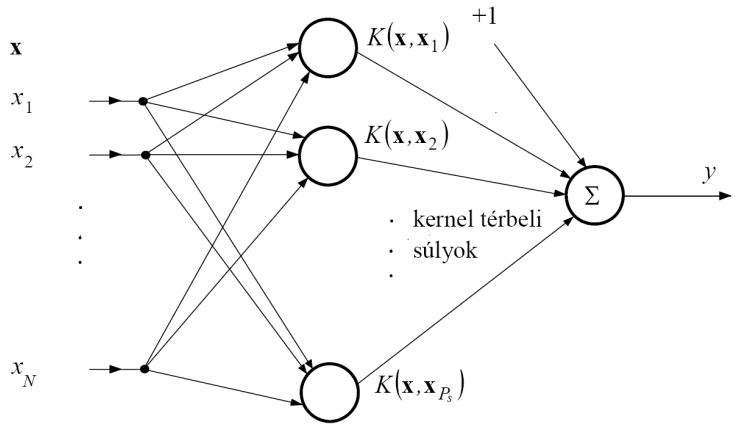
A regressziós feladat megoldása során az SVM által konstruált optimális hipersík – amennyire lehetséges – jó választ ad a tanító pontokra, azaz illeszkedik

rájuk. A modell akkor működik megfelelően, ha nemcsak a tanítópontok esetében, hanem később a teszteléshez használt pontok esetén is a hipersíkra esik a kimenet, vagy másképpen a hipersík értéke jól közelíti a valós kimenet értéket.

7.3.5. Az SVM neurális értelmezése

Bár a gyakorlatban a szupport vektor gépeket ritkán alkalmazzák neurális háló zat alakjában, ez az értelmezés hasznos, mert egyszerűbb tárgyalásmódot tesz lehetővé a pusztán matematikai megközelítésénél.

A szupport vektor gépek tanítása, és használata is matematikai számítások sorozata, de a válasz számításának összefüggése pontosan megfeleltethető egy egy-rejtett-rétegű neurális hálózatnak. A rejtett réteg tipikusan nemlineáris neuronokat tartalmaz. A 7.11. ábra egy szupport vektor gépnek megfeleltethető neurális hálózatot mutat be.



7.11. ábra. A szupport vektor gépnek megfeleltethető neurális hálózat

A bemenet egy N-dimenziós vektor. A nemlineáris kernel függvényeket a rejtett réteg neuronjai tartalmazzák. Ezen neuronok száma megegyezik a szupport vektorok számával (P_s). A hálózat válasza (y) a rejtett réteg neuronjai kimenetének súlyozott összege. Az súlyokat a tanítás során kiszámított Lagrange multiplikátor értékek határozzák meg. Ennek megfelelően, minél kisebb a hálózat, annál kevesebb számításra van szükség a válasz megadásához, így a cél a lehető legkisebb hálózat elérése.

Láthatóan a kernel gép neurális interpretációja egy struktúrájában a bázisfüggvényes hálózatokkal gyakorlatilag tökéletesen megegyező háló. Ez azonban csak formai hasonlóság, a kétféle megközelítés közötti lényeges különbséget nem szabad szem elől téveszteni.

7.3.6. Az SVM hatékonyabb megvalósításai

A Support Vector Gépek (Support Vector Machines – SVM) gyakorlati felhasználásának egyik legnagyobb akadálya az SVM tanításához szükséges kvadratikus programozás (Quadratic Programing – QP) nagy algoritmikus komplexitása, valamint a kernel mátrix nagy mérete. Az alábbiakban olyan módszereket mutatunk be röviden, melyek a számítás gyorsítását, valamint a szükséges memória méretének csökkentését célozzák. Ezt a feladatot kétféleképpen oldhatjuk meg. Ezek:

– az eredeti felírás megoldásához szükséges számítások (a kvadratikus programozás) felgyorsítása, valamit ezek kisebb tárterületen történő megoldása.

– az eredeti felírás módosítása úgy, hogy könnyebben megoldható legyen, s így gyorsabb algoritmussal hasonló tulajdonságú megoldásra vezessen. Itt olyan módosításról beszélünk, ami az eredeti SVM optimumot adja, tehát a feladatot nem, csak annak matematikai felírását változtatja. (A későbbiekben olyan SVM változatok bemutatására is sor kerül, ahol azonos elméleti háttér mellet már egy másik probléma (optimum) kerül kiszámításra.)

A QP hatékonyabb megoldása

A tanítási sebesség növeléséhez az SVM tanításban használt kvadratikus programozási lépést kell gyorsabban (és kevesebb memória felhasználásával) végrehajtani. Erre számos hatékony QP megoldó áll rendelkezésre. Ezek iteratív eljárások, amelyek az eredeti optimalizálási feladatot kisebb részfeladatok sorozataként oldják meg. A részfeladatok kiválasztásának alapját az optimalizálási feladat KKT (Karush-Kuhn-Tucker) feltételei jelentik. Azok a minták, amelyek teljesítik az optimumra vonatkozó KKT feltételeket, nem vesznek részt az optimalizálásban, míg a feltételeket megszegő pontok igen. A KKT feltételek osztályozós esetre az alábbiak $\forall i$-re:

$$\begin{aligned} \alpha_i = 0 \quad &\Longrightarrow \quad d_i y(\mathbf{x}_i) \geq 1 \\ 0 < \alpha_i < C \quad &\Longrightarrow \quad d_i y(\mathbf{x}_i) = 1 \\ \alpha_i = C \quad &\Longrightarrow \quad d_i y(\mathbf{x}_i) \leq 1 \end{aligned} \tag{7.78}$$

Egy mintapont akkor és csak akkor lehet optimum pontja (7.53)-nak, ha teljesülnek a KKT feltételek és a kernel mátrix pozitív definit.

A szupport vektor gépekhez használt három legfontosabb, azonos alapelvek szerint működő, egymást továbbfejlesztő gyorsítási algoritmus a legegyszerűbbtől a legkiforrottabbig [Sch99]:

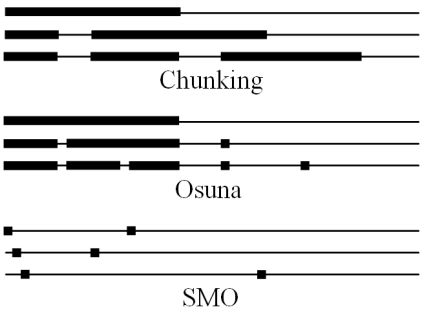
Szeletetés (Chunking). A szeletelési *(chunking)* eljárás [Vap79] alapja, hogy a nulla α_i-khez tartozó minták nem befolyásolják a megoldást. Először a minták egy részhalmazán oldjuk meg a QP-t, majd az adott alproblémából mindig a szupport vektorokat tartjuk meg. Ezekhez azt az M tanítómintát

veszünk hozzá, amelyek a leginkább megszegik a KKT-feltételeket, majd újra megoldjuk a kvadratikus optimalizálást. Ha M-nél kevesebb minta szegi meg a feltételeket, akkor mindet beillesztjük a következő alfeladatba. Az iteráció végére az összes nem nulla multiplikátort meghatározzuk, így az eredeti feladatot is megoldottuk. A kezelt alprobléma nagysága az iterációk során általában nő. A chunking nem jól használható, ha nagyon sok a tanítópont vagy sok a nem nulla α_i, hiszen ebben az esetben még mindig nagy a memóriaigény.

Osuna algoritmusa. Az előzőhöz hasonló módszer Osuna dekompozíciós módszere [Osu98], amely mindig egy állandó méretű részhalmazon dolgozik, miközben a többi változót fix értéken tartja. A kvadratikus programozást így mindig egy fix méretű munkahalmazon oldja meg, és kisebb optimalizálási feladatok sorozatán keresztül jut el a teljes megoldásig.

SMO. A Sequential Minimal Optimization (SMO) olyan dekompozíciós eljárás [Pla99], amely során iterációnként mindössze két α_i változó optimalizálása történik meg. Előnye, hogy teljesen kikerüli a QP számítást, mert a két változó analitikusan is optimalizálható. Két változó analitikus optimalizálása után a következő két α_i kiválasztására kerül sor, amely heurisztikusan történik.

A 7.12. ábra az említett három alapeljárás szeletelési stratégiáját mutatja. Mindhárom módszernél három lépést mutat az ábra, ahol a legfelső a kezdőlépés. A vízszintes vonalak minden esetben a tanító készletet reprezentálják, a megvastagított fekete szakaszok pedig az adott iterációs lépésben optimalizált Lagrange multiplikátorokat jelzik.



7.12. ábra. A legelterjedtebben használt QP megoldók „szeletelési” (dekomponálási) stratégiái

Az SVM módosítása

Az SVM módosított verziói közül a hatékonyság és a memóriaigény szempontjából Mangasarian és munkatársainak a munkái a legfontosabbak.

SSVM Az SSVM (Smooth Support Vector Machine) [Lee01a] az eredeti feladatot egy olyan optimalizálási feladatra fogalmazza át, ami egyszerű gradiens módszerekkel megoldható. Az így kapott kvadratikus hibafelület minimuma (itt már nincsenek feltételek, mint az eredeti QP-ben) megfelel az SVM által megoldott kvadratikus programozási feladat minimumának.

RSVM Az RSVM (Reduced Support Vector Machine) [Lee01b] valójában az SSVM egy kiterjesztése, ahol az összes mintapontnak csak egy részhalmazából kerülhetnek ki szupport vektorok. Az eljárás ezzel valójában tovább redukálja a megoldandó feladat méretét, hiszen az eredeti kvadratikus kernel mátrix helyett egy kisebb mátrixot kell csak előállítani. Ennek a kulcsa, hogy ellentétben a kvadratikus programozással, ami csak kvadratikus mátrixra futtatható, a gradiens módszerekkel már egy ilyen egyenletrendszer is megoldható.

7.4. SVM változatok

A szupport vektor gépek alkalmazásának legjelentősebb akadálya a nagy memória- és számítási komplexitás. Mint láttuk, ennek oka, hogy az SVM tanítása során egy nagy számításigényű kvadratikus programozási feladatot kell megoldani. Az előzőekben röviden összefoglalt néhány hatékony eljárás vagy a kvadratikus programozás gyors megvalósításával, vagy az SVM optimalizálási feladatának módosításával ér célt. Ezek a megoldások azonban mind az eredeti feladat megoldását keresik.

A probléma egy másik lehetséges megközelítése, ha az optimalizálási feladatot módosítjuk, az SVM elméleti hátterének, valamit előnyeinek lehetőség szerinti megtartása mellett. Ekkor azonban már nem az eredeti SVM feladat megoldására törekszünk. A biztonsági sáv maximalizálása, a duális felírás és a kernel trükk alkalmazása jelentik a módszer azon magját, amelyet meghagyva a számítási feladat átfogalmazásával az eredeti módszerhez hasonló kernel gépeket kaphatunk. Az SVM változatokkal szembeni fő elvárás, hogy egyszerűsítsék a szükséges számításokat, mindemellett megtartsák az eredeti eljárás jó tulajdonságait, mint a jó általánosítóképességet mind a kapott megoldás ritkaságát *(sparseness)*.

Az egyik legelterjedtebb ilyen módosítás a négyzetes hibára optimalizált szupport vektor gép, az LS-SVM (Least Squares Support Vector Machine) [Suy00, Suy02]. Ez a megoldás az SVM-től eltérően a klasszikus neuronhálóknál szokásos négyzetes hibafüggvénnyel dolgozik, továbbá a kielégítendő feltételeket egyenlőségek formájában fogalmazza meg. Ennek köszönhetően a megoldás egy lineáris egyenletrendszer eredményeként áll elő, amelynek megoldása lényegesen egyszerűbb, mint a kvadratikus programozási feladaté.

Az SVM változatok közül először a legelterjedtebb LS-SVM-et mutatjuk be, ami megfelel a statisztikából ismert ridge regressziónak, amelyet később szintén bemutatunk röviden. A legkisebb négyzetes hiba alkalmazásából adódóan az eredeti LS-SVM és a ridge regresszió elveszíti a ritkasági tulajdonságot. Megfelelő módosításokkal, kiterjesztésekkel azonban itt is elérhető, hogy ritka megoldást

kapjunk. A ritka megoldásra vezető egyik lehetséges kiterjesztés LS^2-SVM. Ennek tárgyalását a redukált bázisú kernel ridge regresszió *(reduced rank kernel ridge regression, RRKRR)* bemutatása követi, ami más megközelítést alkalmazva, de ugyancsak ritka megoldást biztosít.

7.4.1. Az LS-SVM

A Johan Suykens által kidolgozott LS-SVM [Suy02] a hagyományos szupport vektor gépek olyan módosítása, ami az idő- és erőforrás-igényes kvadratikus programozással szemben, egy lineáris egyenletrendszer megoldására vezet.

