Fəsil 5.

Dəstək Vektor Maşınları (Support Vector Machines)

Dəstək Vektor Maşını (DVM) həm xətti, həm də qeyri-xətti təsnifat, reqressiya və hətta yeni nümunələrin aşkarlanmasını (novelty detection) icra etməyi bacaran güclü və çoxyönlü maşın öyrənməsi modelidir. DVM-lər kiçikdən orta ölçüyə qədər, qeyri-xətti verilənlər çoxluqları (datasets) (yüzdən minə qədər nümunəsi olan) ilə, xüsusilə təsnifat məsələlərində çox yaxşı işləyir. Lakin, görəcəyiniz kimi, çox böyük verilənlər çoxluqları ilə elə də yaxşı uyğunlaşmırlar.
Bu fəsil DVM-lərin əsas anlayışlarını, onlardan necə istifadə etməli olduğumuzu və işləmə prinsiplərini izah edəcək. Gəlin başlayaq!

Xətti DVM Klassifikasiyası

DVM-in arxasındakı əsas ideya ən yaxşı bəzi vizuallarla izah edilə bilər. Fiqur 5-1 4-cü fəslin sonunda təqdim edilən iris verilənlər çoxluğunun (dataset) bir hissəsini göstərir. İki sinif bir xətt vasitəsilə aydın şəkildə ayrıla bilər (*xətti ayrılabiləndirlər*). Soldakı qrafik 3 mümkün təsnifatçının qərar sərhədlərini göstərir. Qərar sərhədləri qırıq xətlərlə ifadə olunmuş modelin o qədər pisdir ki, heç sinifləri düzgün şəkildə ayırmır. Digər iki model təlim çoxluğu üzərində mükəmməl işləyir, lakin onların qərar sərhədləri nümunələrə o qədər yaxınlaşır ki, bu modellər, yəqin ki, yeni nümunələr üzərində belə yaxşı performans göstərməyəcəklər. Əksinə, qrafikdə sağ tərəfdəki qalın xətt DVM təsnifatçının qərar sərhədlərini göstərir; bu xətt nəinki iki sinifi ayırır, həm də ən yaxınındakı təlim nümunələrindən mümkün qədər uzaqda dayanır. DVM təsnifatçının siniflər arasındakı mümkün olan ən geniş küçəyə sığdığını təsəvvür edə bilərsiz.
Buna geniş kənar payı təsnifatı (large margin classification) deyilir.

Diqqət edin ki, “küçənin kənarına” yeni təlim nümunələri əlavə etmək qərar sərhədinə heç bir təsir göstərməyəcək: o tamamilə küçənin kənarındakı nümunələrə əsasən müəyyən olunur (və ya “dəstəklənir”). Bu nümunələr dəstək vektorları adlanır. (Onlar Fiqur 5-1-də dairələnmişdir).

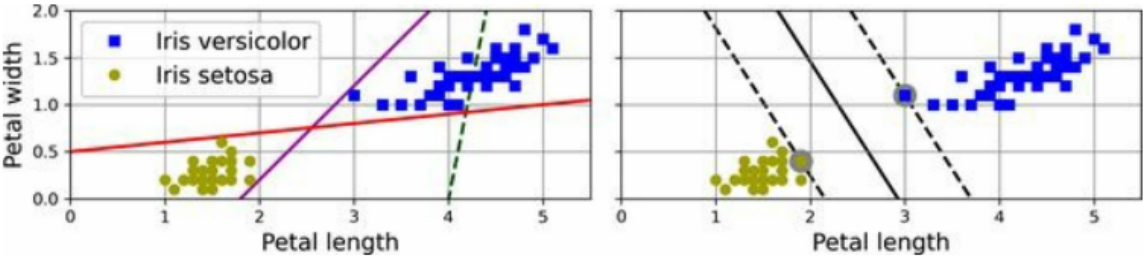
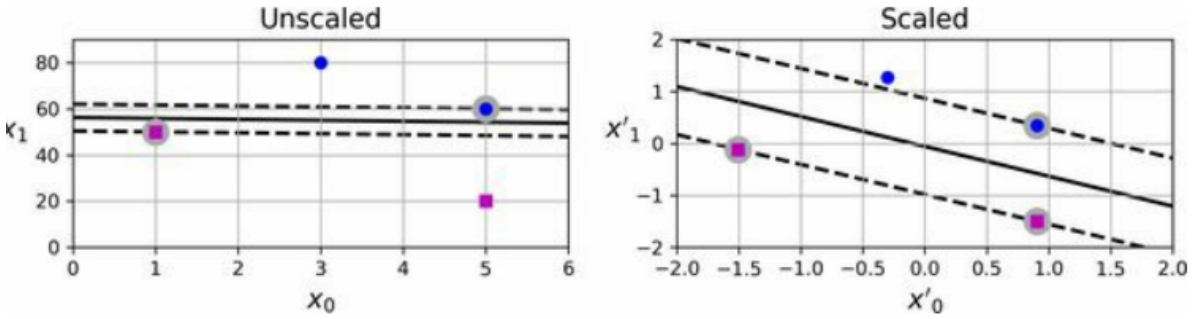


Figure 5-1. Large margin classification

Fiqur 5-1. Geniş kənar payı təsnifatı

XƏBƏRDARLIQ

DVM-lər, Fiqur 5-2-də görə biləcəyiniz kimi, atribut miqyaslarına qarşı həssasdır. Soldakı qrafikdə, şaquli şkala üfüqi şkaladan çox böyükdür, buna görə də, mümkün olan ən geniş küçə üfüqiyə daha yaxındır. Atribut miqyaslandırılmasından sonra (məsələn, Scikit-Learn-in StandartScaler-i), sağ qrafikdəki qərar sərhəddi çox daha yaxşı görünür.

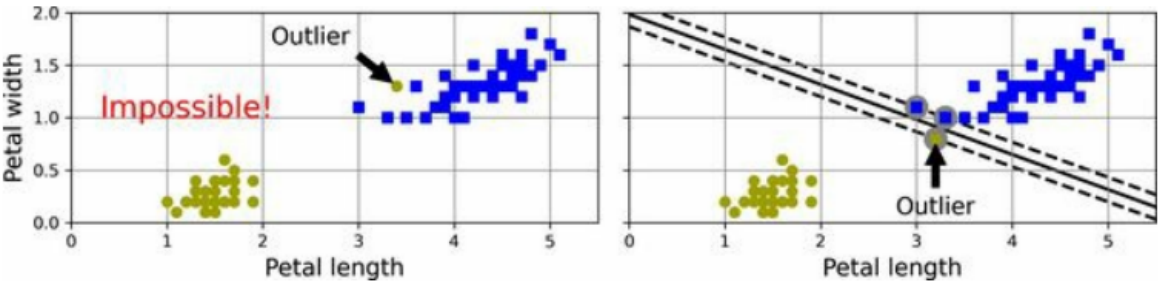


Fiqur 5-2. Atribut miqyaslarına həssaslıq

Yumşaq kənar payı təsnifatı

Əgər biz ciddi şəkildə bütün nümunələrin küçədən kənarda və düzgün tərəfdə olmağa məcbur etsək, buna *sərt kənar payı təsnifatı* deyilir. Sərt kənar payı tənsifatının iki əsas problemi var. Birincisi, təkcə verilənlər xətti ayrılabilən (linearly seperable) olduğu zaman işləyir. İkincisi, kənar dəyərlərə (outliers) qarşı həssasdır. Fiqur 5-3 sadəcə bir əlavə kənar dəyərlə iris verilənlər çoxluğunu (dataset) göstərir: sol tərəfdə sərt kənar payı tapmaq mümkün deyil; sağda isə qərar sərhəddi Fiqur 5-1də

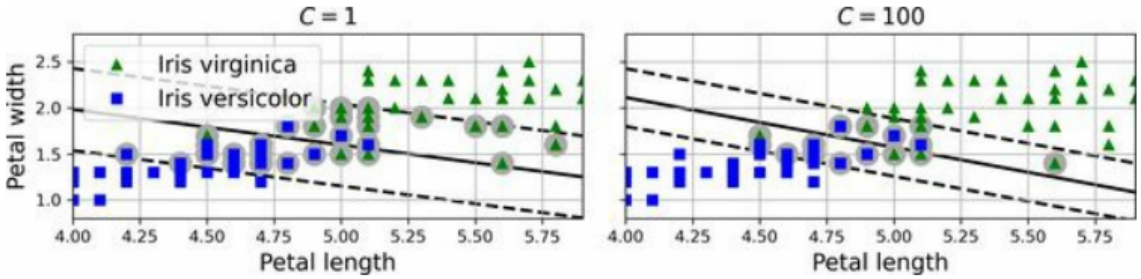
gördüyümüz, kənar dəyərlər (outliers) olmayan qrafikdən çox fərqlənir və böyük ehtimalla, model də ümumiləşdirmə etməyəcək.



