



MACHINE
LEARNING
with
MATLAB

MACHINE
LEARNING
with
MATLAB

14.11.2023

Doğukan Mehmet
KUNG

Bit table dosyasını aşağıdaki gibi tanımlarız-

① `T = readtable("datafile.txt")`

Tabloların her sütunu tabloda bir değişken olarak adlandırılır ve nokta gösterimini kullanarak erişilebilir.

playerinfo.pos
playerinfo.height

Metin etiketlerinin bir oyuncunun pozisyonu gibi sonlu bir olasılık kümesini temsil etmesi amaçlan-
dığında, verileri kategorik bir dizi olarak saklamak daha uygundur.

Bir metin etiketleri dizisini kategorik bir dizeye dönüştürmek için "categorical" kullanılır.

② `labels = categorical(labels)`

↳ Kategorik bir dizedeki tüm olası kategorilerin bir nesnesini döndürür.

Kategorileri belirtmek için "categorical" fonksiyonuna ikinci bir girdi olarak kategori adlarından oluşan bir dizi kullanabiliriz.

③ `data = categorical(data, catNames)`

Konumları bir kategori tarafından belirtilmeyen tüm veriler `<undefined>` olur.

Bir tablodan tanımlanmamış veya eksik değer iteren satırları kaldırmak için "missing" fonksiyonu kullan.

④ `T = rmissing(T)`

"groupsummary" fonksiyonu gruplandırılmış hesaplamaları gerçekleştirir. Örneğin, aşağıdaki kuant data Label ile gruplandırılmış data içindeki verilerin standart sapmasını (std) hesaplar.

⑤ `stdDevData = groupsummary(data, "Label", "std")`

Bir tablodaki değişken adlarını tablonun "properties" özelliğinin "VariableNames" özelliği kullanarak erişebiliriz.

⑥ `table.Properties.VariableNames`

Bir değişkeni boş diziye ([]) atayarak tablodan kaldırabileceğimizi unutmayın.

⑦ `table.variable = []`

Örneğin playerStats'taki GroupCount'u kaldırmak için `playerStats.GroupCount = [];`

Aynı anahatlar değişken değerlerine sahip satırları eşleştirerek iki tabloyu birleştirebiliriz. "innerjoin" işlevi iki tabloyu birleştirir ve yalnızca anahat değişken değerleri her iki tabloda da görünen gözlemleri içerir.

⑧ `JoinedData = innerjoin(tableA, tableB)`

La tableA ve tableB'deki tüm değişkenleri içeren bir tablodur.

Çeşitli istatistikler ve oyuncu pozisyonu arasındaki ilişkileri keşfetmek için basketbol oyuncusu verilerinden bazı sonuçları çıkarabilirsiniz. Örneğin boxplot fonksiyonu her pozisyon için atışın sayısının bir kutu grafiğini oluşturabilir.

⑨ `boxplot (data.points, data.pos)`
`(shot.height, data.pos)`

Pozisyona göre gruplandırılmış iki değişken arasındaki ilişkiyi keşfetmek için "gscatter" kullanılabilir.

⑩ `gscatter (data.varX, data.varY, data.pos)`
`(data.rebounds, data.points, data.pos)`

Bazen gscatter tablosu çok karmaşık veya dağınık olabilir. Bir bağımsız değişkene bağlı gözetil değerleri bölmeye uygundur. Yine örneğin, sayı ve reboundları oynanan maç sayısına bölmektir.

⑪ `data.variable ./ data.GP`

→ Oyun süresine göre de data.kaynakları bölünebilir

Yani `gscatter (data.varX ./ data.GP, data.varY ./ data.GP, data.pos)`
 Bu kod ile
 veri noktalarının (x,y) orijini etrafında kümelenmesi önüne geçilebilir.