Mivel az LS-SVM a hagyományos SVM módosítása, a kiinduló ötletek, illetve összefüggések hasonlóak, ezért a továbbiakban elsősorban az alapváltozattól való eltérések bemutatására törekszünk. A fő különbség a két módszer között az ε-érzéketlenségi sávval rendelkező költségfüggvény négyzetes függvényre való lecserélése, valamint, hogy az SVM mellékfeltételeit megfogalmazó egyenlőtlenségeket egyenlőségekre cseréljük.

Az LS-SVM tárgyalásánál csak a legáltalánosabb nemlineáris változatot származtatjuk.

Az LS-SVM osztályozó

Az osztályozási feladat a már megszokott: adott egy $\{\mathbf{x}_i, d_i\}_{i=1}^P$ tanítópont-halmaz, ahol $\mathbf{x}_i \in \mathsf{R}^N$ egy N-dimenziós bemeneti vektor, illetve $d_i \in [0,1]$ az ehhez tartozó kívánt kimenet. A cél egy olyan modell létrehozása, ami jól reprezentálja a tanítópontok által leírt kapcsolatot.

Az osztályozó esetében az optimalizálási feladat az alábbi:

$$\min_{\mathbf{w},b,\mathbf{e}} J(\mathbf{w},b,\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{P} e_i^2 \tag{7.79}$$

a

$$d_i\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b\right] = 1 - e_i \quad (i = 1,2,\ldots,P) \tag{7.80}$$

mellékfeltételekkel. A fenti képletben a négyzetes hibát megadó e_i értékek a négyzetes költségfüggvényből adódó hibaértékek. Szerepük az SVM-ben bevezetett ξ_i gyengítő változóknak felel meg. Látható, hogy a feltételekben az SVM osztályozóhoz képest (lásd (7.24) összefüggés) az egyenlőtlenség helyett egyenlőség szerepel, valamint hogy a célfüggvényben a hiba négyzete (e_i^2) jelenik meg.

Az osztályozó most is a szokásos bázisfüggvényes alakban írható fel:

$$y(\mathbf{x}) = \operatorname{sign}\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b\right], \tag{7.81}$$

ahol a $\boldsymbol{\varphi}(.) : \mathsf{R}^N \to \mathsf{R}^M$ leképezés az SVM-ben is használt nemlineáris leképezés egy magasabb, M-dimenziós jellemzőtérbe.

A fenti egyenletekből az alábbi Lagrange egyenlet írható fel:

$$L(\mathbf{w},b,\mathbf{e};\boldsymbol{\alpha}) = J(\mathbf{w},b,\mathbf{e}) - \sum_{i=1}^{P}\alpha_i\left\{d_i\left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b\right] - 1 + e_i\right\}, \tag{7.82}$$

ahol az α_i értékek a Lagrange multiplikátorok, amelyek az egyenlőségi feltételek miatt itt pozitív és negatív értékeket is felvehetnek.

Az optimumra vonatkozó feltételek az alábbiak:

$$\begin{aligned}
&\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \longrightarrow \quad \mathbf{w} = \sum_{i=1}^{P} \alpha_i d_i \boldsymbol{\varphi}(\mathbf{x}_i) \\
&\frac{\partial L}{\partial b} = 0 \quad \longrightarrow \quad \sum_{i=1}^{P} \alpha_i d_i = 0 \\
&\frac{\partial L}{\partial e_i} = 0 \quad \longrightarrow \quad \alpha_i = C e_i \qquad\qquad (i = 1,2,\ldots,P) \\
&\frac{\partial L}{\partial \alpha_i} = 0 \quad \longrightarrow \quad d_i\big[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b\big] - 1 + e_i = 0 \quad (i = 1,2,\ldots,P)
\end{aligned} \tag{7.83}$$

A fenti egyenletekből egy lineáris egyenletrendszer írható fel, mely az alábbi kompakt formában is megadható:

$$\begin{bmatrix} 0 & \mathbf{d}^T \\ \mathbf{d} & \boldsymbol{\Omega} + C^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{\mathbf{1}} \end{bmatrix} , \tag{7.84}$$

ahol

$$\begin{aligned}
&\mathbf{d}^T = \begin{bmatrix} d_1 & d_2 & \ldots & d_P \end{bmatrix} \\
&\vec{\mathbf{1}}^T = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix} \\
&\boldsymbol{\alpha}^T = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_P \end{bmatrix} \\
&\Omega_{ij} = d_i d_j \boldsymbol{\varphi}(\mathbf{x}_i)^T \boldsymbol{\varphi}(\mathbf{x}_j) = d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (i,j = 1,2,\ldots,P)
\end{aligned} \tag{7.85}$$

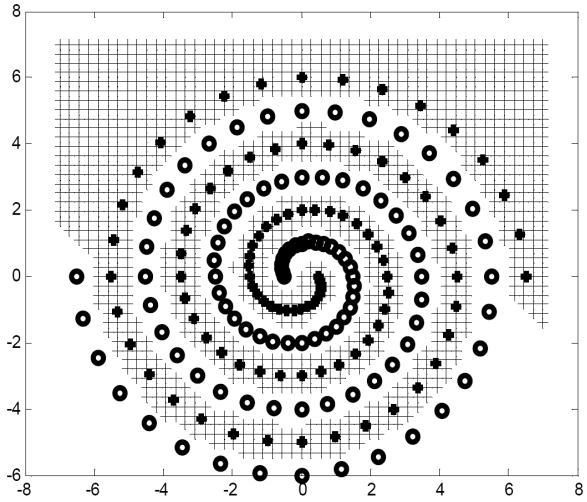
és $\mathbf{I}$ egy megfelelő méretű egységmátrix. Látható, hogy az egyenletrendszer első sora a $\sum_{i=1}^{P} \alpha_i d_i = 0$ feltételt írja le, míg a további sorok a $\dfrac{\partial L}{\partial \alpha_i} = 0$ feltételből a adódó egyenleteket írják le, behelyettesítve a (7.83)-ban megadott derivált számításokból $\mathbf{w}$-re és e_i-re származtatott összefüggéseket.

Az osztályozó válasza – hasonlóan a hagyományos SVM-hez – a következő alakban írható fel:

$$y(\mathbf{x}) = \sum_{i=1}^{P} \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i) + b\,. \tag{7.86}$$

Az α_i súlyok és a b eltolásérték *(bias)* a fenti egyenletrendszerből számíthatók. Az eredmény jellemzője, hogy a megoldásban az összes tanítópont szerepel, tehát mind szupport vektor, azaz az egyenlőség használata miatt nincsenek nulla súlytényezők. Ezzel elveszítjük az SVM ritkasági tulajdonságát, nevezetesen, hogy a bemeneti vektoroknak csak egy kis része lesz szupport vektor. Az α_i súlyok nagyság szerinti sorba rendezéséből látszik, hogy melyek a fontosabb, illetve melyek a kevésbé fontos bemeneti vektorok. Ezen alapul a később bemutatásra kerülő komplexitás redukciós metszési *(pruning)* eljárás, ami az SVM ritkasági tulajdonságát adja vissza.

A 7.13. ábrán a szupport vektor gépek összehasonlításánál gyakran alkalmazott kettős spirál probléma LS-SVM-mel történő megoldását mutatjuk. Jól látható, hogy ennél az erősen nemlineáris feladatnál, amely számos neuronháló számára kimondottan nehéz feladat probléma, az LS-SVM – hasonlóan az SVM-hez – nagyon jól teljesít.



7.13. ábra. A kettős spirál probléma megoldása LS-SVM-el

Az LS-SVM regresszió

Az LS-SVM – hasonlóan az eddig látott hálózatokhoz – az osztályozós feladaton túl regressziós esetre is alkalmazható. Ekkor az eddigieknek megfelelően egy $f(\mathbf{x})$ függvény közelítése a cél adott $\{\mathbf{x}_i, d_i\}_{i=1}^{P}$ tanítópont-halmaz alapján, ahol $\mathbf{x}_i \in$ $\in \mathsf{R}^N$ egy N-dimenziós bemeneti vektor és $d_i \in \mathsf{R}$ a kívánt kimenet.

A regresszió esetén az optimálási feladat az alábbi:

$$\min_{\mathbf{w},b,\mathbf{e}} J(\mathbf{w},b,\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{P} e_i^2 \tag{7.87}$$

a

$$d_i = \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i \quad (i = 1,2,\ldots,P) \tag{7.88}$$

feltételekkel.

A regresszió válasza a következő alakban írható fel:

$$y(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b\,, \tag{7.89}$$

ahol a $\boldsymbol{\varphi}(.) : \mathsf{R}^N \rightarrow \mathsf{R}^M$ nemlineáris transzformáció most is a jellemzőtérbe való leképezést biztosítja. Az osztályozós esthez hasonlóan itt is felírhatunk egy

Lagrange egyenletet:

$$L(\mathbf{w},b,\mathbf{e};\boldsymbol{\alpha}) = J(\mathbf{w},b,\mathbf{e}) - \sum_{i=1}^{P} \alpha_i \left\{ \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i - d_i \right\} , \tag{7.90}$$

ahol az α_i értékek a Lagrange multiplikátorok. Az optimumra vonatkozó feltételek az alábbiak:

$$\begin{aligned}
&\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \longrightarrow \quad \mathbf{w} = \sum_{i=1}^{P} \alpha_i \boldsymbol{\varphi}(\mathbf{x}_i) \\
&\frac{\partial L}{\partial b} = 0 \quad \longrightarrow \quad \sum_{i=1}^{P} \alpha_i = 0 \\
&\frac{\partial L}{\partial e_i} = 0 \quad \longrightarrow \quad \alpha_i = C e_i \qquad\qquad (i = 1,2,\ldots,P) \\
&\frac{\partial L}{\partial \alpha_i} = 0 \quad \longrightarrow \quad \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i - d_i = 0 \quad (i = 1,2,\ldots,P)
\end{aligned} \tag{7.91}$$

A $\mathbf{w}$ és e kifejezése után a következő lineáris egyenletrendszer írható fel:

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^T \\ \vec{\mathbf{1}} & \boldsymbol{\Omega} + C^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix} , \tag{7.92}$$

ahol az $\boldsymbol{\Omega}$ kivételével az egyenletrendszerben szereplő többi jelölés megegyezik a (7.85)-ben megadottakkal. Az $\boldsymbol{\Omega}$ mátrix (i,j)-edik eleme:

$$\Omega_{ij} = \boldsymbol{\varphi}(\mathbf{x}_i)^T \boldsymbol{\varphi}(\mathbf{x}_j) = K(\mathbf{x}_i,\mathbf{x}_j) \quad (i,j = 1,2,\ldots,P) . \tag{7.93}$$

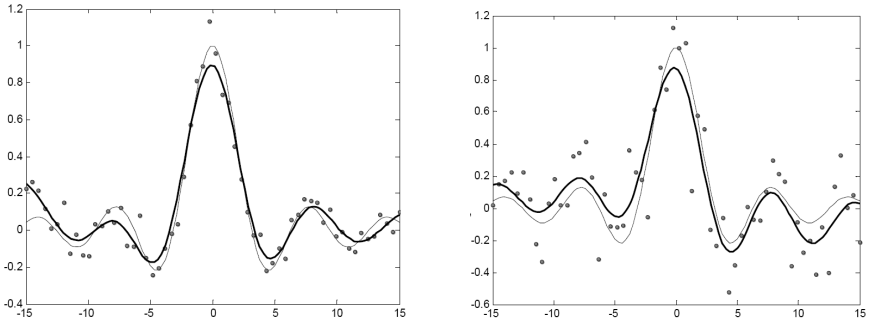
Az eredmény – hasonlóan a hagyományos SVM-hez – a következő alakban írható fel:

$$y(\mathbf{x}) = \sum_{i=1}^{P} \alpha_i K(\mathbf{x},\mathbf{x}_i) + b , \tag{7.94}$$

ahol az α_i-k és b a fenti lineáris egyenletrendszer megoldása.

Az egyenletrendszer felírása a parciális deriváltakból az osztályozós esetnél bemutatottak szerint történik. A regressziós esetben azonban az eredmény kiszámítása ((7.94) összefüggés) és az egyenletrendszer sorai között egyértelműen látszik a kapcsolat, nevezetesen, hogy az egyenletrendszer sorai a regularizációt ($C^{-1}\mathbf{I}$) leszámítva pontosan megfelelnek az eredmény kiszámítására szolgáló összefüggésnek. Ez nem meglepő, hiszen a tanítópontok által reprezentált feltételek gyakorlatilag azt fogalmazzák meg, hogy az adott bemenetre milyen kimenetet várunk a modelltől. Ez a működés jól követhető a 7.10. ábrán. A tanítás során bevezetett regularizáció pedig megadja, hogy mennyire (mekkora hibával) kell illeszkedni a tanítópontokban. Az erősebb regularizáló hatás pontatlanabb illeszkedést követel és simább (egyszerűbb) megoldást, simább válaszfüggvényt eredményez.