Fiqur 5-3. Sərt kənar payının kənar dəyərlərə həssaslığı

Bu problemlərdən qurtulmaq üçün daha elastik bir model istifadə etməliyik. Hədəf küçəni mümkün qədər geniş saxlayarkən, kənar payı pozuntularının (məs., küçənin ortasında və ya yalnış tərəfdə yerləşdirilən nümunələr) minimuma endirilməsidir. Buna *yumşaq kənar payı təsnifatı* deyilir.

Scikit-Learn-dən istifadə edərək DVM modeli yaradan zaman, tənzimləmə hiperparametri C daxil olmaqla bir neçə hiperparametri müəyyənləşdirə bilərsiniz. Əgər ona aşağı dəyər versəniz, nəticə Fiqur 5-4-dəki model olacaq. Yüksək dəyər verərək sağdakı modeli ala bilərsiniz. Gördüyünüz kimi, C-ni azaltmaq küçəni daha geniş edir. Lakin eyni zamanda daha çox kənar payı pozuntularına gətirib çıxarır. Başqa sözlərlə, C-ni azaltmaq daha çox küçəni dəstəkləyən nümunə ilə nəticələnir, beləcə burada daha az həddən artıq öyrənmə (overfitting) riski olur. Lakin əgər onu həddən artıq azaltsanız, onda model underfitting edəcək, burdakı nümunədən göründüyü kimi: C=100 olan model C=1 olandan daha yaxşı ümumiləşdirmə edə biləcək kimi görünür.



Fiqur 5-4. Geniş kənar payı (sol) vs. daha az kənar payı pozuntuları (sağ)

MƏSLƏHƏT

Əgər DVM modeliniz həddən artıq öyrənibsə (overfitting), onu C-ni azaltmaqla tənzimləməyi yoxlaya bilərsiniz.

Aşağıdakı Scikit-Learn kodu iris verilənlər çoxluğunu (dataset) yükləyir və *Iris virginica* güllərini müəyyən etmək xətti DVM təsnifatçısını öyrədir. Emal xətti (pipeline) ilk öncə atributları miqyaslandırır, daha sonra C=1 götürərək LinearSVC (xətti DVM) istifadə edir:

```
from sklearn.datasets import load_iris
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
iris = load_iris(as_frame=True)
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = (iris.target == 2) # Iris virginica
svm_clf = make_pipeline(StandardScaler(),
        LinearSVC(C=1, random_state=42))
svm_clf.fit(X, y)
```

Nəticədə alınan model Fiqur 5-4-də göstərilmişdir.
Daha sonra, həmişəki kimi, modeli öngörmələr vermək üçün istifadə edə bilərsiniz:

```
>>> X_new = [[5.5, 1.7], [5.0, 1.5]]
>>> svm_clf.predict(X_new)
array([ True, False])
```

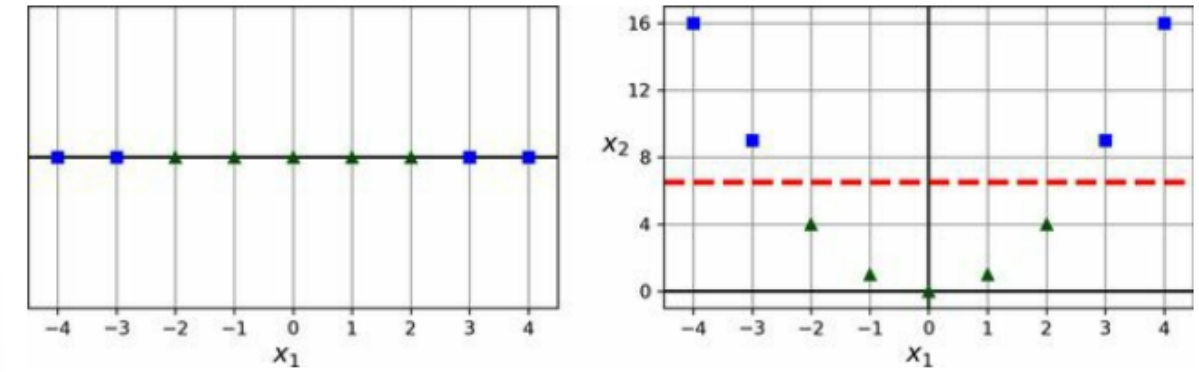
Birinci bitki *Iris virginica* kimi təsnif olunub, ikinci isə yox. Gəlin DVM-in bu öngörmələri vermək üçün istifadə etdiyi xallara baxaq. Bunlar hər bir nümunə və qərar sərhədi arasında işarələnmiş məsafəni ölçür:

```
>>> svm_clf.decision_function(X_new)
 array([ 0.66163411, -0.22036063])
```

LogisticRegression-dan fərqli olaraq, LinearSVC siniflərin ehtimallarını təxmin etmək üçün predict_proba() metoduna sahib deyil. Buna baxmayaraq, əgər LinearSVC yerinə SVC class-ı (qısa zamanda bəhs ediləcək) istifadə etsəniz, və onun probability (ehtimal) hiperparametrini True (Doğru) etsəniz, model təlimin sonunda DVM qərar funksiyasının nəticələrini təxmin edilmiş ehtimallara çevirmək üçün əlavə bir model fit edəcək . Arxa planda, bunun baş verməsi hər bir təlim nümunəsi üçün nümunələrdən kənar (out of sample) öngörmələr yaratmaq məqsədilə 5-qatlı çarpaz validasiyanın (5-fold cross-validation) istifadə olunmasını, daha sonra LogisticRegression modelinin təlim edilməsi tələb edir, buna görə də təlimi nəzərəçarpan dərəcə yavaşladacaq. Bundan sonra, predict_proba() və predict_log_proba() metodları istifadə oluna biləcək.

Qeyri-xətti DVM təsnifatı

DVM təsnifatçılarının səmərəli olmasına və çox vaxt təəccüblü şəkildə yaxşı işləməsinə baxmayaraq, verilənlər çoxluqlarının (datasets) əksəriyyəti xətti ayrıla bilən olmağa yaxın belə deyil. Qeyri-xətti verilənlər çoxluqlarını həll etməyin bir yolu da daha çox atribut əlavə etməkdir, məsələn, polinom atributlar (4-cü fəsildə etdiyimiz kimi); bəzi hallarda bu xətti ayrılabilən verilənlər çoxluğu (linearly seperable dataset) ilə nəticələnir. Fiqur 5-5-in sol tərəfindəki qrafikə nəzər salın: yalnız bir atribut, x_1-dən ibarət bəsit bir verilənlər çoxluğunu göstərir. Gördüyünüz kimi, bu verilənlər çoxluğu xətti ayrıla bilən deyil. Lakin ikinci atribut kimi $x_2 = (x_1)^2$ əlavə etsəz, alınan 2-ölçülü verilənlər çoxluğu mükəmməl şəkildə xətti ayrıla bilən olacaq.



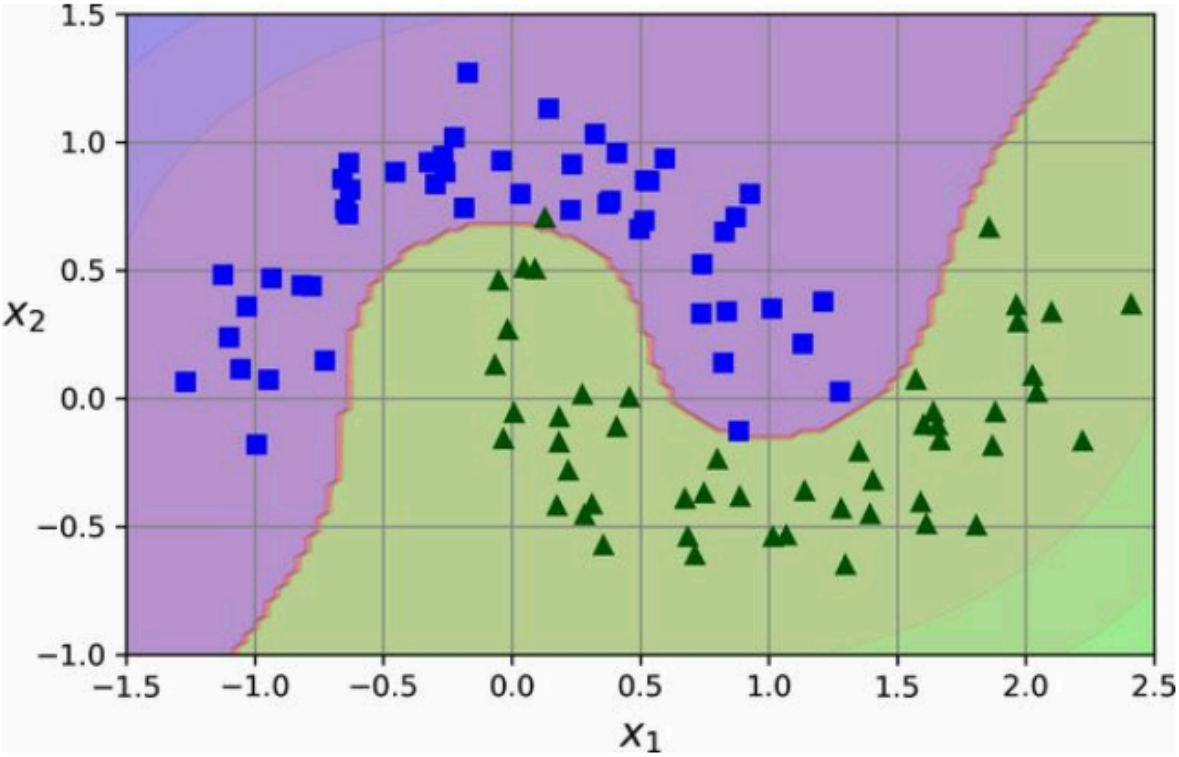
Fiqur 5-5. Verilənlər çoxluğunu xətti ayrıla bilən etmək üçün atributlar əlavə etmək