MODEL EĞİTME VE TAHMİNDE BULUNMA

Bir KNN (k-en yakın komşu) modeli, bir gözetim bilinen en yakın örneklerle aynı sınıf olarak sınıflandırır. "fitknn" fonksiyonuna bir veri tablosu ileterek bir KNN modeli kurabilirsiniz.

⑫ `mdl = fitknn (table, "responseVariable")`

↑
uygun modeli
kayd eden
değişken.

→ Tablodaki yanıt değişkeni adıdır.
Yani modelin tahmin etmesini istediğiniz değişken.

Tahmin edilenleri yeni gözlemlerin tahmin edilen sınıfını belirler.

(13) `predClass = predict(model, newdata)`

Her gözlem için tahmin edilen sınıfın kategorik dizisi.

Eğitilmiş model

Modeli eğitmek için kullanılan tahmin edici değişkenlere aynı olan bir gözlem tablosu

Bir modeli değerlendirmek için yaygın olarak kullanılan bir metriğe ya da sınıflarına oranı (ya da tahminlerin oranı) 'dır. Bu metriğe model kaybı (model loss) 'ta derir.

Bir veri seti için yukarı sınıflarına oranını hesaplamak üzere kayıp fonksiyonunu kullanabiliriz.

(14) `L = loss(model, table)`

L bu değerin küçük olması beklenir.

Koraklık matrisi (confusion matrix), her bir sınıftan her bir sınıf olduğu tahmin edilen gözlemlerin sayısını verir.

"`confusionchart`" fonksiyonunu kullanarak bir koraklık matrisi görselleştirebiliriz.

(15) `confusionchart(ytrue, ypred)`

y-true tahminlerin gerçek değerler. (bilinen sınıfların bir vektörü)

y-true tahmin edilen sınıfların bir vektörüdür

ÖRNEĞİN

`confusionchart(dataTest_pos, predClass)`

Test verisiyle ilgili bağımlı değişken (bilinen vektör)

Tahmin edilen vektörler (predict ile)

Varsayalım ki, fitcknn $k=1$ olan bir kNN modeline uyuyor. Model yeni bir gözlem sınıflandırmak için en yakın tek "komsunun" sınıfını kullanır.

Model performansı, k -değeri artırılırsa, yani sadece bir sınıf yerine bir kaç komsunun en yakın sınıfını kullanarak artabilir.

fitcknn çağırılırken "NumNeighbors" özelliği ayarlayarak k değerini değiştirebiliriz.

(16) `mdl = fitcknn (table, "Response Variable", ...
"NumNeighbors", 7)`

Bir çok makine öğrenmesi modeli benzer ölçütler olarak gözlemler arası mesafeyi kullanır. Daha küçük mesafeler daha benzer mesafeleri gösterir.

Benzer özellikleri olan veri içitlerinde bu durum sıkıntı yaratabilir. Bunu aşmak için de standartlaştırma kullanılabilir.

"fitcknn" fonksiyonunda "Standardize" özelliği "true" olarak ayarlandığında, tahmin edilecek verilerin her bir sütunu ortalaması 0 ve standart sapma 1 olarak şekilde normalleştirilir. Ardından model standartlaştırılmış veriler kullanılarak eğitilir.

(17) `mdl = fitcknn fitcknn (table, "Response Variable", ...
"NumNeighbors", 7, ...
"Standardize", true)`

Çok fazla değişken içeren verileri etkili bir şekilde görselleştirmek için çok boyutlu ölçeklendirme ve Temel Bileşen Analizi (PCA) gibi boyut azaltma tekniklerini kullanabiliriz.

ÇOK BOYUTLU ÖLÇEKLENDİRME

Adım 1: İkili mesafeleri hesaplayın. Bunun için "pdist" fonksiyonunu kullanabiliriz. Girdi sayısı bir matristir.

$$(18) \quad D = \text{pdist}(\text{data}, \text{"distance"})$$

D: Her gözlem çifti arasındaki mesafeyi içeren bir mesafe veya gözlem çifti.