A 7.14. ábrán egy egyszerű példát mutatunk be LS-SVM használatával megoldott regressziós feladatra. Látható, hogy még jelentősebb mértékben zajos tanítópontok mellett is jó megoldást kapunk.



(a) $\sigma_n = 0{,}01$ (b) $\sigma_n = 0{,}04$

7.14. ábra. A zajos sinc(x) LS-SVM modellje. (Gauss kernel; $\sigma = \pi$, $C = 10$. A Gauss zaj szórása σ_n.) Az összes tanítópont szupport vektor.

Ritka LS-SVM

A Least-Squares megoldás egyik hátránya, hogy a megoldás nem ritka *(sparse)*, hiszen mint láttuk, minden tanító vektor szupport vektor is egyben. Ha a végeredményt neurális hálózatként értelmezzük, akkor ebben P (azaz a felhasznált tanítópontok számával megegyező számú) nemlineáris neuron található, így az eredmény komplexitása (a háló mérete) nagyobb, mint a hagyományos SVM-mel nyert hálózatoké, melyek valóban szelektálnak a bementek közül. Ez abból is látszik, hogy a megoldásban felhasználjuk az $\alpha_i = Ce_i$ egyenletet, amiből látszik, hogy a i-edik tanítópontban kapott hiba arányos a tanítóponthoz, mint szupport vektorhoz tartozó α_i súllyal. Figyeljük meg, hogy a négyzetes hibafüggvény alkalmazása miatt az ε sáv hiányában gyakorlatilag nincs 0 hibaérték, így nincs $\alpha_i = 0$ érték sem. Ahhoz, hogy a hagyományos SVM ritkasági tulajdonságát visszanyerjük további lépésekre van szükség.

Intuitíven állíthatjuk, hogy a kisebb α_i-k kevésbé járulnak hozzá a megoldáshoz, azaz a kialakított modellhez. A következő „pruning” (metszési) eljárás egy ezen alapuló iteratív módszer, mellyel az LS-SVM alkalmazásánál is elérhető egy egyszerűbb eredmény, azaz kisebb hálózat. A megoldás menete az alábbi:

1. Tanítsuk a hálózatot az összes rendelkezésre álló (P) tanítóponttal.

2. Távolítsuk el egy kisebb részét (pl. 5%-át) a pontoknak úgy, hogy azokat hagyjuk el, melyekhez a legkisebb $|\alpha_i|$-k tartoznak.

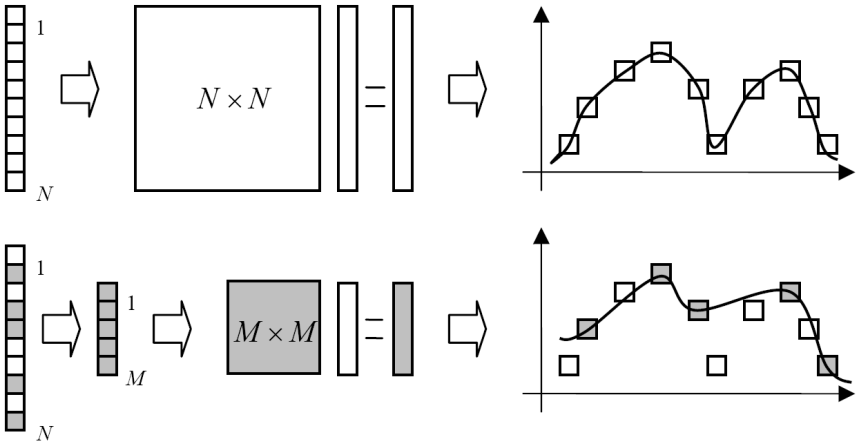
3. Tanítsuk újra az LS-SVM-et a kisebb tanítókészlettel.

4. Folytassuk a 2. lépéstől, amíg a válasz minősége nem romlik. Ha a teljesítmény romlik, akkor a hiperparaméterek (C és Gauss kernel esetén a σ) hangolásával esetleg korrigálhatjuk, illetve még tovább csökkenthetjük a megoldás méretét.

A negyedik pontban említett hiperparaméter hangolás oka jól látható például egy Gauss kerneles hálózat esetén. Ha a metszési *(pruning)* eljárás eredményeképpen csökken a kernelek száma, akkor az egymástól távolabb eső középpontok miatt gyakran növelni kell a Gauss függvények szélességét (σ).

Fontos megjegyezni, hogy a metszési eljárás során, a minták elhagyásával gyakorlatilag a tanítókészlet méretét redukáljuk, ami – főként jelentősebb redukció esetén – információvesztéssel és így a háló minőségének romlásával jár(hat). Ez egyértelmű különbség a Vapnik-féle SVM-hez képest, hiszen ott a ritka megoldást úgy nyertük, hogy közben a Lagrange multiplikátorok meghatározásában az össze tanítópont részt vett. Ennek a kedvezőtlen hatásnak egyfajta mérséklését, vagy kiküszöbölését oldja meg a későbbiekben bemutatásra kerülő LS^2-SVM. A mintapontok redukciójának hatását illusztrálja a 7.15. ábra. Az ábrán az üres négyzetek képviselik az össze tanítópontot, az árnyékolt négyzetek pedig a redukált mintakészlet pontjait.

A 7.16. ábrasorozat az eljárás valós példán történő futtatásának eredményét mutatja. Az (a) ábrán az eredeti LS-SVM megoldása látható. A további ábrák az egyre nagyobb mértékű redukció eredményét mutatják: a (b), (c) és (d) ábrákon egyre több pontot hagytunk el az eredeti tanítókészletből, miközben az approximáció minősége jelentősen nem változott.



7.15. ábra. A metszési eljárás, azaz a tanító minták elhagyásának hatása a feladat méretére valamint az approximáció eredményére

Súlyozott LS-SVM

A négyzetes hibafüggvény a nem normál eloszlású zaj (például kilógó minták, „outlier”-ek) esetén nem optimális, ezért ilyen tanítómintáknál a modellt egy további módosítással hangolhatjuk. A módszer, hasonlóan a „pruning”-hoz az $\alpha_i = Ce_i$ összefüggésen alapul. Ez az összefüggés azt mutatja, hogy az egyes

pontokban kapott hiba, e_i arányos az eredményként kapott α_i súlyokkal. A módosított eljárás célja, hogy az egyes mintapontok szerepét a szerint mérlegelje, hogy ott mekkora a hiba. Ha e_i nagy, akkor az ehhez tartozó mintapont kevésbé megbízható, tehát a teljes eljárásban a szerepét célszerű csökkenteni, míg a pontosabb mintapontok szerepét, ahol e_i kisebb, érdemes növelni. Ezt a hatást egy újonnan bevezetett ν_i súlykészlettel érjük el. Az optimalizálandó egyenlet (a regressziós esetre felírva) ekkor a következőképpen módosul:

$$\min_{\mathbf{w},b,\mathbf{e}} J(\mathbf{w},b,\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{P}\nu_i e_i^2 \tag{7.95}$$

A tanítópontokra vonatkozó feltételek nem változnak:

$$d_i = \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i \quad (i = 1,2,\ldots,P)\,. \tag{7.96}$$

A lineáris egyenletrendszerben ez a következőket jelenti:

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^T \\ \vec{\mathbf{1}} & \boldsymbol{\Omega} + \mathbf{V}_C \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix}\,, \tag{7.97}$$

ahol a $\mathbf{V}_C$ az alábbi diagonálmátrix:

$$\mathbf{V}_C = \operatorname{diag}\left(\begin{bmatrix} \frac{1}{C\nu_1} & \ldots & \frac{1}{C\nu_P} \end{bmatrix}\right) \tag{7.98}$$

A ν_i súlyokat az $e_i = \dfrac{\alpha_i}{C}$ értékek alapján választhatjuk meg, például a következők szerint:

$$\nu_i = \begin{cases} 1 & \text{ha } |e_i/s| \le c_1 \\ \dfrac{c_2 - |e_i/s|}{c_2 - c_1} & \text{ha } c_1 \le |e_i/s| \le c_2 \\ 10^{-4} & \text{egyébként} \end{cases} \tag{7.99}$$

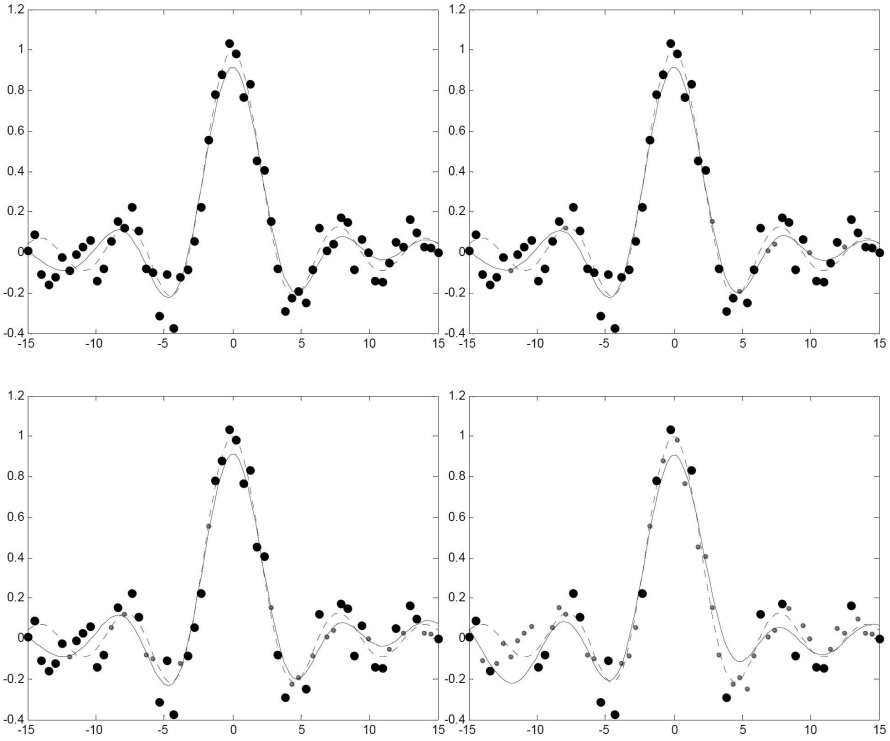
ahol c_1, c_2 és s megválasztása a statisztikában ismert módszerek alapján történhet [Suy02].
Az algoritmus menete a következő:

1. Tanítsuk a hálózatot súlyozás nélkül az összes rendelkezésre álló (P) tanítóponttal és határozzuk meg az e_i értékeket.

2. Határozzuk meg a ν_i súlyokat a fentiek szerint.

3. Számítsunk ki egy súlyozott LS-SVM modellt a ν_i súlyok segítségével.

Ez az eljárás iteratívan ismételhető, de a gyakorlatban egy súlyozó lépés általában elegendő.



7.16. ábra. A metszési eljárás folyamatának négy állapota egy zajos sinc(x) megoldása során. (RBF kernel; $\sigma = \pi$, $C = 10$. A zaj szórása $\sigma_n = 0{,}01$.) A mintapontok közül 22 darabot tartunk meg.

7.4.2. Az LS-SVM hatékonyabb megoldása

Az LS-SVM tanításának komplexitása jóval kisebb, mint a hagyományos SVM tanításé, hiszen a QP helyett itt „csak” egy lineáris egyenletrendszert kell megoldani. Az LS-SVM használatának azonban vannak hátrányai is, mint pl. a ritkaság elvesztése. Mindemellett még az LS-SVM esetén is felmerül a hatékonyabb megoldás iránti igény, hisz a gyakorlati felhasználásokban még ez a módszer is túl számítás- és memóriaigényesnek bizonyulhat. Itt a fő probléma a kernel mátrix méretének csökkentése, ami így természetesen gyakran összekapcsolódik az LS-SVM egyéb hátrányinak a kiküszöbölésével.

A számítás gyorsítása

A tanítási sebesség növelésére, valamint a memóriaigények csökkentésére az LS-SVM lineáris egyenletrendszerét kell gyorsabban, kisebb memóriában megoldani. Erre Suykens a Heistens-Stiefel konjugált gradiens módszert javasolja [Suy02]. Ez a lineáris egyenletrendszer egy iteratív megoldása, ami ez eredeti LS-SVM

felírásnak megfelelő egyenletrendszer eredményére vezet.