Scikit-Learn istifadə edərək bu ideyanı tətbiq etmək üçün PolynomialFeatures transformeri (“Polinom Reqressiya”-da bəhs olunub), ardıyca StandartScaler və LinearSVC təsnifatçıdan ibarət bir pipeline yarada bilərsiz. Gəlin bunu aylar verilənlər çoxluğunda (the moons dataset) test edək. Bu, verilənlər nöqtələrinin 2 iç-içə keçmiş ayparalar şəklində olduğu bir oyuncaq verilənlər çoxluğudur (Fiqur 5-6-ya baxın). Bu verilənlər çoxluğunu make_moons() funksiyasından istifadə edərək yarada bilərsiniz.

```
from sklearn.datasets import make_moons
from sklearn.preprocessing import PolynomialFeatures

X, y = make_moons(n_samples=100, noise=0.15, random_state=42)

polynomial_svm_clf = make_pipeline(
        PolynomialFeatures(degree=3),
        StandardScaler(),
        LinearSVC(C=10, max_iter=10_000, random_state=42)
       )
       polynomial_svm_clf.fit(X, y)
```



Fiqur 5-6. Polinom atributlardan istifadə edən xətti DVM təsnifatçı

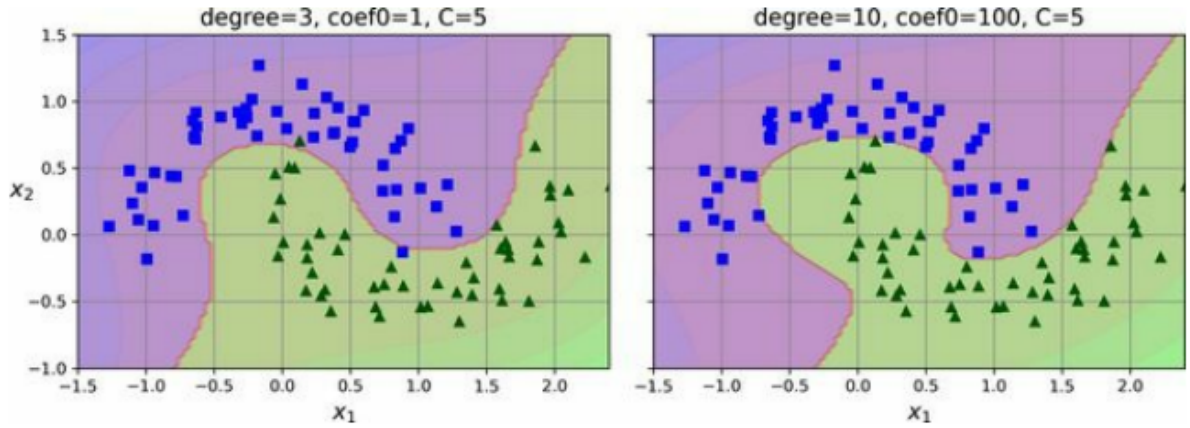
Polinom Kernel

Polinom atributlar əlavə etmək praktikada asandır və bütün növ maşın öyrənmə alqoritmləri ilə əla işləyə bilir (sadəcə DVM-lərlə deyil). Bununla bərabər, aşağı polinom dərəcəsində çox mürəkkəb verilənlər çoxluqları ilə işləyə bilmir, yuxarı polinom dərəcəsiylə isə çox sayda atributlar yaradaraq modeli yavaşladır.

Xoşbəxtlikdən, DVM-lərdən istifadə edərsən *kernel hiyləsi* (kernel trick) (fəsilin davamında izah olunub) adlanan möcüzəvi bir riyazi texnikadan istifadə edə bilərsiniz. Kernel hiyləsi, hətta daha yüksək dərəcədə belə, polinom atributları həqiqətən əlavə etmədən onları əlavə edəndə aldığımız nəticəni almağı mümkün edir. Bu o deməkdir ki, atributların sayında kombinatorial partlayış (combinatorial explosion) baş vermir. Bu hiylə SVC class-ı ilə edilir. Gəlin onu aylar (moons) verilənlər çoxluğunda test edək:

```
from sklearn.svm import SVC
poly_kernel_svm_clf = make_pipeline(StandardScaler(),
                    SVC(kernel="poly", degree=3, coef0=1, C=5))
poly_kernel_svm_clf.fit(X, y)
```

Bu kod Fiqur 5-7-də göstərilən üçüncü dərəcəli polinom kerneldən istifadə edərək DVM təsnifatçını təlim edir. Sağ tərəfdə onuncu dərəcəli polinom kernel istifadə edən başqa bir DVM təsnifatçı var. Aydındır ki, əgər sizin modeliniz overfit edirsə, polinomun dərəcəsini azaltmaq istəyə bilərsiniz. Əksinə, əgər underfit edirsə, artırmağı yoxlaya bilərsiniz. coef() hiperparametri modelin aşağı dərəcələli hədlərlə müqayisədə yüksək dərəcəli hədlərdən nə qədər təsirləndiyini kontrol edir.



Fiqur 5-7. Polinom kernel ilə DVM təsnifatçı

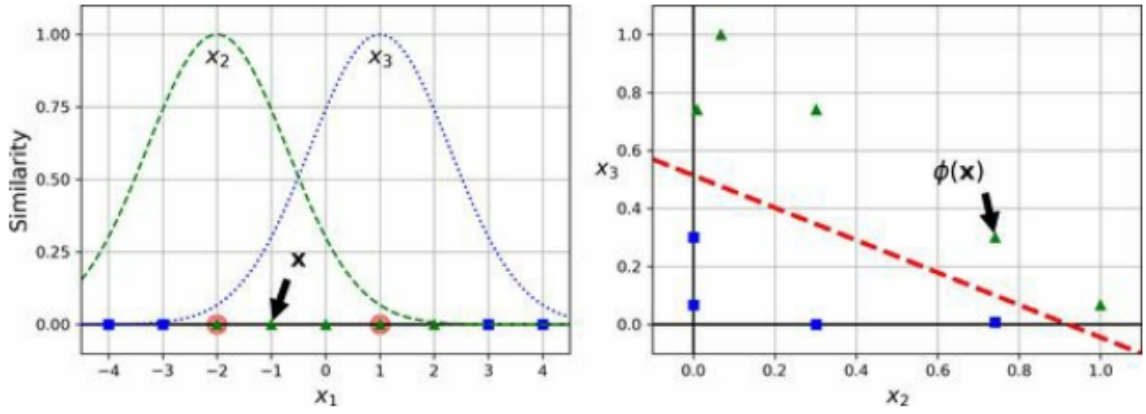
MƏSLƏHƏT

Hiperparametrlərin adətən avtomatik müəyyənləşdirilməsinə (məsələn, random axtarışdan istifadə edərək) baxmayaraq, hər bir parametrin əslində nə etdiyini və digər hiperparametrlərlə necə əlaqədə olduğunu bilmək yaxşıdır: beləcə, çox daha kiçik bir sahədə axtarış edə bilərsiniz.

Oxşarlıq Atributları (Similarity Features)

Qeyri-xətti problemləri həll etməyin bir başqa yolu da ikinci fəsildə coğrafi oxşarlıq atributlarını əlavə edərkən etdiyimiz kimi, hər bir nümunənin xüsusi bir nöqtəyə nə qədər bənzədiyini ölçən oxşarlıq funksiyasından istifadə edərək hesablanmış atributlar əlavə etməkdir. Məsələn, əvvəlki 1-ölçülü verilənlər çoxluğunu (dataset) götürək və $x_1 = -2$ və $x_1 = 1$ nöqtələrini işarələyək (Fiqur 5-8, soldakı qrafikə baxın). Daha sonra, oxşarlıq funksiyasını $\gamma = 0.3$ parametri ilə Gaussian RBF olaraq müəyyənləşdirək. Bu, 0-dan (işarələdiyimiz nöqtədən çox uzaqda) 1-ə (işarə elədiyimiz nöqtəyə) qədər dəyişən zınqırov formalı funksiyadır.