D'nin boyutu $n(n-1)/2$ 'dir.

data: Verileri içeren m 'ye n sayısal matris. m -satırının her biri bir gözlem olarak kabul edilir.

"distance": Mesafeyi veya farklılığı hesaplama yöntemini gösteren isteğe bağlı bir girdi. Yaygın olarak kullanılan yöntemler "euclidean" (varsayılan öklid), "cityblock" ve "correlation".

Adım 2: Çok boyutlu ölçeklendirmeyi gerçekleştirin.

Farklılık (dissimilarity) vektörünü "cudscale" fonksiyonuna girdi olarak kullanabiliriz.

$$(19) \quad [x, e] = \text{cudscale}(D)$$

x: q boyutlu yaya yeniden oluşturulan koordinatların $(m \times q)$ matrisi. q , verilen ikili mesafelere ulaşmak için gereken minimum boyut sayısıdır.

e: $(x^T x)$ matrisinin özdeğerleri

D: uzaklık ve farklılık vektörü

Sütun sayısı referans alınarak en iyi boyut kullanarak görselleştirme sağlanabilir. Çok boyutlu ölçeklendirme yapmak için öncelikle gözlemler arasındaki ikili uzaklıkları hesaplamamız gerekir. Bu uzaklıkları hesaplamak için "pdist" kullanılır.

(20) `distances = pdist(data);`

"cudscale" işlevi, bir dizi ikili mesafe için bir yapılandırma matrisi ve buna karşılık gelen ödeğerleri bulur.

(21) `[configMat, eigval] = cudscale(distances);`
 Yapılandırma matrisi Ödeğerler.

Bir vektörün göreceli büyüklüklerini azalan sırada görselleştirebilir bir Pareto grafiği oluşturmak için "pareto" fonksiyonu kullanırız.

(22) `pareto(vector)`
 ↳ eigval

Pareto grafiğinde, dağılımın %90'ından fazlasının sadece iki değereyle tanımlandığını görebiliriz. Bir M matrisinin ilk iki sütununun dağılım grafiğini oluşturmak için "scatter" fonksiyonunu kullanabiliriz.

(23) `scatter(M(:,1), M(:,2))`
 ↳ configMat

Scatter grafikler dağılımın kaç boyuta dağıldığını gösterirler. Buna göre boyut analizi yapılır. Kümelennmiş veriler görür, buradan yola çıkarak küme sayısına göre kaç boyutlu olduğunu belirtiriz.

Ça boyut için "scatter3" fonksiyonu, 3 boyutlu bir dağılım grafiği oluşturur. Bir M matrisinin 3. sütununun dağılım grafiğini oluşturmak için scatter3'ü kullanabiliriz.

(24) `scatter3(M(:,1), M(:,2), M(:,3))`

"mdscale" fonksiyonunun kullandığımızda, yapılandırma matrisinde kaç boyut döndürüleceğini belirtiriz.

Belirtilen sayıda boyuta sahip bir yapılandırma matrisi bulmak için "mdscale" fonksiyonu kullanabiliriz.

(25) `configmat = mdscale(distances, numDim, numDim);`

pdist'in çıktı

Ardından "scatter" ile etrafa basarız.

PCA amacı:

... büyük bir veriyi, verideki temel özellikleri yakalayarak daha az sayıda değişkelle göstermektir.



FIGES

TEMEL BİLEŞEN ANALİZİ PRINCIPAL COMPONENT ANALYSIS (PCA)

How görselleri
girdi ölçütü /20.....
gür-

Boyutsallığın azaltılması için de yaygın olarak kullanılan bir diğer yöntem temel bileşen analizidir. (PCA). Temel Bileşen Analizini gerçekleştirmek için "pca" fonksiyonunu kullanırız.