Más lehetőség a ritka megoldás és a hatékonyság növelésének együttes alkalmazása, ami az LS-SVM módosítását jelenti. Ennek eléréséhez például a következőkben bemutatásra kerülő LS^2-SVM-et vagy az LS-SVM-mel lényegében ekvivalens ridge regresszió egy redukált változatát, a csökkentett rangú kernel ridge regressziót alkalmazhatjuk.

7.4.3. LS^2-SVM

Az LS^2-SVM fő célja egy ritka LS-SVM megoldás elérése, csökkentve a modell komplexitását, azaz a rejtett rétegbeli nemlineáris neuronok számát. A módszer két fontos lépést tartalmaz:

- Az *első lépés* átalakítja az LS-SVM megoldást úgy, hogy az a tanítópontoknak csak egy részhalmazát használja „szupport vektornak". Ennek következtében egy túlhatározott egyenletrendszert kapunk, melynek LS megoldása adja az eredményt.
- A *második lépés* a „szupport vektorok" (a kernel függvényeket meghatározó vektorok) automatikus meghatározására ad eljárást.

Túlhatározott egyenletrendszer

Az új megközelítés kiindulópontja az LS-SVM lineáris egyenletrendszere. Ha a tanító készlet P mintapontot tartalmaz, akkor az egyenletrendszer $(P+1)$ ismeretlent, az α-kat és a b-t, $(P+1)$ egyenletet és $(P+1)^2$ együtthatót tartalmaz. Az együtthatók a $K(\mathbf{x}_i, \mathbf{x}_j)$ $(i, j = 1, \ldots, P)$ kernel függvény értékek. A mintapontok száma tehát meghatározza az egyenletrendszer méretét, ami egyúttal a megoldás komplexitását, a hálózat méretét is megszabja. Hogy ritka megoldást, azaz egy kisebb modellt kapjunk, az egyenletrendszert, illetve az együtthatómátrix méretét redukálni kell.

Vizsgáljuk meg közelebbről az LS-SVM regressziós problémát leíró egyenletrendszert és jelentését. Az első sor jelentése:

$$\sum_{k=1}^{N} \alpha_k = 0 \tag{7.100}$$

míg a j-edik sor a

$$b + \alpha_1 K(\mathbf{x}_j, \mathbf{x}_1) + \cdots + \alpha_k \left[K(\mathbf{x}_j, \mathbf{x}_k) + C^{-1}\mathbf{I}\right] + \cdots + \alpha_P K(\mathbf{x}_j, \mathbf{x}_P) = d_j \tag{7.101}$$

feltételt, megkötést tartalmazza.

Az egyenletrendszer redukálásánál sorokat, illetve oszlopokat hagyhatunk el.

- Ha a j-edik oszlopot hagyjuk el, akkor az ennek megfelelő $K(\mathbf{x}, \mathbf{x}_j)$ kernel szintén „eltűnik", így az eredményként megkapott hálózat mérete csökken. Az első sor által támasztott feltétel azonban automatikusan alkalmazkodik, hisz a megmaradó α-k összege 0 marad.

– Ha a j-edik sort töröljük, akkor az $(\mathbf{x}_j, d_j)$ tanító pontnak megfelelő információ elvész, hiszen a j-edik megkötést elveszítjük.

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

Minden oszlop egy 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. Az $(\mathbf{x}_j, d_j)$ 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 elemeket szürkével jelöltük.

$$\left[\begin{array}{c|cccc} 0 & \vec{\mathbf{1}} & & & \\ \hline \vec{\mathbf{1}}^T & \Omega_{11} + \frac{1}{C} & \Omega_{12} & \dots & \Omega_{1P} \\ & \Omega_{21} & \Omega_{22} + \frac{1}{C} & \dots & \Omega_{2P} \\ & \vdots & \vdots & \ddots & \vdots \\ & \Omega_{(P-1)1} & \Omega_{(P-1)2} & \dots & \Omega_{(P-1)P} \\ & \Omega_{P1} & \Omega_{P2} & \dots & \Omega_{PP} + \frac{1}{C} \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_P \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline d_1 \\ d_2 \\ \vdots \\ d_P \end{array}\right] \tag{7.102}$$

A hagyományos metszési (pruning) technika esetében pontosan ez történik, hisz a metszési algoritmus iteratívan kihagyja a tanítópontok egy részét. Az elhagyott tanítópontok által hordozott információ ezért teljesen elvész. A tanítópontok által hordozott információ megőrzése a részleges redukció mellett lehetséges.

Részleges redukció. A $(\mathbf{x}_j, d_j)$ tanító mintát csak részben hagyjuk el, úgy, hogy töröljük a ponthoz tartozó oszlopot, de megtartjuk a hozzá tartozó sort. Így a tanítóminta 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 ($M < P$) *„szupport vektort”* (oszlopot), de megtartjuk az összes megkötést (sort), az egyenletrendszer túlhatározottá válik. A részleges redukció hatását a mutatja a következő összefüggés, ahol a

szürkével jelölt részeket távolítjuk el.

$$\left[\begin{array}{c|ccccc} 0 & \vec{\mathbf{1}} & & & & \\ \hline \vec{\mathbf{1}}^T & \Omega_{11}+\frac{1}{C} & \Omega_{12} & \dots & \Omega_{1P} \\ & \Omega_{21} & \Omega_{22}+\frac{1}{C} & \dots & \Omega_{2P} \\ & \vdots & \vdots & \ddots & \vdots \\ & \Omega_{(P-1)1} & \Omega_{(P-1)2} & \dots & \Omega_{(P-1)P} \\ & \Omega_{P1} & \Omega_{P2} & \dots & \Omega_{PP}+\frac{1}{C} \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_P \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline d_1 \\ d_2 \\ \vdots \\ d_P \end{array}\right] \tag{7.103}$$

A fentiek alapján kapott túlhatározott ($P \times M$ méretű) egyenletrendszert legkisebb négyzetes hiba értelemben oldhatjuk meg. Egyszerűsítsük a jelöléseinket úgy, hogy a fenti egyenletrendszer mátrix alakja legyen:

$$\mathbf{A}\mathbf{u} = \mathbf{v}\,, \tag{7.104}$$

melynek legkisebb négyzetes hibájú megoldása:

$$\mathbf{u} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{v}\,. \tag{7.105}$$

Az oszlopok törlése a sorok megtartása mellet biztosítja, hogy a kernelek (neuronok) száma csökkenjen, miközben az eljárás az összes ismert megkötést (tanítómintát) figyelembe veszi. Ez a kulcsa annak, hogy a modell komplexitása a pontosság megtartása mellett is csökkenthető legyen.

Látható, hogy a kernel mátrix redukciója során az oszlopok elhagyásával a regularizálás „aszimmetrikussá" válik, hiszen a továbbiakban nem minden sorban szerepel regularizáció. Ennek kiküszöbölésére megtehető, hogy a regularizációt már csak a kernel térbeli megoldás során használjuk, ami a kernel térbeli margó maximalizálásnak ($||\alpha||$ minimalizálásának) felel meg. Ez hasonló az SV megoldások nemlineáris kiterjesztéséhez, ahol a $\mathbf{w}$ súlyvektor minimalizálása valójában a jellemző térben történik. A kernel térben regularizált megoldás:

$$(\mathbf{A}^T\mathbf{A} + C^{-1}\mathbf{I})\mathbf{u} = \mathbf{A}^T\mathbf{v}\,, \tag{7.106}$$

ahol

$$\mathbf{A} = \left[\begin{array}{c|c} 0 & \vec{\mathbf{1}} \\ \hline \vec{\mathbf{1}}^T & \boldsymbol{\Omega} \end{array}\right], \boldsymbol{\Omega} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & \dots & K(\mathbf{x}_1,\mathbf{x}_M) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_P,\mathbf{x}_1) & \dots & K(\mathbf{x}_P,\mathbf{x}_M) \end{bmatrix}. \tag{7.107}$$

A módosított, részlegesen redukált egyenletrendszert legkisebb négyzetes hibájú (least-squares) értelemben oldjuk meg, ezért nevezzük ezt a módszert Least Squares LS-SVM-nek vagy röviden LS^2-SVM-nek [Val04].

Az itt bemutatott részleges redukció hasonlít a hagyományos SVM kiterjesztéseként bevezetett redukált Szupport Vektor Gép (Reduced Support Vector Machine – RSVM) alapelvéhez [Lee01b]. Az RSVM esetén azonban, mivel az SVM eleve kisebb (ritka) modellt eredményez, a részleges redukció célja az algoritmikus komplexitás csökkentése, míg az LS-SVM esetében elsődlegesen a modell komplexitásának, a hálózat méretének csökkentése, azaz a ritka LS-SVM elérése a cél.

A kiválasztási eljárás

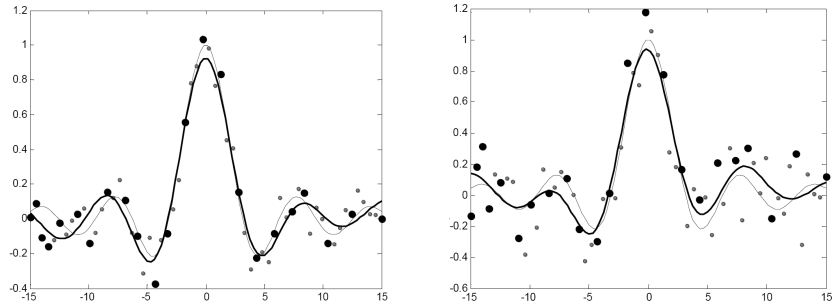
A fenti részleges redukció alkalmazása esetén szükség van valamilyen módszerre, ami meghatározza a szükséges szupport vektorokat. A kernel mátrix oszlopaiból kiválasztható egy olyan lineárisan független részhalmaz, melynek elemeiből lineáris kombinációival a többi előállítható. Ez a kernel mátrix „bázisának" meghatározásával érhető el, hiszen az oszlopvektorok terének bázisa az a legkisebb vektorkészlet, amellyel a feladat megoldható. Ha a bázis meghatározásánál nem követeljük meg, hogy az oszlopvektorok által kifeszített tér és a kiválasztott bázisok által kifeszített tér azonos legyen, hanem bizonyos toleranciát is megengedünk, akkor a tolerancia mértékének meghatározásával egy szabad paramétert kaptunk. Ez a szabad paraméter lehetővé teszi, hogy kontroláljuk a bázisvektorok számát (M), hiszen a célunk az, hogy a P darab P-dimenziós oszlopvektorból $M < P$ bázisvektort határozzunk meg, ahol M minél kisebb.

A bázisvektorok száma, ami egyben a szupport vektorok száma is valójában nem függ a mintapontok számától (P), 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 RRE alakra *(reduced row echelon form, RREF)* hozása során részleges pivotolással végrehajtott Gauss-Jordan eliminációval történik [Pre02, Gol89] (ld. Függelék). A tolerancia figyelembevétele a pivot elem (p) ellenőrzésével lehetséges. 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 $p \leq \varepsilon'$ (ahol ε' a tolerancia paraméter), akkor az adott oszlop elhagyható, ellenkező esetben az oszlopnak megfelelő bemenet egy szupport vektor. A módszer azon oszlopvektorok listáját adja meg, melyek az ε' toleranciaérték értelmében lineárisan függetlenek.

A megfelelő tolerancia helyes meghatározása hasonló a többi hiperparaméter (C, ε, valamint a kernel paraméterek) megválasztásához. Egy lehetséges megoldás a kereszt kiértékelés (cross-validation) alkalmazása. Minél nagyobb a tolerancia, annál kisebb a hálózat, és annál nagyobb az approximáció hibája. Az ε' helyes megválasztása egy olyan kompromisszum alapján lehetséges, ahol

a pontos approximáció, illetve a modell komplexitása áll szemben egymással.



(a) $\sigma_n = 0{,}01$ (b) $\sigma_n = 0{,}04$

7.17. ábra. A zajos sinc(x) LS^2-SVM modellje. (RBF kernel; $\sigma = \pi$, $C = 100$, $\varepsilon' = 0{,}01$. A zaj szórása: σ_n.) Az RREF módszer eredményeképp a megoldás 23 szupport vektoron alapul.

Fontos hangsúlyozni, hogy nulla tolerancia esetén az LS^2-SVM és az LS-SVM megoldás azonos, mivel ebben az esetben a kiválasztási módszer a mátrix minden oszlopát, azaz az összes tanítómintát megtartja (kivéve, ha a tanítópontok között volt redundancia).

Az alábbiakban az LS^2-SVM alkalmazására mutatunk két mintapéldát. A 7.18. ábrán egy kétdimenziós sinc függvény LS^2-SVM-mel történő approximációja látható. A bal oldali ábra a kiinduló tanítópontokat mutatja, a jobboldali pedig a megoldást, ahol külön megjelennek az eljárás által meghagyott tanítópontok (szupport vektorok) is. Láthatóan a tanítópontok elenyésző töredéke elég a megfelelően pontos approximáció eléréséhez. A 7.19. ábra a klasszikus kettős spirál probléma megoldását adja meg. A módszer hatékonyságnövelő hatása itt is jól követhető.