Artıq yeni atributları hesablamağa hazırıq. Məsələn, $x_1 = -1$ nöqtəsindəki nümunəyə baxaq: birinci işarələdiyimiz nöqtədən 1 məsafə, ikincidən isə 2 məsafə uzaqlıqdadır. Buna görə də, onun yeni atributları $x_2 = \exp(-0.3 \times 1) \approx 0.74$ və $x_3 = \exp(-0.3 \times 2) \approx 0.30$-dir. Fiqur 5-8-də sağ tərəfdəki qrafik dəyişmiş (transformed) verilənlər çoxluğunu göstərir (orjinal atributları buraxaraq). Görə biləcəyiniz kimi, verilənlər çoxluğu indi xətti ayrıla biləndir (linearly seperable).



Fiqur 5-8. Gaussian RBF istifadə edərək yaradılan oxşarlıq atributları

İşarələyəcəyimiz xüsusi nöqtələri necə seçəcəyimiz sizə maraqlı gələ bilər. Ən bəsit yanaşma verilənlər çoxluğundakı bütün nümunələrin yerini işarələməkdir. Bunu etmək çoxlu ölçülər yaradır və dəyişdirilmiş verilənlər

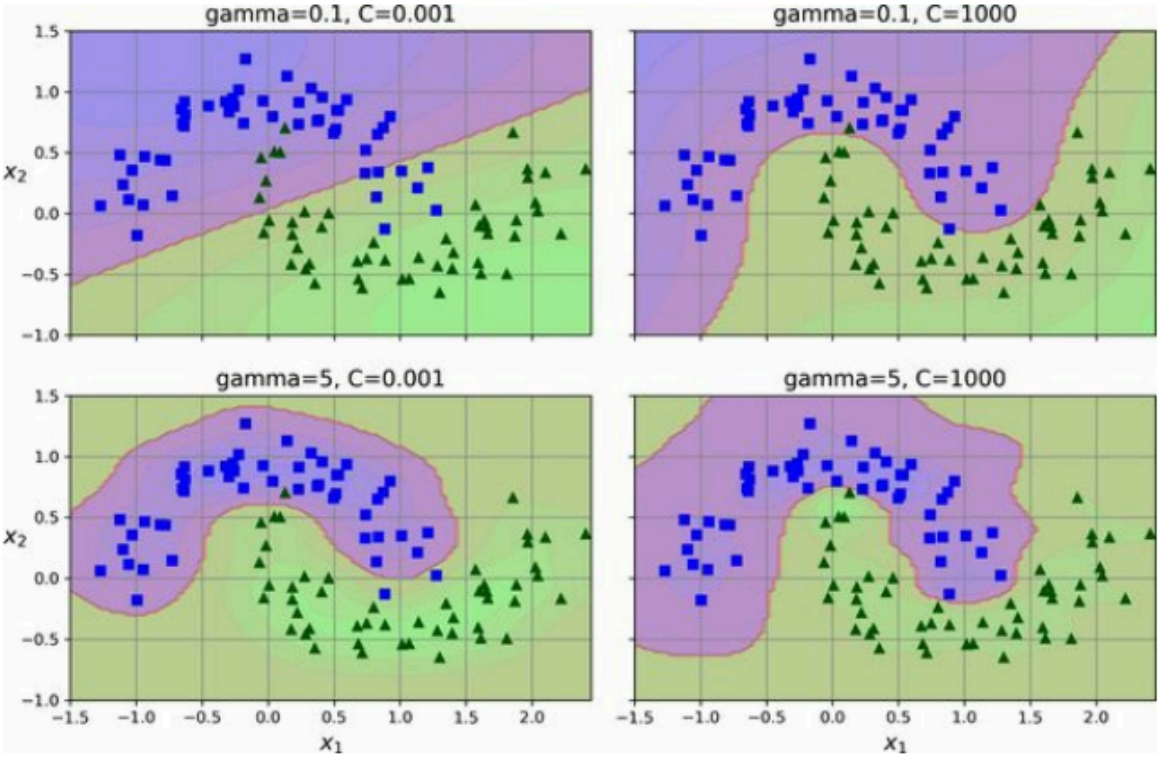
çoxluğunun xətti ayrıla bilən olma şansını artırır. Bunun mənfi tərəfi odur ki, m nümunəyə (instances) və n atributa (features) sahib təlim çoxluğu (training set) m nümunə və m atributa sahib bir təlim çoxluğuna çevrilir (orjinal atributları buraxdığımızı güman etsək). Əgər təlim çoxluğunuz çox böyükdürsə, bir o qədər də çox sayda atribut əldə edirsiniz.

Qauss RTF(Radial Təməl Funksiyası) Kerneli

Polinom atributlar metodu kimi, oxşarlıq atributları metodu da istənilən maşın öyrənmə alqoritmi ilə faydalı ola bilər, lakin bütün əlavə atributları hesablamaq hesab intensiv gələ bilər (xüsusilə də, böyük təlim çoxluqlarında). Bir daha, kernel hiyləsi özünün DVM (SVM) sehrini edir, həqiqətən o qədər atribut artırmadan, çoxlu oxşarlıq atributu artıranda aldığınıza oxşar nəticə almağınızı mümkün edir. Gəlin SVC class-ını Qauss RTF kernelində yoxlayaq:

```
rbf_kernel_svm_clf = make_pipeline(StandardScaler(),
                                   SVC(kernel="rbf", gamma=5, C=0.001))
rbf_kernel_svm_clf.fit(X, y)
```

Bu model Fiqur 5-9-da sol aşağıda göstərilib. Digər qrafiklər qamma (γ) və C hiperparametrlərinin müxtəlif qiymətləri ilə təlim edilmiş modelləri göstərir. Qammanı artırmaq zınqırovşəkilli əyrini daraldır (Fiqur 5-8-də sol tərəfdəki qrafiklərə baxın). Nəticədə, hər bir nümunənin təsir diapazonu daha kiçikdir: qərar sərhədi daha nizamsız olur, fərdi nümunələrin ətrafında o tərəf bu tərəfə tərpənir. Əksinə, qammanın kiçik qiyməti zınqırovşəkilli əyrini genişləndirir: nümunələrin daha böyük təsir diapazonu olur, və qərarvermə sərhəddi daha hamar olur.
Yəni, γ tənzimləmə parametri kimi çıxış edir: əgər modeliniz overfit edirsə, γ-nı azaltmalısınız; underfit edirsə, artırmalısınız (C hiperparametrinə bənzərdir).



Fiqur 5-9. RTF kerneli istifadə edən DVM təsnifatçıları

Başqa kernellər də mövcuddur, lakin çox nadirən istifadə olunurlar. Bəzi kernellər spesifik data strukturları üçün ixtisaslaşmışdır. *Strinq kernelləri* bəzən mətn sənədlərini və ya DNT ardıcıllıqlarını təsnif edərkən istifadə olunur (məsələn, strinq alt ardıcıllıq kerneli və ya Levenşteyn məsafəsinə əsaslanan kernellər).

MƏSLƏHƏT

Seçə biləcəyimiz bu qədər kernel varkən, hansını istifadə etməyə necə qərar verəcəksiniz? Əsas qayda olaraq, həmişə birinci xətti kerneli yoxlamalısınız. LinearSVC class-ı SVC(kernel=”linear”) metodundan çox daha sürətlidir, xüsusilə də təlim çoxluğunun çox böyük olduğu vaxtlarda. Əgər çox böyük deyilsə, Qauss RTF-dən başlayaraq, kernelləşdirilmiş DVM-ləri yoxlaya bilərsiniz; çox vaxt həqiqətən yaxşı işləyir. Daha sonra, boş vaxtınız və hesablama gücünüz varsa, hiperparametr axtarışından istifadə edən digər kernelləri yoxlaya bilərsiniz. Əgər sizin təlim

çoxluğunuzun verilənlər strukturu üçün ixtisaslaşmış kernellər mövcuddursa, onları da yoxladınığınızdan əmin olun.

SVM(DVM) Class-ları və hesabi mürəkkəblik (computational complexity)

LinearSVC class-ı xətti DVM-lər (linear SVMs) üçün optimizasiya olunmuş alqoritm tətbiq edən liblinear kitabxanasına əsaslanır. Kernel hiyləsini (the kernel trick) dəstəkləmir lakin təlim nümunələri və atributların sayı ilə demək olar ki, xətti miqyaslayır. Onun təlim vaxt mürəkkəbliyi (time complexity) təxminən O(m × n)-dir. Əgər çox yüksək dəqiqlik tələb etsəniz, alqoritm daha çox vaxt alacaq. Bunu tolerans hiperparametri ϵ (Scikit-Learn-də adı “tol”-dur) kontrol edir. Bir çox təsnifat (classification) məsələlərinə standart tolerans uyğundur.

SVC class-ı kernel hiyləsini (the kernel trick) dəstəkləyən alqoritm tətbiq edən libsvm kitabxanasına əsaslanır. Təlimin vaxt mürəkkəbliyi (time complexity) , adətən, O(m × n) və O(m × n) arasındadır. Təəssüf ki, bu o deməkdir ki, təlim nümunələrinin sayı artdıqca (yüzlərə, minlərə çatdıqca), təlim dəhşətli dərəcədə yavaşlayır, buna görə də bu alqoritmə ən uyğunu kiçik yaxud ortaölçülü qeyri-xətti təlim çoxluqlarıdır. O, atributların sayı ilə çox yaxşı miqyaslanır, xüsusilə də dağınıq (sparse - hər bir nümunənin bir neçə 0-dan fərqli atributu olduğu zaman) atributlarla. Bu halda, alqoritm hər bir nümunə üçün orta 0-dan fərqli atribut sayına uyğun təxmini miqyas götürür.