(26) $[pcs, sers, N, N, pexp] = pca(data)$

PAZETO GRAFİĞİ
BURADAN ELDE
EDİLİR

pcs: Temel bileşenlerin n'ye n matrisi

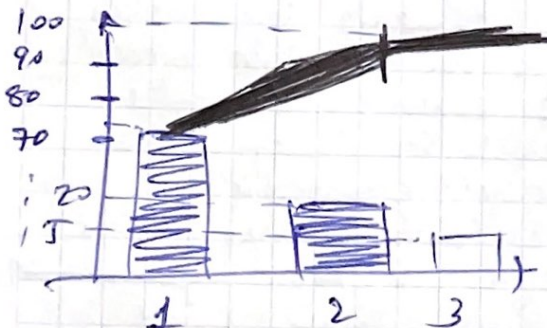
ser: Doğrusal koordinat dönüşüm matrisi adettleri (ilk çıktı) kullanılarak dönüştürülen verileri içeren bir $(m \times n)$ matris.

pexp: Her bir temel bileşen tarafından açıklanan varyans yüzdesini içeren n uzunluğunda bir vektör

data: Bir $(m \times n)$ sayısal matris. N sütun, n gözlemlenen değişkelle karşılık gelir. M satırın her biri bir gözleme karşılık gelir.

Ham veriler üzerinde Temel Bileşen Analizi (PCA) gerçekleştirmek için "pca" fonksiyonunu kullanabiliriz.

(27) $[coeffs, score, N, N, expl] = pca(data)$



Bu pareto grafiğinde der ki;
1. temel bileşenle %70
1.ve 2. temel bileşenle %100
1,2,3. temel bileşenle %100
açıklama yapılabilir.

K-Means Clustering

Bu verilerde veri seti içerisindeki veriler ayrılmalıdır. Bu oluşturulan grup sayısına karar verdiğimiz sayıya gözlemleri grup veya kümelere ayırmak için k-ortaklı küme yöntemini kullanabiliriz.

K-Means kümelemesini gerçekleştirmek için "kmeans" işlevini kullanır.

(28) $idx = kmeans(X, k)$

idx : Sayısal sütun vektörü olarak döndürülen küme endeksleri

X : Sayısal matris olarak belirtilen veriler.

k : Küme sayısı

SEÇENEKLER

- Uzaklık Metriği: Varsayılan olarak iki gözlem arasındaki büteliğe erişmek için öklit mesafesi kullanılır. Korelasyon gibi diğer ölçümleri de kullanabiliriz.

(29) $g = kmeans(X, 2, "distance", "correlation")$

Optimum kümelemeyi elde etmenin bir başka yolu, analizi farklı başlangıç merkezleriyle birden çok kez gerçekleştirmek ve ardından merkezler ile gözlemler arasındaki mesafelerin toplamını en aza indiren kümelere sınımaı seçmektir.

Bu "Replicates" özelliği kullanılarak yapılabilir. Aşağıdaki komut kümelemeyi 5 kez ~~tekrarlar~~ tekrarlar ve en düşük toplamı sahip kümeleleri döndürür.

(30) $g = kmeans(X, 2, "Replicates", 5);$

Verileri k gruba veya kümeye ayırmak için `kmeans` fonksiyonunu kullanabilirsiniz.

$$(31) \quad \text{groupNum} = \text{kmeans}(\text{data}, k)$$

Verileri bir dağılım grafiğinde görüntüleyerek ve her gruba farklı bir renk atayarak grupları görselleştirebilirsiniz.

Grupların vektörünü `gscatter` fonksiyonuna ikinci bir girdi argümanı olarak sağlayın.

$$(32) \quad \text{gscatter}(X(:,1), X(:,2), \text{groupNum})$$

"`randata`" komut dosyasını birkaç kez çalıştırırsanız, kümelerin her çalıştırmada aynı olmadığını görebilirsiniz.