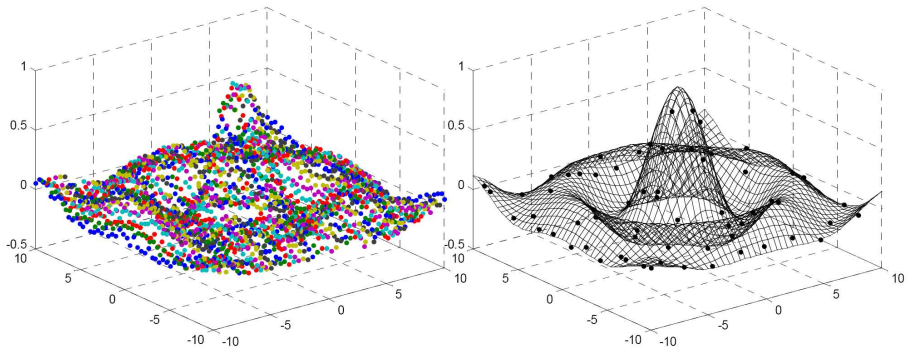
Súlyozott LS^2-SVM

Az LS-SVM-nél megmutattuk hogy egy iteratív eljárással súlyozott megoldás is adható, ami zajos esetben, főként kilógó minták (outlierek) esetén jelentősen jobb megoldásra vezet.

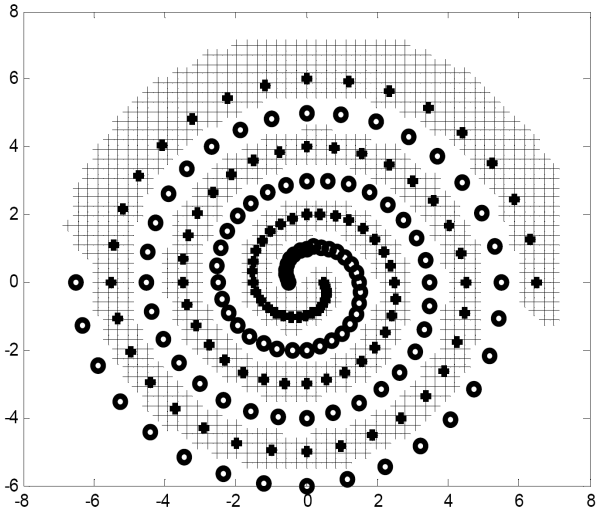
Amennyiben ismert vagy becsülhető a tanító mintákat terhelő zaj mértéke, ezt az információt az LS^2-SVM esetén is fel lehet használni. A mintapontok azonos kezelése helyett ebben az esetben a pontosabb mintákat nagyobb, a pontatlan, azaz zajos mintákat kisebb súllyal célszerű figyelembe venni. A hiba súlyozott számítása:

$$\varepsilon_\nu = \sum_{i=1}^{P} \nu_i e_i^2 = \sum_{i=1}^{P} \nu_i (d_i - y_i)^2 \,. \tag{7.108}$$

A túlhatározott LS^2-SVM egyenletrendszer súlyozott megoldása az előbbi-



7.18. ábra. Egy kétdimenziós sinc függvény approximációja LS^2-SVM-el. A bal oldali képen az összes (2500) tanítópont, a jobb oldalin az LS^2-SVM eljárással kapott 63 szupport vektor (fekete pontok) és az approximáció eredménye látható. (Gauss kernel; $\sigma = \pi$, $C = 1000$, $\varepsilon' = 0{,}15$.)



7.19. ábra. A kettős spirál probléma megoldása LS^2-SVM alkalmazásával. A mintapontok száma 194, a szupport vektorok száma 119. (Gauss kernel; $\sigma = 0{,}5$, $C = 10$, $\varepsilon' = 0{,}9$.)

ekben bevezettet átírást alkalmazva:

$$\mathbf{A}^T\mathbf{V}\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{V}\mathbf{v}\,, \tag{7.109}$$

ahol a $\mathbf{V}$ súlyozó mátrix egy diagonál mátrix, amely a főátlóban a ν_i súlyokat tartalmazza. A súlyokat úgy kell beállítani, hogy jól tükrözzék az egyes tanítóminták hatását a lineáris illesztésre. A zaj mértéke, illetve a súlyozás meghatározható egy előzetes megoldás alapján (vagy *a priori* információ alapján), de

a túlhatározott egyenletrendszer megoldására használhatunk olyan statisztikai megoldó módszereket is, amelyek csökkentik a nagyon hibás adatok hatását. Ilyen például a Least Absolute Residuals – LAR, a Bisquare weights vagy a Least Trimmed Squares – (LTS) módszer [Ciz04, Hol77, Hub81].

7.4.4. Ridge regresszió

A fejezet elején egy egyszerű kernel gép származtatásánál láttuk, hogy a négyzetes hiba minimalizálását célzó lineáris gép értelmezhető kernel gépként is. A megoldást a tanítópontok bemeneti vektoraiból képezett mátrix pszeudo-inverze segítségével tudtuk meghatározni (ld. (7.8) összefüggés). A könnyebb követhetőség miatt ezt az összefüggést itt megismételjük:

$$\hat{\mathbf{w}}^* = \hat{\mathbf{X}}^T \left(\hat{\mathbf{X}} \hat{\mathbf{X}}^T \right)^{-1} \mathbf{d} \,. \tag{7.110}$$

Azt is megmutattuk, hogy az így származtatott súlyvektorral kapott háló kernel gépként is értelmezhető, ahol a választ az alábbi formában is felírhatjuk:

$$y(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \hat{\mathbf{X}}^T \boldsymbol{\alpha} = \sum_{i=1}^{P} \alpha_i \left(\hat{\mathbf{x}}^T \hat{\mathbf{x}}_i \right) = \sum_{i=1}^{P} \alpha_i K_i(\hat{\mathbf{x}}) \,. \tag{7.111}$$

Amennyiben az $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ mátrix ragja nem teljes, vagy bármilyen pl. numerikus instabilitás miatt nem invertálható, a következőképp módosított összefüggés alapján nyerhető a megoldás:

$$\hat{\mathbf{w}}^* = \hat{\mathbf{X}}^T \left(\hat{\mathbf{X}} \hat{\mathbf{X}}^T + \lambda \mathbf{I} \right)^{-1} \mathbf{d} \,, \tag{7.112}$$

ahol $\mathbf{I}$ az $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ mátrixnak megfelelő méretű egységmátrix, λ pedig egy kis pozitív konstans. Ez az összefüggés valójában a rosszul kondicionált mátrixok invertálásánál alkalmazott regularizációs eljárás, ami a matematikában Tyihonov regularizáció néven is ismert. Erre az összefüggésre azonban több úton is eljuthatunk. A fenti megoldás a Tyihonov regularizációtól függetlenül az ún. ridge regresszió *(ridge regression)* [Hoe70] útján is származtatható.

A következőkben röviden bemutatjuk a ridge regresszió származtatását és azt, hogy a ridge regresszió és az LS-SVM probléma-megfogalmazása, és így a megoldás is lényegében azonos. Mivel ez a tárgyalás főként a neurális modellekhez, illetve a szupport vektor gépekhez kapcsolódik, a ridge regressziós eljárásra csak röviden térünk ki. A ridge regressziót többnyire eltolásérték *(bias)* nélkül alkalmazzák, a következőkben itt is ezt a változatot mutatjuk be.

Lineáris ridge regresszió

A korábban bemutatott osztályozási problémának megfelelően itt is az $\{\mathbf{x}_i, d_i\}_{i=1}^{P}$ tanító mintakészlet alapján keressük az $y(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ lineáris modellt. A ridge regresszió is a négyzetes hibát minimalizálja, de az LS-SVM-hez

hasonlóan közben a súlyvektor hosszának minimumát is biztosítani szeretné. Ennek megfelelően a $\mathbf{w} \in \mathsf{R}^N$ súlyvektort ridge regresszióval a következőképpen határozzuk meg. Jelölje $J(\mathbf{w})$ most is a minimalizálandó kritériumfüggvényt (az 1/2-es szorzó a deriválás utáni eredmény egyszerűsítése miatt szerepel):

$$J(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \frac{1}{2}C\sum_{i=1}^{P}(d_i - \mathbf{x}_i^T\mathbf{w})^2 \tag{7.113}$$

A $\dfrac{\partial J}{\partial \mathbf{w}}$ parciális deriválás eredményéből az optimális $\mathbf{w}$ az alábbiak szerint számítható:

$$\sum_{i=1}^{P}(d_i - \mathbf{x}_i^T\mathbf{w})\mathbf{x}_i = \frac{1}{C}\mathbf{w} \tag{7.114}$$

$$\mathbf{w} = \left(\sum_{i=1}^{P}\mathbf{x}_i\mathbf{x}_i^T + \frac{1}{C}\mathbf{I}\right)^{-1}\left(\sum_{j=1}^{P}d_j\mathbf{x}_j\right) \tag{7.115}$$

Mátrix alakban felírva (ahol $\mathbf{d} = \begin{bmatrix} d_1 & \dots & d_P \end{bmatrix}^T$ a tanítópontok kívánt válaszaiból képezett vektor és $\mathbf{X}$ az $\mathbf{x}_i$ $(i = 1, \dots, P)$ tanítóminta-bemenetekből, mint sorvektorokból képezett mátrix):

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_P^T \end{bmatrix}, \tag{7.116}$$

a megoldás súlyvektor:

$$\mathbf{w} = \left(\mathbf{X}^T\mathbf{X} + \frac{1}{C}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{d} = \mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T + \frac{1}{C}\mathbf{I}\right)^{-1}\mathbf{d}\,. \tag{7.117}$$

Az

$$\boldsymbol{\alpha} = \left(\mathbf{X}\mathbf{X}^T + \frac{1}{C}\mathbf{I}\right)^{-1}\mathbf{d} \tag{7.118}$$

jelölés bevezetésével

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{\alpha} \tag{7.119}$$

a végeredmény:

$$y(\mathbf{x}) = \mathbf{x}^T\mathbf{X}^T\boldsymbol{\alpha} = \sum_{i=1}^{P}\alpha_i\mathbf{x}^T\mathbf{x}_i\,. \tag{7.120}$$

Vegyük észre, hogy ez a felírás egy kis különbséggel megegyezik a 1.1 részben bevezetett egyszerű kernel gép összefüggéseivel. A kis különbség abból adódik, hogy itt a kiinduló kritériumfüggvény a súlyvektor hosszának négyzetét, mint minimalizálandó mennyiséget, is tartalmazza. Ennek következménye a $\mathbf{w}$ illetve az $\boldsymbol{\alpha}$ vektorok összefüggéseiben ((7.117), illetve (7.118) összefüggések) a $C^{-1}\mathbf{I}$

regularizációs tag megjelenése. A ridge regresszió tehát az egyszerű kernel gép regularizált változata. Azt is észrevehetjük, hogy ez az eredmény megegyezik az eltolás-érték nélküli lineáris LS-SVM-re kapott összefüggéssel.

Természetesen a ridge regresszió is kiterjeszthető nemlineáris esetre a szokásos $\boldsymbol{\varphi}(\mathbf{x})$ transzformáció, és a kernel trükk alkalmazásával.

Nemlineáris kernel ridge regresszió

A nemlineáris ridge regresszió esetén – csakúgy, mint az SVM vagy az LS-SVM kiterjesztésénél – a lineáris regressziós modellt egy magasabb dimenziós térben, a jellemző térben alkalmazzuk, ami a szokásos módon egy nemlineáris transzformáció eredménye.

A $\boldsymbol{\varphi}(.)$ leképezés segítségével a lineáris esetben megismert célfüggvény a következőképpen módosul:

$$J(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \frac{1}{2}C\sum_{i=1}^{P} e_i^2 \tag{7.121}$$

ahol

$$e_i = d_i - \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) \quad (i = 1, \ldots, P)\,. \tag{7.122}$$

Az LS-SVM regressziónál bemutatott lépéseknek megfelelően az $L(\mathbf{w}, \mathbf{e}, \boldsymbol{\alpha})$ Lagrange-egyenlet írható fel az α_i $(i = 1, \ldots, P)$ Lagrange együtthatókkal. A deriválások után a kernel trükk alkalmazásával az alábbi megoldás nyerhető (mátrix alakban):

$$\boldsymbol{\alpha} = \left(\boldsymbol{\Omega} + \frac{1}{C}\mathbf{I}\right)^{-1}\mathbf{d}\,, \tag{7.123}$$

ahol az $\Omega_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ a tanító mintákból képzett kernel mátrix. Látható, hogy a nemlineáris eset itt is a kernel térben kerül megoldásra. A megoldás ugyanakkor megfelel egy regularizált kernel térbeli megoldásnak. Ez annyit tesz, hogy a $\left\|\mathbf{w}\right\|^2$ jellemző térbeli minimalizálása az $\left\|\boldsymbol{\alpha}\right\|^2$ kernel térbeli minimalizálására vezet.