SGDClassifier class-ı həmçinin standart olaraq geniş kənar payı təsnifatı (large margin classification) edir və hiperparametrləri - xüsusilə tənzimləmə parametrləri (alpha və penalty (cəza)) və learning_rate (öyrənmə dərəcəsi) - xətti DVM-lərə oxşar nəticələr almaq üçün uyğunlaşdırıla bilər. təlim inq üçün artıraraq öyrənməyə (incremental learning) icazə verən və az yaddaş istifadə edən stoxastik dərəcəli enmə (stochastic gradient descent) istifadə edir (Fəsil 4-ə baxın), beləcə siz ondan bir modeli RAM-a sığmayan böyük

verilənlər çoxluğu (dataset) təlim etmək üçün istifadə edə bilərsiniz (məsələn, out-of-core learning (core-dan kənar öyrənmə) üçün). Əlavə olaraq, onun hesabı mürəkkəbliyi O(m × n) olduğu üçün, çox yaxşı miqyaslanır. Cədvəl 5-1 Scikit-Learn-in DVM (SVM) təsnifat class-larını müqayisə edir.

Cədvəl 5-1. DVM təsnifatı üçün Scikit-Learn class-larının müqayisəsi

Class	Vaxt Mürəkkəbliyi	Out-of-core dəstəyi	Miqyaslamaya ehtiyac	Kernel Hiyləsi
Linear SVC	O(m × n)	Xeyr	Bəli	Xeyr
SVC	O(m² × n) to O(m³ × n)	Xeyr	Bəli	Bəli
SGDClassifier	O(m × n)	Bəli	Bəli	Xeyr

İndi isə, gəlin DVM (SVM) alqoritmlərinin xətti və qeyri-xətti reqressiya üçün necə istifadə oluna biləcəyinə baxaq.

DVM Reqressiya (SVM Regression)

DVM-lərdən təsnifat yerinə reqressiya üçün istifadə etməyin hiyləsi, obyektivi dəyişməkdir: iki sinif arasında kənar paylarını pozmadan ən geniş küçəni tapmaq yerinə, DVM reqressiya kənar paylarını pozmadan (küçədən kənar nümunələr) mümkün qədər çox nümunəni öyrənməyə çalışır. Küçənin genişliyi ϵ hiperparametri tərəfindən kontrol edilir. Fiqur 5-10 xətti verilənlər üzərində təlim edilmiş 2 xətti DVM (SVM) reqressiya modellərini göstərir, biri kiçik (ϵ = 0.5), digəri daha böyük (ϵ = 1.2) ilə.

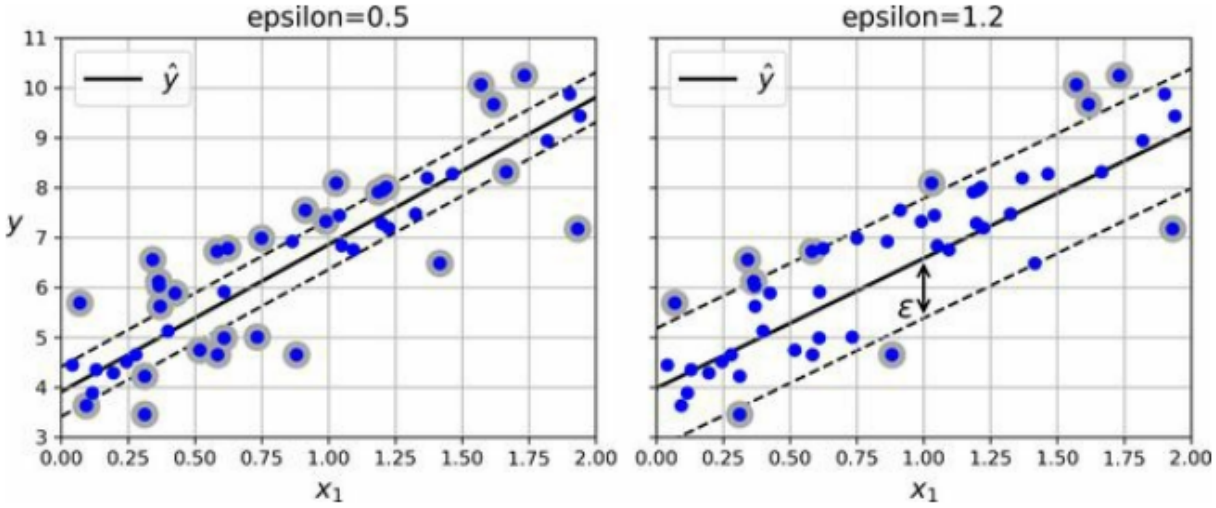


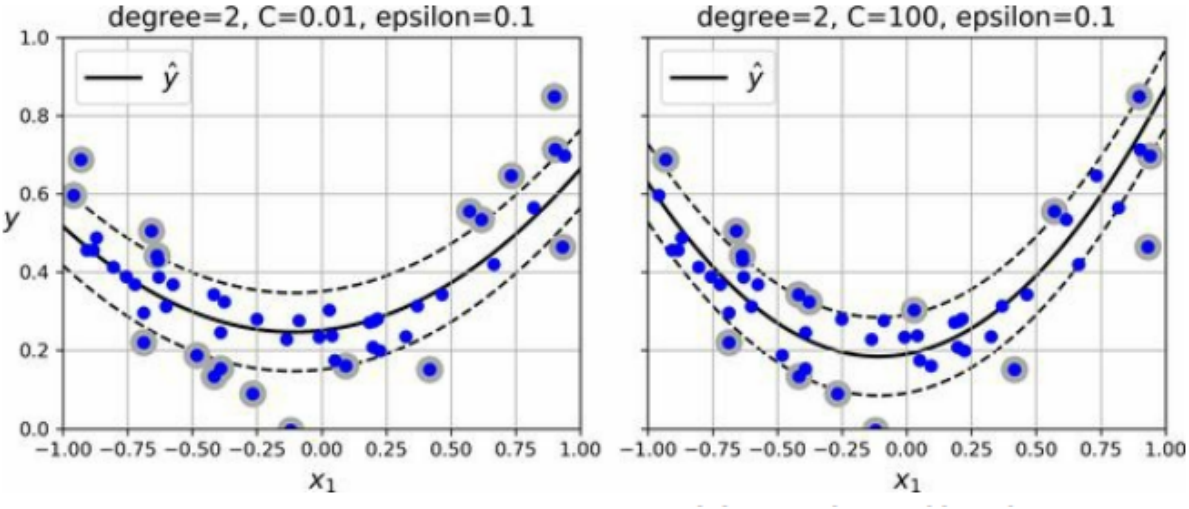
Figure 5-10. DVM (SVM) reqressiya

ϵ-ni azaltmaq dəstək vektorlarının sayını azaldır, onlar da modeli tənzimləyir. Bundan əlavə, əgər kənar payı (margin) daxilində daha çox təlim nümunəsi əlavə etsəniz, bu, modelin öngörmələrinə təsir etməyəcək; beləliklə, model ϵ-yə qarşı qeyri-həssasdır.

Xətti DVM (Linear SVM) reqressiyanı yerinə yetirmək üçün Scikit-Learn-in LinearSVR class-dan istifadə edə bilərsiniz. Aşağıdakı kod Fiqur 5-10-da göstərilən modeli yaradır:

```
from sklearn.svm import LinearSVR
X, y = [...] # a linear dataset
svm_reg = make_pipeline(StandardScaler(), LinearSVR(epsilon=0.5, random_state=42))
svm_reg.fit(X, y)
```

Qeyri-xətti reqressiya məsələlərini həll etmək üçün kernelizə olunmuş DVM (kernelized SVM) modelini istifadə edə bilərsiniz. Fiqur 5-11 ikinci dərəcəli polinom kernel istifadə edən, təsadüfi kvadratik təlim set üzərində təlim edilmiş DVM reqressiyanı göstərir. Sol tərəfdəki qrafikdə bir qədər tənzimləmə var (məsələn, kiçik C dəyəri), sağ qrafikdə isə demək olar yoxdur (çox böyük C dəyəri).