GAUSS KARİŞİM MODELLERİ

Kullanabileceğimiz başka bir kümeleme yöntemi Gauss Karışım Modelidir (Gaussian Mixture Models (GMM)). GMM kümelemesi, verilere bir kaç n -boyutlu normal dağılım uydurmayı ve bu dağılımları kullanarak her gözlemi bir kümeye atanayı içerir.

GMM KÜMELEMESİ (GMM Clustering)

Adım-1: Gauss Karışım Modelini Uygulayın.

Çeşitli çok boyutlu Gauss (normal) dağılımlara uyum sağlamak için "`fitgmdist`" fonksiyonunu kullanabiliriz. Aşağıdaki komut iki dağılımın gmm karışımına uyur-

$$(33) \quad \text{gm} = \text{fitgmdist}(X, 2)$$

Adım-2 : Kümeleri Tanımlayın

Artık veriler, her bir bileşen için her bir gözlemin sonsal olasılığını hesaplayarak olasılıksal olarak kümelenebilir.

(34) $g = \text{cluster}(gm, x);$

Kümele belirlemek için kullanılan bireysel olasılıkları da döndürebilirsiniz. "P" matrisinin her iki küme için bir tane olmak üzere iki sütunu vardır.

(35) $[g, n, p] = \text{cluster}(gm, x);$

"k" dağılımlarını kullanarak verilere bir Gauss karışım modeli uydurmak için "fitgmdist" fonksiyonu kullanabiliriz.

(36) $gmModel = \text{fitgmdist}(data, k);$

Küme izleri, "gmModel" kullanarak verilerdeki gözlemleri bölümlere ayırır.

(37) $grps = \text{cluster}(gmModel, data);$

Verileri bir dağılım grafiğinde görüntüleyerek ne ne gruba farklı bir renk atayarak grupları görselleştirebiliriz.

(38) $gscatter(x(:,1), x(:,2), grpNum)$

Grafikleri tek tek görselleştirip birbiri üzerine de ekleyebiliriz. Fakat tek tek yapmak yerine aşağıdaki kodu yazabiliriz.

(39) `parallelcoords(X, "Group", g)`

X: Sayısal matris olarak verilen veriler.

"Group": Property Name

g: Gözlemlerin grup veya küme tanımlayıcılarını içeren bir vektör.

kmeans fonksiyonu, her grubun merkezlerini içeren isteye bağlı ikinci bir çıktı döndürebilir.

(40) `[grpNum, centroids] = kmeans(data, k)`

Sonuçları görselleştirmek için "parallelcoords" fonksiyonunu kullanabiliriz.

(41) `parallelcoords(data, ---
"Property Name", Property Value)`

Born veri setlerinde, gözlemlerin kendileri ile ilişkilendirilmiş bir kategorisi zaten vardır. Kümelerin orijinal kategoriler arasındaki dağılımını incelemek için "crosstab" fonksiyonu kullanılabilir.

(42) `counts = crosstab(yCat, grp)`

"crosstab" a ilk girdi sayım satırına, ikinci girdi ise sayım sütununa karşılık gelir.

Sayıları görselleştirmek için ısı haritası ilelerini kullanabiliriz.

(43) `heatmap(counts)`

"YDisplayLabels" cırtılığını kullanarak ısı haritasındaki original kategorileri etiketleyebiliriz-

(44) heatmap (counts, ---
"YDisplayLabels", categories(cats))

Her bir gözlemin aynı kümedeki diğer gözlemlere ne kadar yakın olduğunu görmek için "silhouette" fonksiyonu kullanılır.

(45) silhouette (data, groups)

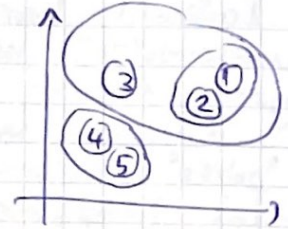
Kaç grup kullanacağımızdan emin değiliz "evalclusters" ile en iyi sonucu veren küme sayısını belirlemek için kmeans ile silhouette değeri kullanılır.