A kernel ridge regressziós modell:

$$y(\mathbf{x}) = \sum_{i=1}^{P} \alpha_i K(\mathbf{x}, \mathbf{x}_i)\,, \tag{7.124}$$

ami a regularizációtól eltekintve ismét megegyezik az egyszerű nemlineáris kernel gép válaszával (ld. (7.14) összefüggés).

Csökkentett rangú kernel ridge regresszió

Az eredeti ridge regressziós megoldásokban a kernel mátrix négyzetes, ami azt jelenti, hogy a modell a tanító pontoknak megfelelő számú szupport vektort tartalmaz, azaz nem ritka. A csökkentett rangú kernel ridge regresszió *(Reduced Rank Kernel Ridge Regression – RRKRR)* [Caw02] eljárás a ritka megoldást a

kernel mátrix redukciójával éri el. Az RRKRR eljárás tehát hasonló célt tűz ki, mint az LS^2-SVM, a célhoz azonban más úton jut el.

Az eljárást eltolásérték (bias) alkalmazása mellett mutatjuk be, hogy az eredmény összevethető legyen az LS-SVM eredményével.

A ritka megoldáshoz az approximációban felhasznált kernel függvények – azaz a kernel középpontoknak megfelelő szupport vektorok – számát kell csökkenteni. Az eredeti ridge regresszió (és természetesen az LS-SVM is) olyan kernel mátrixot használ fel, ahol mind az oszlopok, mind a sorok számát a tanítópontok száma határozza meg, hiszen minden tanítópont egy kernel középpont – ezek felelnek meg az oszlopoknak – és ugyanakkor minden tanítópont a jellemzőtérben egy lineáris egyenletet is jelent – ezek felelnek meg a soroknak. Az RRKRR eljárás az oszlopok számát redukálja úgy, hogy a tanítópontokból kiválaszt egy részhalmazt – ezek fogják képezni a kernel középpontokat –, és megtartja az összes tanítópontot, tehát megtartja a sorok számát. Az RRKRR eljárás tehát az LS^2-SVM-hez hasonlóan részleges redukciót alkalmaz.

Az oszlopok kiválogatásához az alábbiakból indulhatunk ki. Az LS-SVM (7.83) és (7.91) összefüggései a $\mathbf{w}$ súlyvektort, mint a jellemzőtérbeli $\boldsymbol{\varphi}(\mathbf{x}_i)$ vektorok súlyozott összegét fejezik ki:

$$\mathbf{w} = \sum_{i=1}^{P} \alpha_i \boldsymbol{\varphi}(\mathbf{x}_i) \,. \tag{7.125}$$

Ez azt jelenti, hogy a súlyvektorok terét a $\boldsymbol{\varphi}(\mathbf{x}_i)$ vektorok meghatározzák; mint bázisvektorok kifeszítik a teret. Kérdés azonban, hogy az összes $\boldsymbol{\varphi}(\mathbf{x}_i)$ $(i = 1, \ldots, P)$ jellemzőtérbeli vektorra szükség van-e a súlyvektor (adott pontosságú) reprezentációjához. Tételezzük fel, hogy ehhez a tanítópontok egy S részhalmaza elegendő. Ekkor a $\mathbf{w}$ súlyvektor felírható az S részhalmazba tartozó jellemzővektorok súlyozott összegeként:

$$\mathbf{w} = \sum_{i \in S} \beta_i \boldsymbol{\varphi}(\mathbf{x}_i) \,. \tag{7.126}$$

Ha az S részhalmazba tartozó jellemzővektorok bázis alkotnak a jellemzőtérben, akkor bármely $\mathbf{x}$ bemenet jellemzőtérbeli képe is kifejezhető, mint az S-beli $\boldsymbol{\varphi}(\mathbf{x}_i)$ vektorok lineáris kombinációja:

$$\boldsymbol{\varphi}_S(\mathbf{x}) = \sum_{i \in S} \zeta_i \boldsymbol{\varphi}(\mathbf{x}_i) \,. \tag{7.127}$$

Ha a bázisvektorok $\boldsymbol{\varphi}(\mathbf{x})$-nek csak közelítő reprezentációját, $\hat{\boldsymbol{\varphi}}(\mathbf{x})$-t adják, akkor olyan bázisfüggvényeket kell kiválasztanunk, melyek mellett ez a közelítés a legkisebb hibájú. A hibát most mint a két vektor normalizált euklideszi távolságát definiálhatjuk:

$$\delta_j = \frac{\left\| \boldsymbol{\varphi}(\mathbf{x}_j) - \hat{\boldsymbol{\varphi}}_S(\mathbf{x}_j) \right\|^2}{\left\| \boldsymbol{\varphi}(\mathbf{x}_j) \right\|^2} \,. \tag{7.128}$$

Megmutatható [Bau01], hogy ez a hiba a kernel trükk felhasználásával a kernel mátrix segítségével is felírható:

$$\delta_j = 1 - \frac{\mathbf{K}_{Sj}^T \mathbf{K}_{SS}^{-1} \mathbf{K}_{Sj}}{k_{jj}}\,, \tag{7.129}$$

ahol $\mathbf{K}_{SS}$ a $\mathbf{K}$ kernel mátrix almátrixa: $\mathbf{K}_{SS} = \{K_{ij}\}_{i,j\in S}$, $\mathbf{K}_{Sj} = \{K_{ji}\}_{j\in S}^T$ pedig egy belső szorzatokból képezett oszlopvektor.

A jellemzőtér bázisának meghatározásához az így definiált hiba átlagát kell minimalizálni a tanítókészlet összes pontját tekintetbe véve. Vagyis maximalizálni kell a

$$J(S) = \frac{1}{P} \sum_{j=1}^{P} \frac{\mathbf{K}_{Sj}^T \mathbf{K}_{SS}^{-1} \mathbf{K}_{Sj}}{k_{jj}} \tag{7.130}$$

mennyiséget. Az eljárást az üres halmazzal, vagyis $S = \emptyset$-val indítjuk, és mohó stratégiát alkalmazunk. Minden egyes lépésben azt a tanítóvektort adjuk S-hez, mely a (7.130) kifejezést maximálja. Az eljárás magától leáll, ha $\mathbf{K}_{SS}$ már nem invertálható.

Mivel a súlyvektor (7.126) szerint kifejezhető S-be tartozó bázisvektorok lineáris kombinációjaként, a következő duális egyenletet kapjuk, melyet $\boldsymbol{\beta}$ és b szerint kell minimalizálnunk.

$$J(\boldsymbol{\beta}, b) = \frac{1}{2} \sum_{i,j\in S} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2} C \sum_{i=1}^{P} \left(d_i - \sum_{j\in S} \beta_j K(\mathbf{x}_i, \mathbf{x}_j) - b \right)^2 . \tag{7.131}$$

A $\boldsymbol{\beta}$ és b szerinti parciális deriváltak alapján megkeresve a minimumhelyeket, valamint ezt C-vel leosztva az alábbi egyenletet kapjuk:

$$\sum_{i\in S} \beta_i \sum_{j=1}^{P} K(\mathbf{x}_i, \mathbf{x}_j) + b = \sum_{j=1}^{P} d_j \tag{7.132}$$

és $\forall r \in S$-re

$$\begin{aligned} &\sum_{i\in S} \beta_i \left(\frac{1}{C} K(\mathbf{x}_i, \mathbf{x}_r) + \sum_{j=1}^{P} K(\mathbf{x}_j, \mathbf{x}_r) K(\mathbf{x}_j, \mathbf{x}_i) + b \sum_{i=1}^{P} K(\mathbf{x}_i, \mathbf{x}_r) \right) \\ &= \sum_{j=1}^{P} d_j K(\mathbf{x}_i, \mathbf{x}_r)\,. \end{aligned} \tag{7.133}$$

A fenti egyenletek az alábbi $(|S|+1)\times(|S|+1)$ méretű lineáris egyenletrendszerre vezetnek ($|S|$ az S halmaz elemeinek a száma):

$$\begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\Phi} \\ \boldsymbol{\Phi} & P \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \sum_{i=1}^{P} y_i \end{bmatrix}\,, \tag{7.134}$$

ahol

$$\Omega_{ri} = \frac{1}{C}K(\mathbf{x}_i, \mathbf{x}_r) + \sum_{j=1}^{P} K(\mathbf{x}_j, \mathbf{x}_r)K(\mathbf{x}_j, \mathbf{x}_i) \quad i, r \in S$$

$$\Phi_r = \sum_{i=1}^{P} K(\mathbf{x}_i, \mathbf{x}_r) \quad \forall i \in S$$

$$c_r = \sum_{j=1}^{P} y_j K(\mathbf{x}_i, \mathbf{x}_r) \quad \forall i \in S$$

Az egyenletrendszer megoldása egy csökkentett rangú (bázisú) ritka megoldásra vezet, mivel az csak $|S|$ számú kernelt és az eltolást tartalmazza.

7.5. Kernel CMAC: egy LS-SVM gép véges tartójú kernel függvényekkel

Az eddigiekben láttuk, hogy a bázisfüggvényes hálózatok értelmezhetők kernel gépként is, amennyiben a jellemzőtérből a belső szorzat felhasználásával áttérünk a kernel térre. A kernel interpretáció származhat újfajta megközelítésből (szupport vektor gépek), melynek nincs a klasszikus neuronhálók között ekvivalense, de olyan úton is származtatható, hogy az eredmény valójában a klasszikus bázisfüggvényes megoldással ekvivalens (LS-SVM, ridge regresszió és változataik). Az előbbi esetben a komplexitásra vonatkozó előnyökön túl további előnyök is jellemzik a kernel gépeket: a szupport vektor gépeknél az általánosítóképességre is megfogalmazhatók állítások. Az utóbbi esetben ilyen következménye nincs a kernel megközelítésnek. A komplexitás jelentős redukciója, a limitált komplexitás biztosítása azonban önmagában is jelentős előny. Az LS-SVM vagy a ridge regressziós megközelítést tehát érdemes akkor is alkalmazni, ha ezzel csupán a kapott háló komplexitását tudjuk redukálni. Ennek különösen akkor van jelentősége, ha az eredeti változatot a nagy komplexitás miatt esetleg meg sem tudjuk valósítani, miközben a kernel változat könnyen implementálható. Ilyen esetre ad példát az előző fejezetben bemutatott CMAC háló, amely számos előnyös tulajdonsága ellenére, többdimenziós esetben komplexitás-problémák miatt csak korlátozottan alkalmazható.

A következőkben azt mutatjuk be, hogy ez a komplexitás-probléma a kernel reprezentáció segítségével hogyan oldható meg. A kernel-CMAC származtatásához induljunk ki a CMAC válaszát megadó (6.48) összefüggésből:

$$y(\mathbf{x}) = \mathbf{a}(\mathbf{x})^T \mathbf{w}\,. \tag{7.135}$$

A választ megkapjuk, ha az optimális súlyvektor (6.55) kifejezését ebbe behelyettesítjük. Ezzel egy adott $\mathbf{x}$-re a CMAC válasza:

$$y(\mathbf{x}) = \mathbf{a}(\mathbf{x})^T \mathbf{w}^* = \mathbf{a}(\mathbf{x})^T \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{d}\,. \tag{7.136}$$

Vegyük észre, hogy ez az összefüggés formailag azonos az egyszerű kernel gép válaszával ((7.9) összefüggés), ha az ottani $\hat{\mathbf{x}}$ bemenet helyére a CMAC kimeneti rétegének bemenetét (az asszociációs vektort) helyettesítjük, és hasonlóan az $\hat{\mathbf{X}}$ mátrix helyére az asszociációs mátrixot $\mathbf{A}$-t írjuk. A CMAC bázisfüggvényes háló, tehát az asszociációs vektor a bázisfüggvényes hálók $\boldsymbol{\varphi}(\mathbf{x})$ vektorának felel meg. Vagyis a CMAC válasza az (7.14) összefüggéshez hasonlóan kernel formában is megadható:

$$y(\mathbf{x}) = \mathbf{a}(\mathbf{x})^T \mathbf{A}^T \boldsymbol{\alpha} = \sum_{i=1}^{P} \alpha_i \left(\mathbf{a}(\mathbf{x})^T \mathbf{a}(\mathbf{x}_i)\right) = \sum_{i=1}^{P} \alpha_i K_i \left(\mathbf{a}(\mathbf{x})\right) . \tag{7.137}$$

A kernel függvény a CMAC-nál az asszociációs vektorok skalár szorzataként származtatható:

$$K_i \left(\mathbf{a}(\mathbf{x})\right) = \mathbf{a}(\mathbf{x})^T \mathbf{a}(\mathbf{x}_i) = K\left(\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{x}_i)\right) = K(\mathbf{x}, \mathbf{x}_i) . \tag{7.138}$$

Az $\boldsymbol{\alpha}$ vektor a kernel térbeli súlyvektor:

$$\boldsymbol{\alpha} = (\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{d} . \tag{7.139}$$

Előfordulhat, hogy az $\mathbf{A}\mathbf{A}^T$ mátrix nem invertálható, pl. numerikus problémák miatt. Ekkor a szokásos regularizáció alkalmazható, vagyis $\mathbf{A}\mathbf{A}^T$ helyett a regularizált $(\mathbf{A}\mathbf{A}^T + C^{-1}\mathbf{I})$ mátrixot invertáljuk. Vegyük észre, hogy ekkor valójában egy eltolásérték *(bias)* nélküli LS-SVM-et oldottunk meg, hiszen – mint láttuk – a mátrix inverziónál bevezettet regularizáció hatása ekvivalens egy olyan LS-SVM megoldásával, ahol a kritériumfüggvénybe a súlyvektor hosszának négyzete, mint minimalizálandó mennyiség is szerepel.