Fiqur 5-11. İkinci dərəcəli polinom kerneli istifadə edən DVM reqressiya

Aşağıdakı kod Fiqur 5-11-də sol tərəfəki qrafiki yaratmaq üçün Scikit-Learn-in SVR class-dan (bu class kernel hiyləsini dəstəyir) istifadə edir:

```
from sklearn.svm import SVR
X, y = [...] # a quadratic dataset
svm_poly_reg = make_pipeline(StandardScaler(),
                             SVR(kernel="poly", degree=2, C=0.01, epsilon=0.1))
svm_poly_reg.fit(X, y)
```

SVR class-ı SVC class-nın reqressiya ekvivalentidir, LinearSVR isə LinearSVC class-nın reqressiya ekvivalenti. LinearSVR class-ı təlim çoxluğunun ölçüsü ilə xətti miqyaslanır (eynilə LinearSVC class-ı kimi), SVR class-ı isə təlim çoxluğu çox böyük olduqda, həddən artıq yavaşkayır (eynilə SVC class-ı kimi).

QEYD

Fəsil 9-da görəcəyiniz kimi, DVM-lər həm də yeni nümünələrin (novelty) aşkarlanması üçün istifadə oluna bilər.

Bu fəsilin davamı xətti DVM (linear SVM) təsnifatçılarından başlayaraq, DVM-lərin öngörmələri necə etdiyini və onların təlim alqoritmlərinin necə işlədiyini izah edir. Əgər maşın öyrənməsinə yeni başlayırsınızsa, buranı təhlükəsiz bir şəkildə ötürə və birbaşa fəslin sonundakı tapşırıqlara keçə, DVM-ləri daha dərindən başa düşmək istədiyiniz zaman bu hissəyə geri dönə bilərsiniz.

Xətti DVM(SVM) təsnifatçıların daxilində

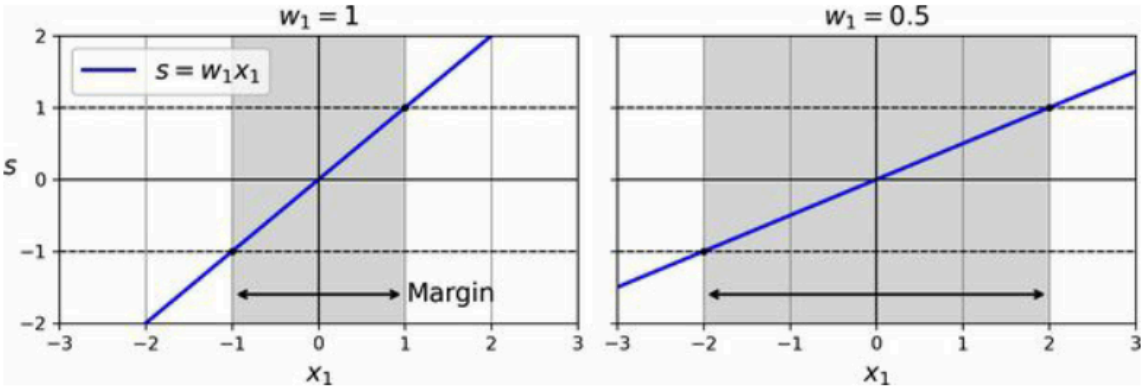
Xətti DVM(SVM) təsnifatçı əvvəlcə x_0-ın yanılma atributu (bias feature) olduğu (həmişə 1-ə bərabərdir) $\mathbf{\theta}^T x = \theta_0 x_0 + \cdots + \theta_n x_n$ qərar funksiyasını hesablayaraq yeni nümunənin sinfini ön görür. Əgər nəticə müsbətdirsə, onda öngörülmüş sinif $\hat{y}$ müsbət sinifdir (1); deyilsə, mənfi sinif (0). Bu eynilə Logistic Reqressiya kimidir. (Fəsil 4)

QEYD

İndiyə qədər, yanılma dəyəri (bias term) $\mathbf{\theta_0}$ və daxil olan atribut çəkiləri $\mathbf{\theta_1}$ və $\mathbf{\theta_n}$ daxil olmaqla bütün model parametrləri yeganə vektor **θ**-da toplamaq konvensiyasından istifadə etmişəm. Bu, bütün nümunələrə yanılma dəyəri $x_0 = 1$ əlavə edilməsini tələb edirdi. Digər çox yayılmış konvensiya isə yanılma dəyəri b-ni ($\mathbf{\theta_0}$-a bərabər olan) və atribut çəkilərinin vektoru w-nu ($\mathbf{\theta_1}$ və $\mathbf{\theta_n}$ daxil olan) ayırmaqdır. Bu halda, daxil olan atribut vektorlara yanılma atributunun əlavə edilməsinə ehtiyac yoxdur və xətti DVM(SVM)-in

qərar funkasiyası $\mathbf{w}\,\boldsymbol{x} + b = w\,x + \cdots + w\,x + b$-ə bərabərdir. Kitabın davamı boyunca bu konvensiyadan istifadə edəcəyəm.

Deməli, xətti DVM (SVM) istifadə edərək öngörmələr etmək kifayət qədər birbaşadır. Bəs təlim? Bunun üçün kənar payı pozuntularını (margin violations) limitləyərkən, "küçəni" yaxud kənar payını (margin) mümkün qədər geniş edən çəki vektoru w və yanılma həddi b -ni tapmalıyıq. Küçənin genişliyi ilə başlayaq : onu daha geniş etmək üçün w-nu daha balaca götürməliyik. Fiqur 5-12-dən göstərildiyi kimi, bunun 2-ölçülü vizualizasiyası daha rahat ola bilər. Gəlin, qərar funksiyasının -1 və ya +1 olduğu nöqtələrdə küçənin sərhədlərini müəyyən edək. Sol tərəfdəki qrafikdə çəki w_1 1-dir, deməli, w_1x_1 = -1 və ya +1 olduğu nöqtələr x_1 = -1 və +1-dir : buna görə də, kənar payı (margin) ölçüsü 2-dir. Sağ tərəfdəki qrafikdə çəki 0.5-dir, deməli w_1x_1 = -1 və ya +1 olduğu nöqtələr x_1 = -2 və +2dir: kənar payının (margin) ölçüsü 4-dür. Deməli, w-nu mümkün qədər balaca saxlamalıyıq. Qeyd edək ki, yanılma dəyərinin (bias term) kənar payının (margin) ölçüsünə heç bir təsiri yoxdur: onu dəyişdirmək ölçüsünü dəyişmədən kənar payını sadəcə ətrafda döndürür.



Fiqur 5-12. Daha kiçik çəki vektoru daha geniş kənar payı yaradır

Biz həmçinin kənar payı pozuntularından (margin violations) qaçınmaq istəyirik, buna görə də qərar funksiyası bütün müsbət təlim nümunələri üçün 1-dən böyük və bütün mənfi treyininq nümunələri üçün -1-dən kiçik olmalıdır. Əgər bütün mənfi nümunələr üçün ($y^{(i)}$ = 0 olduqda)

$t^{(i)} = -1$ və müsbət nümunələr üçün ($y^{(i)} = 1$ olduqda) $t^{(i)} = 1$ təyin etsək, bu məhdudiyyəti bütün nümunələr üçün $t^{(i)} (\mathbf{w}\, \mathbf{x}^{(i)} + b) \geq 1$ kimi yaza bilərik.

Beləcə, sərt kənar payı xətti DVM təsnifatçısının (hard margin linear SVM classifier) hədəfini Bərabərlik 5-1-dəki məhdudulaşdırılmış optimizasiya problemi kimi ifadə edə bilərik.

Bərabərlik 5-1. Sərt kənar payı xətti DVM təsnifatçısının hədəfi
minimize w,b 1 2 w ⊤ w subject to t (i) (w ⊤ x (i) + b) ≥ 1 for i = 1 , 2 , ⋯ , m

QEYD

Biz // **w** // -i (w-nun normunu) normallaşdırmaqdansa, ½ **w** **w** -nu minimallaşdırırıq, hansı ki
½ // **w** // ^{2} -ə bərabərdir. Həqiqətən də, ½ // **w** // ^{2} sadə, qəşəng törəməyə sahibdir (sadəcə **w**-a bərabərdir), // **w** // isə **w**=0 olduqda törəməsi yoxdur. Optimiziasiya alqorimtləri adətən differensiallaşan funksiyalarla çox daha yaxşı işləyir.

Yumşaq kənar payının (soft margin) hədəf funksiyasını almaq üçün, hər bir nümunə üçün slack dəyişənini ($\zeta^{(i)} \geq 0$) tanıtmalıyıq : $\zeta^{(i)}$ hər bir nümunəyə kənar payını pozmağa nə qədər icazə verildiyini göstərir. İndi bizim iki zidd hədəfimiz var: kənar payı pozuntularını (margin violations) azaltmaq üçün slack dəyərini mümkün qədər kiçik etmək və kənar payını artırmaq üçün ½ **w** **w** -ni mümkün qədər kiçik etmək. Burada C hiperparametri gəlir: o bizə bu iki hədəf arasındakı mübadiləni (trade-off) müəyyənləşdirməyə icazə verir. Bu bizə Bərabərlik 5-2-dəki məhdudlaşdırılmış optimizasiya problemini verir.

Bərabərlik 5-2. Yumşaq kənar payı xətti DVM təsnifatçısı hədəfi
minimize w,b,ζ 1 2 w ⊤ w + C ∑ i=1 m ζ (i) subject to t (i) (w ⊤ x (i) + b) ≥ 1 - ζ (i) and ζ (i) ≥ 0 for i = 1 , 2 , ⋯ , m

Sərt və yumşaq kənar payı problemləri hər ikisi xətti məhdudlaşdırıcıları olan konveks kvadratik optimizasiya problemləridir. Belə problemlər

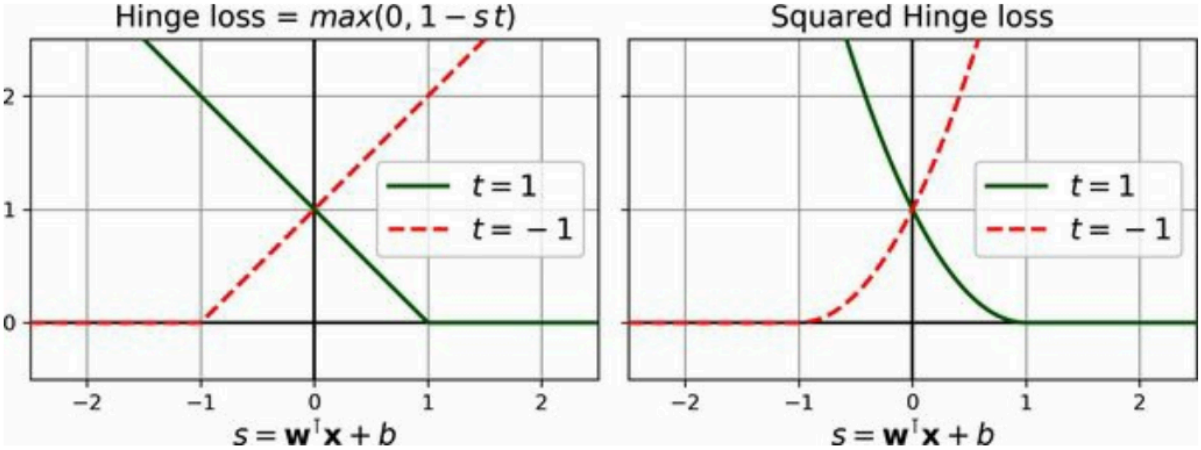
kvadratik proqramlaşdırma (KP) problemləri adlanır. Bir çox hazır həll edicilər KP problemlərini bu kitabın əhatəsinin xariivndə olan yollarla həll edə bilirlər. [4]

KP həlledici istifadə etmək DVM (SVM) təlim etməyin bir yoludur. Bir başqası isə hinge xərcini (loss) və ya kvadrat hinge xərcini (loss) minimallaşdırmaq üçün qradiyent azalma istifadə etməkdir (gradient descent) (Fiqur 5-13-ə bax). Müsbət sinfə aid olan (yəni, t=1) bir nümunə x üçün, əgər qərar funksiyasının çıxışı s (s=wx+b) 1-dən böyük və ya ona bərabərdirsə, xərc (loss) 0 olur. Bu, nümunənin küçədən kənarda və müsbət tərəfdə olduğu zaman baş verir.

Mənfi sinfə aid olan (yəni, t=−1) bir nümunə üçün isə, əgər s≤−1 olarsa, itki 0 olur. Bu, nümunənin küçədən kənarda və mənfi tərəfdə olduğu zaman baş verir.

Bir nümunə kənar payının düzgün tərəfindən nə qədər uzaqdırsa, itki bir o qədər artır: bu, hinge xərci üçün xətti şəkildə, kvadrat hinge xərci üçün isə kvadratik şəkildə böyüyür. Bu, kvadrat hinge xərcinin kənar dəyərlərə (outliers) qarşı daha həssas olmasına səbəb olur. Lakin, əgər verilənlər çoxluğu təmizdirsə, kvadrat hinge xərci daha sürətli yaxınlaşma (konvergensiya) təmin edir.

Standart olaraq, **LinearSVC** kvadrat hinge xəricndən istifadə edir, **SGDClassifier** isə hinge xərcindənistifadə edir. Hər iki sinif, xərc hiperparametrini "hinge" və ya "squared_hinge" olaraq təyin etməklə xərci seçməyə imkan verir. **SVC** sinfinin optimallaşdırma alqoritmi isə hinge xərcinin minimallaşdırılmasına bənzər bir həll tapır.



Fiqur 5-13. Hinge itkisi (solda) və kvadrat hinge itkisi (sağda)

Növbəti olaraq, xətti SVM təsnifatçısını təlim etməyin başqa bir üsuluna - dual problemi həll etməyə baxacağıq.

Dual Problemi

Məhdudlaşdırılmış optimallaşdırma problemi, yəni əsas problem (primal problem) veriləndə, onunla slx əlaqəli, lakin fərqli bir problem olan dual problem ifadə edilə bilər. Dual problemin həlli adətən əsas problemin həlli üçün aşağı sərhəd verir, lakin bəzi şərtlər altında hər iki problemin həlli eyni olur. Xoşbəxtlikdən, SVM problemi bu şərtlərə uyğundur, buna görə də əsas problemi və ya dual problemi həll etməyi seçə bilərsiniz; hər iki halda eyni həll alınacaq. 5-3 bərabərliy xətti SVM-in məqsəd funksiyasının dual formasını göstərir. Əgər dual problemi əsas problem üzərindən necə çıxarmaq sizə maraqlıdırsa, bu fəslin notebook-dakı əlavə material bölməsinə baxın.

Bərabərlik 5-3. Xətti DVM (linear SVM) obyektivinin dual problemi

minimize α 12∑i=1m

∑j=1mα(i)α(j)t(i)t(j)x(i)ᵀx(j) - ∑i=1mα(i)

subject to α(i)≥0 for all i=1,2,…,m and ∑i=1mα(i)t(i)=0

Bu bərabərliyi minimallaşdıran α^ vektorunu (QP həlli vasitəsilə) tapdıqdan sonra, əsas (primal) problemi minimallaşdıran b^ və w ^ dəyərlərini hesablamaq üçün 5-4 bərabərliyindən istifadə edin. Bu bərabərlikdə, n_s dəstəki vektorlarının (support vectors) sayını ifadə edir.

Bərabərlik 5-4. Dual həlldən primal həllə keçid

w ^ = ∑ i=1 m α ^ (i) t (i) x (i) b ^ = 1 n s ∑ i=1 α ^ (i) >0 m t (i) - w ^ ⊤ x (i)

Dual problemi həll etmək, təlim nümunələrinin sayı atributların sayından az olduqda, əsas problemi həll etməkdən daha sürətlidir. Daha vacibdir ki, dual problem kernel hiyləsini mümkün edir, halbuki əsas (primal) problem bunu etmir. Bəs kernel hiyləsi nədir?

Kernelizasiyalı DVM-lər (Kernelized SVMs)

Tutaq ki, iki ölçülü təlim çoxluğuna (məsələn, moons təlim çoxluğu) ikinci dərəcəli polinomial bir transformasiya tətbiq etmək, sonra isə transformasiya olunmuş təlim dəstində xətti bir DVM təsnifatçısı öyrətmək istəyirsiniz. 5-5 bərabərliyi, tətbiq etmək istədiyiniz ikinci dərəcəli polinomial proyeksiya funksiyası ϕ-ni göstərir.

Bərabərlik 5-5: İkinci dərəcəli polinomial proyeksiya

ϕ x = ϕ x 1 x 2 = x 1 2 2 x 1 x 2 x 2 2

Fikir verin ki, transformasiya olunmuş vektor artıq 2 ölçülü deyil, 3 ölçülüdür. İndi isə, iki 2 ölçülü vektorlara (a və b) bu ikinci dərəcəli polinomial proyeksiyanı tətbiq etdikdə və transformasiya olunmuş vektorların skalyar hasilini hesabladıqda nə baş verdiyinə baxaq (bax: Bərabərlik 5-6).

Bərabərlik 5-6. İkinci dərəcəli polinomial proyeksiyası üçün kernel hiyləsi

$$\phi(a)^T \phi(b) = a_1^2 2a_1 a_2 a_2^2 {}^T b_1^2 2 b_1 b_2 b_2^2 = a_1^2 b_1^2 + 2 a_1$$

$$b_1 a_2 b_2 + a_2^2 b_2^2 = a_1 b_1 + a_2 b_2^2 = a_1 a_2 {}^T b_1 b_2^2 = (a^T b)^2$$

Nə baş verir? Transformasiya olunmuş vektorların skalyar hasili, orijinal vektorların skalyar hasilinin kvadratına bərabərdir: $\phi(\mathbf{a})\,\phi(\mathbf{b}) = (\mathbf{a}^T\,\mathbf{b})$.