A kernel reprezentáció előnye nyilvánvaló: amíg az eredeti jellemzőtérbeli reprezentációnál a w_i súlyok száma a tanítópotok számánál jóval nagyobb (ld. a (6.50)–(6.52) összefüggéseket), addig a kernel térbeli α_i súlyok száma a tanítópontok számával egyezik meg. A különbség különösen többdimenziós esetben jelentős. A 6. fejezetben láttuk, hogy a súlymemória többdimenziós esetben még viszonylag kisméretű (néhány dimenziós) feladatnál is olyan óriási, hogy semmilyen eszközzel nem implementálható. A különféle módosítások (hash kódoló réteg bevezetése, a többdimenziós feladat egy- vagy kétdimenziós részfeladatokra való dekomponálása, stb.) célja az volt, hogy a hatalmas méretű memóriát megvalósítható méretűre redukáljuk. Láthatóan a kernel reprezentáció is egy eszköz az extrém komplexitás letörésére.

A kernel térben a CMAC válaszát kernel függvények súlyozott összegeként állítjuk elő. A kernel gépekre ugyanakkor az is jellemző, hogy a kernel függvényekből indulunk ki, és a jellemzőtérbeli leképezés nemlineáris transzformációját megvalósító $\boldsymbol{\varphi}(\mathbf{x})$ leképezést csak implicit módon határozza meg a kernel függvény. A kernel CMAC-nál nem ez a helyzet. Itt valóban a CMAC négyszögletes (bináris) bázisfüggvényeiből belső szorzat útján határozzuk meg a kernel függvényeket.

A bináris véges tartójú bázisfüggvények miatt kernel függvényként egydimenziós esetben a háromszögfüggvényt kapjuk. Többdimenziós esetben a kernel

függvény attól függ, hogy teljes lefedést vagy csak C-szeres lefedést alkalmazunk. Az eredeti Albus CMAC esetében az extrém komplexitás miatt a teljes lefedés szóba sem kerülhet, ha a bemeneti dimenzió 3–4-nél nagyobb. A kernel reprezentációnál azonban a komplexitást nem a jellemzőtér, hanem a kernel tér dimenziója határozza meg. A kernel tér dimenziója semmiképpen sem lehet nagyobb, mint a mintapontok száma, bármekkora – akár végtelen – is a jellemzőtér dimenziója. Ez azt jelenti, hogy a kernel reprezentáció esetén nincs szükség a lefedések számának korlátozására, a teljes lefedés is alkalmazható. A kétféle megoldás mindössze annyiban tér el egymástól, hogy más kernel függvényt eredményez, de a kernel függvények száma azonos lesz.

A teljes lefedés azzal az előnnyel is jár, hogy így elkerülhető a C-szeres lefedés következtében előálló approximációs képességbeli korlátozás. Láttuk, hogy többdimenziós esetben a C-szeres lefedésű bináris CMAC csak az additív függvényeket tudja hibátlanul megtanulni (ld. (6.58) konzisztencia-egyenlet). Teljes lefedés, illetve az annak megfelelő kernel változat mellett ez a korlátozás megszűnik.

A 7.20. ábrán a bináris CMAC kernel függvényét mutatja egydimenziós (a) és többdimenziós esetben. A többdimenziós CMAC-nál mind a C-szeres lefedésnek (b), mind a teljes lefedésnek (c) megfelelő kernel függvény látható. A teljes lefedéshez tartozó kernel függvényt kvantált változatban is megadjuk (d).

A kernel változatnak azonban van egy hátrányos következménye. Az ábrából látható, hogy a kernel függvények már nem bináris függvények, emiatt a kimenet előállításánál a szorzásokat ténylegesen el kell végezni, a szorzónélküli felépítés, mint előnyös tulajdonság elveszett. A véges tartójú bázisfüggvények következtében azonban a kernel függvények is véges tartójúak, ami azt jelenti, hogy adott bemenet mellett a kimenet előállításában a kernel függvényeknek csak egy töredéke vesz részt, így a szorzások száma is erősen korlátozott.

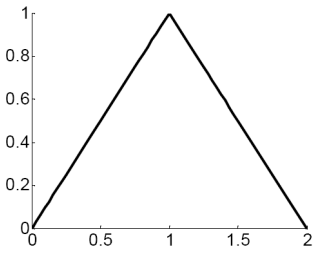
A válaszokat tekintve a kernel változat ekvivalens az eredeti változattal. Ez azt jelenti, hogy a kernel reprezentáció mellett is megjelenik a nagy általánosítási hiba mindazon esetekben, ha a C/t nem egész (t az egyes dimenziók mentén az egyenletesen elhelyezkedő mintapontok távolságát jelenti kvantumokban mérve). Ennek a hibának a jelentős mérséklését egy súlykiegyenlítő vagy súlysimító regularizáció alkalmazásával értük el (ld. 6 fejezet). A következőkben röviden azt foglaljuk össze, hogy ez a súlykiegyenlítő regularizáció a kernel CMAC esetében is alkalmazható [Hor06].

7.5.1. Kernel CMAC súlykiegyenlítő regularizációval

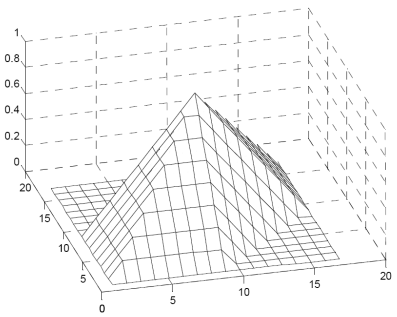
A regularizált kernel CMAC megoldását a minimalizálandó kritériumfüggvény felírásából származtathatjuk.

$$J(\mathbf{w},\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{j=1}^{P} e_j^2 + \frac{\lambda}{2}\sum_{j=1}^{P}\sum_{i:a_{ji}=1}\left(\frac{d_j}{C} - w_{ji}\right)^2, \qquad (7.140)$$

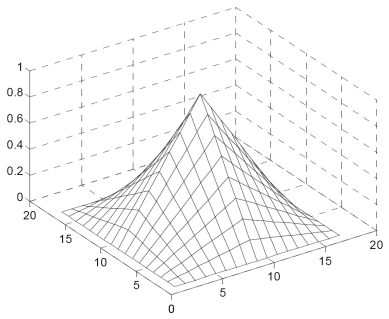
ahol a_{ji} a j-edik bemenethez rendelt asszociációs vektor i-edik bitje. (A továbbiakban a négyzetes hibatag együtthatóját γ-val jelöljük az eddigi C-vel szemben,



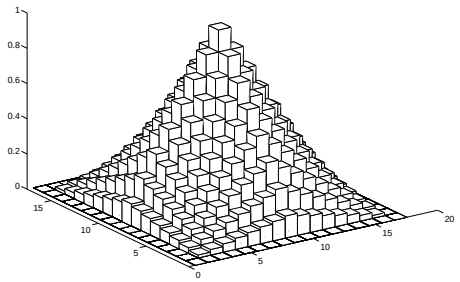
(a) egydimenziós eset



(b) kétdimenziós C-szeres lefedésű eset



(c) kétdimenziós teljes lefedésű eset



(d) kétdimenziós teljes lefedésű kvantált eset

7.20. ábra. A CMAC hálózat kernel függvényei

hogy a CMAC C paraméterével való jelölésbeli ütközést elkerüljük.) Látható, hogy a kritériumfüggvény az LS-SVM kritériumfüggvényének felel meg, azzal a különbséggel, hogy egy regularizációs taggal bővült. A regularizációs tag együtthatója λ. Valójában az LS-SVM kritériumfüggvénye is értelmezhető regularizált kritériumfüggvényként, így itt most két egymástól független regularizációt al-

kalmazunk.

Az LS-SVM-nél szokásos utat követve felírhatjuk a Lagrange egyenletet:

$$L(\mathbf{w},\mathbf{e},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{j=1}^{P} e_j^2 + \frac{\lambda}{2}\sum_{j=1}^{P}\sum_{i:a_{ji}=1}\left(\frac{d_j}{C} - w_{ji}\right)^2 - \\ - \sum_{j=1}^{P}\alpha_j\left(\mathbf{w}^T\mathbf{a}_j + e_j - d_j\right), \tag{7.141}$$

ami a következő alakra hozható:

$$L(\mathbf{w},\mathbf{e},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{j=1}^{P} e_j^2 - \sum_{j=1}^{P}\alpha_j\left(\mathbf{a}_j^T\mathrm{diag}(\mathbf{a}_j)\mathbf{w} + e_j - d_j\right) + \\ + \frac{\lambda}{2}\sum_{j=1}^{P}\frac{d_j^2}{C} - \lambda\sum_{j=1}^{P}\frac{d_j}{C}\mathbf{a}_j^T\mathrm{diag}(\mathbf{a}_j)\mathbf{w} + \frac{\lambda}{2}\sum_{j=1}^{P}\mathbf{w}^T\mathrm{diag}(\mathbf{a}_j)\mathbf{w}, \tag{7.142}$$

ahol $\mathrm{diag}(\mathbf{a}_j)$ egy olyan diagonál mátrix, melynek főátlójában $\mathbf{a}_j$ található. A megfelelő deriválások és behelyettesítések elvégzése után itt is egy lineáris egyenletrendszert kapunk, melynek megoldása:

$$\boldsymbol{\alpha} = \left(\mathbf{K_D} + \frac{1}{\gamma}\mathbf{I}\right)^{-1}\left(\mathbf{I} - \frac{\lambda}{C}\mathbf{K_D}\right)\mathbf{d}. \tag{7.143}$$

Itt $\mathbf{D} = \sum_{j=1}^{P}\mathrm{diag}(\mathbf{a}_j)$ és

$$\mathbf{K_D} = \mathbf{A}(\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{A}^T \tag{7.144}$$

a regularizált esetnek megfelelő kernel mátrix. Látható, hogy a regularizáció hatására a kernel mátrix megváltozott. A regularizáció tehát módosítja a kernel függvényt. A (7.138) által megadott kernel függvény helyett a

$$K_{Dj}\left(\mathbf{a}(\mathbf{x})\right) = \mathbf{a}(\mathbf{x})^T(\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{a}(\mathbf{x}_j) = K_D\left(\mathbf{a}(\mathbf{x}),\mathbf{a}(\mathbf{x}_j)\right) = K_D(\mathbf{x},\mathbf{x}_j) \tag{7.145}$$

kernel függvényt kapjuk. Mivel $\mathbf{D}$ a tanítópontok elhelyezésének függvénye, valójában adatfüggő kernel függvényt kaptunk. A kernel függvény azonban csak a tanítópontoknak a bemeneti térben való elhelyezkedésétől függ, a kívánt válaszoktól nem.

A háló válasza a regularizált esetben a következőre adódik:

$$y(\mathbf{x}) = \mathbf{a}(\mathbf{x})^T(\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{A}^T\left[\boldsymbol{\alpha} + \frac{\lambda}{C}\mathbf{d}\right]. \tag{7.146}$$

Ez az alábbi formában is felírható:

$$y(\mathbf{x}) = \sum_{j=1}^{P}\beta_j K_D(\mathbf{x},\mathbf{x}_j) = \mathbf{K}_\mathbf{D}^T(\mathbf{x})\boldsymbol{\beta}, \tag{7.147}$$

ahol a $\boldsymbol{\beta}$ együtthatóvektor a módosított kernel függvényekkel felépített kernel térbeli megoldás súlyvektorát jelöli:

$$\boldsymbol{\beta} = \boldsymbol{\alpha} + \frac{\lambda}{C}\mathbf{d}\,. \tag{7.148}$$

A regularizálás a kernel reprezentációs megoldás mellett is biztosítja a háló sima válaszát, tehát az általánosítóképesség jelentős javítását. A regularizált kernel CMAC tehát nemcsak a komplexitás jelentős redukcióját biztosítja, hanem az approximációs- és általánosító-képességet is javítja.