Əsas fikir buradadır: əgər bütün təlim nümunələrinə ϕ transformasiyasını tətbiq etsəniz, dual problemdə (bax: Bərabərlik 5-3) $\phi(x)\,\phi(x)$ skalyar hasil olacaq. Lakin, əgər ϕ Bərabərlik 5-5-də müəyyən edilən ikinci dərəcəli polinomial transformasiya olarsa, bu skalyar hasili sadəcə

$(x(i)^T x(j))^2$ ilə əvəz edə bilərsiniz. Yəni, təlim nümunələrini ümumiyyətlə transformasiya etməyə ehtiyac yoxdur; sadəcə Bərabərlik 5-3-də skalyar hasilii onun kvadratı ilə əvəz edin. Nəticə təlim çoxluğunu transformasiya edib sonra xətti DVM (SVM)alqoritmini tətbiq etməklə əldə ediləcək nəticə ilə tamamilə eyni olacaq, lakin bu yanaşma prosesi hesablamalar baxımından xeyli daha səmərəli edir.

Funksiya $K(a,b)=(a^Tb)2= (a^T b)^2$ ikinci dərəcəli polinomial kerneldir. Maşın öyrənməsində kernel, $\phi(a)\,\phi(b)$ skalyar hasilini transformasiyanı (ϕ) hesablamağa (hətta onu bilməyə) ehtiyac olmadan, yalnız orijinal vektorlara (a və b) əsaslanaraq hesablamağa qadir olan bir funksiyadır.

Bərabərlik 5-7. Çox istifadə edilən kernellər

Xətti: $K(a, b) = a^T b$ Polinomial: $K(a, b) = \gamma a^T b + r^d$ Gaussian RBF:

K (a , b) = exp (- γ a-b 2) Sigmoid: K (a , b) = tanh γ a ^{T} b + r

MERCER Teoremi

Mercer teoreminə görə, əgər bir funksiya K(a,b) bir neçə riyazi şərtə, yəni Mercer şərtlərinə uyğun gəlirsə (məsələn, K davamedici və simmetrik olmalıdır, yəni K(a,b)=K(b,a) olmalıdır və s.), onda elə bir ϕ funksiyası mövcuddur ki, o a və b-ni başqa bir fəzaya köçürür (daha çox ölçülü bir fəza olması mümkündür), K(a,b)=ϕ(a)·ϕ(b). Siz ϕ-nin nə olduğunu bilməsəniz belə, onun mövcud olduğunu bildiyiniz üçün K-nı kernel kimi istifadə edə bilərsiniz. Məsələn, Gaussian RBF kernelində göstərilə bilər ki, ϕ hər bir təlim nümunəsini sonsuz ölçülü bir fəzaya köçürür, buna görə də bu köçürməni həqiqətən yerinə yetirməyə ehtiyac olmaması böyük üstünlükdür!

Qeyd etmək lazımdır ki, bəzi geniş istifadə olunan kernellər (məsələn, sigmoid kerneli) Mercer şərtlərinin hamısına əməl etməsələr də, praktiki olaraq yaxşı nəticələr verir.

Hələ də tamamlanması lazım olan bir məqam var. Bərabərlik 5-4 xətti DVM təsnifatçısı üçün dual həldən əsas (primal) həllə keçidin necə baş verdiyini göstərir. Lakin siz kernel hiyləsini tətbiq etsəniz, nəticədə ϕ($x^{(i)}$) -nin içində olduğu bərabərliklər alacaqsınız. Həqiqətən də, Bərabərlik 5-4dəki w^ ϕ($x^{(i)}$) ilə eyni ölçüyə sahib olmalıdır, bu da çox böyük və ya sonsuz ola bilər, beləcə onu hesablaya bilməzsiniz. Bəs w^-nu bilmədən necə öngörmələr edəcəksiniz? Yaxşı xəbər budur ki, w^-nin Bərabərlik 5-4-dəki düsturunu yeni bir nümunə $x^{(n)}$ üçün qərar funksiyasına daxil edə bilərsiniz və nəticədə yalnız giriş vektorları arasındakı skalyar hasilləri ehtiva edən bir bərabərlik əldə edirsiniz. Bu, kernel hiyləsindəndən istifadəni mümkün edir (Bərabərlik 5-8).

Bərabərlik 5-8. Kernelizə edilmiş DVM ilə öngörmələr etmək

h w ^,b ^ ϕ (x (n)) = w ^ ⊤ ϕ (x (n)) + b ^ = ∑ i=1 m α ^ (i) t (i) ϕ(x (i)) ⊤ ϕ (x (n)) + b ^ = ∑ i=1 m α ^ (i) t (i) ϕ (x (i)) ⊤ ϕ (x (n)) + b ^ = ∑ i=1 α ^ (i) >0 m α ^ (i) t (i) K (x (i) , x (n)) + b ^

Qeyd edək ki, sadəcə dəstək vektorları üçün $\alpha^{(i)} \neq 0$ olduğundan, öngörmələr edərkən yeni giriş vektoru $x^{(n)}$-in skalyar hasili yalnız dəstəkləyici vektorlarla hesablanır, bütün təlim nümunələri ilə yox. Əlbəttə ki, bias termini b^-ni hesablamaq üçün də eyni hiylədən istifadə etməlisiniz.

Bərabərlik 5-9. Bias termi hesablamaq üçün kernel hiyləsi istifadə etmək

b ^ = 1 n s ∑ i=1 α ^ (i) >0 m t (i) - w ^ ⊤ ϕ (x (i)) = 1 n s ∑ i=1 α ^ (i) >0 m t (i) - ∑ j=1 m α ^ (j) t (j) ϕ(x (j)) ⊤ ϕ (x (i)) = 1 n s ∑ i=1 α ^ (i) >0 m t (i) - ∑ j=1 α ^ (j) >0 m α ^ (j) t (j) K (x (i) , x (j))

Əgər başınız ağrımağa başlayıbsa, bu tamamilə normaldır: bu, kernel trick-in təəssüf doğuran əks təsirlərindən biridir.

Qeyd

Online kernelizasiyalı DVM-ləri, yəni tədricən öyrənmə qabiliyyətinə malik olan kernelizasiyalı DVM-ləri də həyata keçirmək mümkündür. Bu, “Incremental and Decremental Support Vector Machine Learning” və “Fast Kernel Classifiers with Online and Active Learning” məqalələrində təsvir edilmişdir. Bu kernelizasiyalı DVM-lər Matlab və C++-da reallaşdırılmışdır. Lakin böyük miqyaslı xətti olmayan problemlər üçün random forest-lərdən

(bax: Fəsil 7) və ya neyron şəbəkələrdən (bax: II Hissə) istifadə etməyi nəzərə ala bilərsiniz.

Çalışmalar

1. Dəstək vektor maşınlarının əsas ideyası nədir?
2. Dəstək vektoru nədir?
3. Niyə DVM-lərdən istifadə edərkən girişləri miqyaslamaq vacibdir?
4. DVM təsnifatçısı nümunəni təsnif edərkən əminlik balı (confidence score) verə bilər? Bəs ehtimal?
5. LinearSVC, SVC, və SGDClassifier arasında necə seçim edə bilərsiniz?
6. Deyək ki, RTF (RBF) kernelli bir DVM(SVM) təsnifatçısı təlim etmisiniz, amma təlim çoxluğunu çox az öyrənib (underfit). Qammanı (γ) azaltmalısınız, ya artırmalı? Bəs C?
7. Modelin ϵ-həssas olmaması nə deməkdir?
8. Kernel hiyləsi istifadə etməyin məqsədi nədir?
9. Xətti ayrılabilən bir verilənlər çoxluğunda LinearSVC təlim edin. Daha sonra, eyni verilənlər çoxluğunda SVC və SGDClassifier-i təlim edin. Yoxlayın görün, çox oxşar modellər ala bilirsiniz?
10. Wine verilənlər dəstində DVM təsnifatçısınıı təlim edin. sklearn.datasets.load_wine() funksiyasından istifadə edərək Wine verilənlər çoxluğunu yükləyə bilərsiniz. Bu verilənlər çoxluğu 3 müxtəlif üzüm çeşidindən istehsal edilən 178 şərab nümunəsinin

kimyəvi analizlərini ehtiva edir. Məqsəd, şərabın kimyəvi analizlərinə əsaslanaraq üzüm çeşidini öngörə bilən təsnifat modeli təlim etməkdir. Çünki DVM təsnifatçılar iki sinifli təsnifatçılardır, üç sinifi təsnif etmək üçün "birə-qarşı-hamı" yanaşmasından istifadə etməlisiniz. Hansı dəqiqliyə çata bilərsiniz?

11. California housing verilənlər çoxluğunda DVM regressoru təlim edin və parametrlərini tənzimləyin (fine-tune). İkinci fəsildə istifadə etdiyimiz dəyişiklik edilmiş verilənlər çoxluğu yerinə, sklearn.datasets.fetch_california_housing() ilə orjinal versiyanı istifadə edə bilərsiniz. Hədəflər yüzminlərlə dolları təmsil edir. 20,000-dən çox nümunə olduğundan, DVM yavaş ola bilər, buna görə də hiperparametrləri tənzimləmək üçün çox az sayda nümunə istifadə edib (məsələn, 2000) çoxlu hiperparametr kombinasiyaları yoxlamalısınız. Modelinizin ən yaxşı RMSE nəticəsi nədir?

Bu çalışmaların həlləri fəslin notebook-unun sonundadır: https://homl.info/colab3 .