A regularizált kernel CMAC azon túl, hogy egy jobb tulajdonságú tanuló rendszert eredményez arra is példa, hogy regularizációs formában az alapkritérium mellett további mellékfeltételek is belefogalmazhatók egy feladatba. Az eljárás hatékonysága attól függ, hogy az így definiált optimalizálási feladat milyen nehézségek árán oldható meg.

7.6. A kernel gépek összefoglaló értékelése

A kernel megközelítés az adatokból történő tanuló rendszerek konstrukciójánál két szempontból hozott újat.

Az első újdonság a megoldás komplexitásához köthető. A kernel trükk alkalmazása a komplexitást a tanítópontok száma által meghatározott mértékben mindenképpen korlátozza. A kernel reprezentáció önmagában előnyös mindazon esetekben, amikor a bemeneti tér vagy a bemenetből nemlineáris transzformációval előállított jellemzőtér sokdimenziós. A kernel reprezentáció azt teszi lehetővé, hogy a feladat komplexitása a bemeneti tér, illetve a jellemzőtér dimenziójától független lesz. Ebből adódóan a jellemzőtér tetszőlegesen sokdimenziós, akár végtelen dimenziós is lehet. Ezt az előnyt a kernel gépek mindegyike biztosítja.

A második, az összes kernel gép közül az elsősorban Vapnik nevéhez köthető szupport vektor gépekhez (SVM) kapcsolódik. A szupport vektor gépek azon túl, hogy a tanítópontokban értelmezett hiba minimalizálását célozzák, azt is célnak tekintik, hogy a megoldás „optimális komplexitású" legyen, ezáltal a modell minőségéről is tudjanak valamit állítani. A modell komplexitásának „mérésére" a szupport vektor gépek a VC-dimenziót alkalmazzák. Az SVM a VC-dimenzió korlátozásával a strukturális kockázat minimalizálás (SRM) elvét valósítja meg. Ennek következménye, hogy az SVM hibájára – a valódi kockázatra – felső korlátok fogalmazhatók meg. Az SRM elv érvényesítését a szupport vektor gépek a súlyvektor hosszának minimalizálása útján biztosítják. A felső korlátok és az SVM egyes változatai közötti kapcsolatot pedig az teremti meg, hogy a súlyvektor hosszának minimalizálása a margó maximalizálásán keresztül a VC-dimenziót is minimalizálja.

A súlyvektor hosszának minimalizálása a többi kernel gép származtatásának is része, vagy része lehet. Ez általánosan is „simító" regularizációs hatást eredményez. Az LS-SVM és a ridge regressziós megoldásoknál, valamint ezek változatainál a simító hatás ellenére jelenleg nincs olyan érvényes eredmény, amely az SVM-hez hasonló felső korlátokat tudna megállapítani. Ez annak ellenére van

így, hogy a gyakorlati tapasztalatok szerint a kétféle kernel gép jellegében nagyon hasonló eredményekre vezet. Az SVM és az LS-SVM ilyen szintű kapcsolatának részletes elemzése még hátra van.

Az SVM és a többi kernel gép között abban a tekintetben is eltérés van, hogy a szupport vektor gépek ritka megoldást adnak, kiválogatják azokat a tanítópontokat, melyek a megoldáshoz szükségesek, amelyek a megoldást „tartják". Hasonlóan ritka megoldás az LS-SVM-nél, illetve a ridge regressziónál is biztosítható (pl. LS^2-SVM, RRKRR), ezekben az esetekben azonban ez nem az alapeljárás része, hanem az alapeljárást követő redukció eredménye.

Bár a kernel gépek alapvetően egy matematikai eljárásként jelennek meg, neuronhálóként is értelmezhetők. Mint neuronhálók, a statikus előrecsatolt hálók családjába tartoznak. Ez lehetővé tesz, hogy a klasszikus előrecsatolt statikus hálózatokkal – MLP, bázisfüggvényes hálózatok – is összevetésre kerüljenek. Az MLP-vel való összehasonlítás a legnehezebb, hiszen a kétféle megoldás strukturálisan is eltérő. A kernel gépek és elsősorban az SVM előnye, hogy a komplexitásproblémára megoldást adnak. Minőség tekintetében viszont nem lehet rangsort felállítani, még akkor sem, ha a minőségre vonatkozó korlátok a klasszikus hálókra nem vonatkoznak. A korlátok hiánya ugyanis nem azt mondja, hogy pl. egy MLP-vel nem tudunk hasonlóan jó, vagy netán jobb megoldást is elérni, mint egy SVM-mel, csupán azt, hogy az MLP-vel elérhető valódi kockázatról általános eredményt jelenleg nem tudunk megfogalmazni.

A bázisfüggvényes hálókkal való összehasonlításnál az SVM-et és a többi kernel gépet külön kell vizsgálni. Míg az LS-SVM és a ridge regresszió a különböző változatokkal együtt valójában a bázisfüggvényes hálózatok eltérő, de ekvivalens megvalósítása, az SVM a kiinduló probléma más megfogalmazása miatt elvileg is más eredményt ad. Erre vonatkozó kísérleti eredmények is ezt a különbséget támasztják alá. (ld. pl. az RBF és a Gauss kerneles SVM összehasonlítására vonatkozó eredményt [Sch96]).

A fejezet végén bemutatott kernel CMAC egy konkrét példa a kernel reprezentáció hasznára bázisfüggvényes hálózatoknál. Ennek specialitása a véges tartójú bázisfüggvény és az ebből következő véges tartójú kernel függvény. A regularizált kernel CMAC pedig arra egy példa, hogy regularizációs formában az alapkritérium mellett további mellékfeltételek is belefogalmazhatók egy feladatba. Az eljárás hatékonysága attól függ, hogy az így definiált optimalizálási feladat milyen nehézségek árán oldható meg.

A kernel gépekkel kapcsolatos első eredmények alapvetően a kilencvenes években születtek. A számos nyitott kérdés miatt azonban folyamatosan jelennek meg újabb és újabb eredmények, melyek többirányú kutatás eredményei. A fontosabb irányok: a kernel függvény megválasztása, adaptív és adatfüggő kernel konstrukció, az általánosító-képességre vonatkozó újabb és kevésbé pesszimista korlátok meghatározása, továbbá a ritka megoldások elérése és a számítási komplexitás további mérséklése. Fontos kérdés az is, hogy a kernel gépek alkalmazásával minél több tapasztalat gyűljön össze, melyek a hatékony gyakorlati felhasználást segíthetik. Néhány alkalmazási területről a következő fejezetben lesz szó.

Feladatok

7.1 Mutassa meg, hogy egy pontnak a $\mathbf{w}^T\mathbf{x}+b=0$ egyenlettel definiált lineáris szeparáló felülettől való távolsága

$$r = \frac{\left|\mathbf{w}^T\mathbf{x}+b\right|}{\|\mathbf{w}\|}\,.$$

7.2 A XOR probléma megoldásában az $\left(\mathbf{x}^T\mathbf{x}+1\right)^p$ polinomiális kernel függvény alkalmazható, ha $p=2$. Vizsgálja meg a polinomiális kernel alkalmazhatóságát más pozitív egész p értékekre.

7.3 Konstruáljon szupport vektor gépet az egydimenziós sinc függvény approximálására $\left(\mathbf{x}^T\mathbf{x}+1\right)^p$ polinomiális kernel függvény mellett. Vizsgálja meg p szerepét az approximáció minőségét illetően.

7.4 Készítsen egy osztályozós mintakészletet (ld. 7.21. ábra), valamint egy paraméterezhető (kernel paraméterek – pl. RBF esetén σ, C, ε) tesztkörnyezetet, ami egy tanítókészlet alapján megtanít egy SVM osztályozót és meghatározza a hibás osztályozások számát (arányát) a tesztkészletre. (A feladatok során mindig legyen figyelemmel a megoldás hibájára, a számítási időre valamint a szupport vektorok számára.)

a) Bontsa fel a mintapontokat tartalmazó fájlokat egy tanító és egy teszt készletre. A fájlban a pontok elrendezése (sorrendje) nem ismert, ezért célszerű figyelni arra, hogy mindkét osztályból és egyenletesen válasszunk elemeket. (Segítség: keverje össze az adatfájlt a válogatás előtt.)

b) Határozza meg az SVM egy olyan paraméterezését, melyre az osztályozás elfogadható eredményt ad. (Segítség: használjon kicsi – max. 200 mintapontból álló – tanítókészletet és Gauss kernelt.)

c) Vizsgálja meg, hogy hogyan függ az eredmény a kernel függvény megválasztásától. Tesztelje a hálózatot az egyes kernelek paramétereinek változtatásával is.

d) Hogyan függ az eredmény a C paraméter megválasztásától?

e) Vizsgálja meg, hogy hogyan függ az eredmény és a futási idő a tanítópontok számától. (Segítség: célszerű kisebb tanítókészletből kiindulni és addig növelni a pontok számát, amíg a számítás kivárható.)

7.5 Készítsen egy mintakészletet a $\mathrm{sinc}(x)$ függvény pontjaiból, valamint egy paraméterezhető tesztkörnyezetet (kernel paraméterek – pl. RBF esetén σ, C, ε), ami egy tanítókészlet alapján megtanít egy SVM regresszort és meghatározza a regresszió négyzetes hibáját.

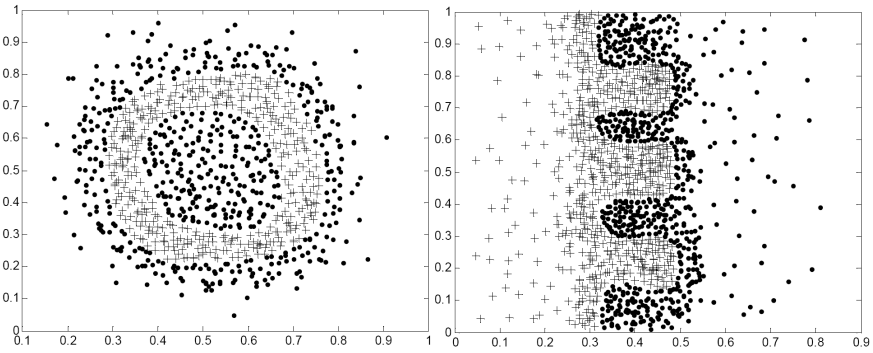
a) Bontsa fel a mintapontokat egy tanító és egy teszt készletre (Segítség: Vegye például az adatfájl minden n-edik elemét, vagy keverje össze a mintapontokat és ezt ossza ketté.)

b) Határozza meg az SVM egy olyan paraméterezését, mely elfogadható eredményt ad. (Segítség: használjon kis – max. 200 mintapont – tanítókészletet és Gauss kernelt.)

c) Vizsgálja meg, hogy hogyan függ az eredmény a kernel függvény megválasztásától. Tesztelje a hálózatot az egyes kernelek paramétereinek változtatásával is.

d) Hogyan függ az eredmény a C paraméter megválasztásától?

e) Vizsgálja meg, hogy hogyan függ az eredmény és a futási idő a tanítópontok számától. (Segítség: célszerű kisebb tanítókészlettel indulni és addig növelni a pontok számát, amíg a számítás kivárható.)



7.21. ábra. Két lehetséges mintakészlet az osztályozós feladatokhoz

7.6 Az előző regressziós feladatokat végezze el LS-SVM illetve LS^2-SVM használatával is.

7.7 Származtassa a magasabbrendű CMAC kernel függvényét, ha a CMAC hálózat bázsifüggvénye k-adrendű B-spline.

7.8 Mutassa meg, hogy az RRKRR eljárásnál a bázisvektorok kiválasztásának alapját képező approximációs hiba (7.128) és (7.129) összefüggése ekvivalens.

Irodalomjegyzék

[Sai88]

[Sch02]

[Vap98]

[Sző72]

[Cri98]

[Ama99]

[Vin00]

[Xio05]

[Mer06]

[Bos92]

[Vap95]

[Cris00]

[Sha02]

[Cha02]

[Kwo03]

[Sch99]

[Vap79]

[Osu98]

[Pla99]

[Lee01a]

[Lee01b]

[Suy00]

[Suy02]

[Val04]

[Pre02]

[Gol89]

[Ciz04]

[Hol77]

[Hub81]

[Hoe70]

[Caw02]

[Bau01]

[Hor06]

[Sch96]

Tárgymutató