(46) eva = evalclusters (data, ---
"kmeans", "silhouette", ---
"klist", vectorOfPositiveIntegers)
L (örneğin 2:5)

parallelcoords : Her gözlemin değişken numarasına göre bir çizgi olarak çizer.

Hiyerarşik KİMELEME

Hiyerarşik Kimelemeyle data büyük kümeler oluşturarak büyük bu kümelerin alt kümelerini keşfedebiliriz.



Bir dizi gözlemleri bir Hiyerarşik Kimeleme Ağacı kodlamak için "linkage" fonksiyonu kullanabiliriz.

(47) $z = \text{linkage}(\text{data})$

İkili küme ağacının hiyerarşisini görselleştirmek için "dendrogram" fonksiyonunu kullanabiliriz.

(48) $\text{dendrogram}(\text{linkages})$
LP "12"

İstediği bağıllı ikinci bir girdi girerek "linkage"nin kümeler arasındaki mesafeyi hesaplama şeklini değiştirebiliriz. Örneğin "ward" yöntemi Ward'ın minimum varyans algoritmasını kullanarak 14 kore mesafesini hesaplar.

~~Görüntüleri~~

(49) $z = \text{linkage}(\text{data}, \text{"ward"})$
 $\text{dendrogram}(z)$

Gözlemleri z bağlantı uzaklıklarına göre k gruptan birine atamak için "cluster" fonksiyonunu kullanabiliriz.

(50) $\text{grp} = \text{cluster}(z, \text{"maxclust"}, k)$

(51) $\text{parallecoords}(X, \text{"Group"}, \text{grp})$

Koheretik korelasyon katsayısı, ağacın gözlemler arasındaki mesafeleri ne kadar doğru temsil ettiğini ölçer. 1'e yakın değerler yüksek kaliteli bir çözüme işaret eder. Original verilerin ikili mesafelerini hesaplamak için "pdist" kullanılır. Ardından Cophenetic korelasyon katsayısını hesaplamak için "cophenet" ileleri kullanılabilir.

(52)

$$ccc = \text{cophenet}(\text{linkages}, \text{distances})$$

Tipik MAKİNE ÖĞRENİMİ İŞ AKIŞI

Prepare Data
Choose Algorithm
Fit Model
Evaluate Model
Update Model
Make Predictions

cvpartition
fitc —
loss
fitc —
predict

Tipik makine öğrenimi iş akışını ve her adımda kullanılan en yaygın MATLAB işlevlerini gördünüz.

Bir Modeli Uydurma

Bir sınıflandırıcıyı sığdırmak için fitc — (sınıflandırıcının adına — yerine) fonksiyonunu kullanabilirsiniz. Örneğin aşağıdaki sınıflandırıcı k-en yakın komşuluk (kNN) sınıflandırma modeli oluşturur.

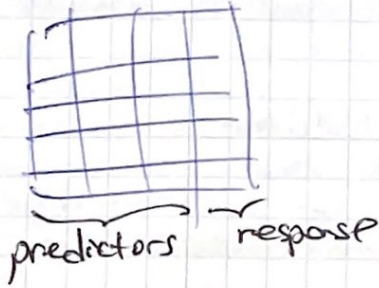
(53)

$$\text{knnmodel} = \text{fitcmm}(\text{tbl}, \text{"response"})$$

knnmodel: Sınıflandırma modeli hakkında bilgi içeren değişken

tbl: Bir tabloda saklanan eğitim verileri

"response": Aynı tabloda saklanan yantı değişkeninin adı.



Bir Modeli Güncelleme

Geçtiğimiz zaman veriler hakkında önceden sahip olduğumuz bilgiler nedeniyle veya sınıflandırma sonuçlarına baktığımızda sonra sınıflandırıcının güncellenmesi istenebilir. Mantık fonksiyonları kullanarak farklı seçenekleri ayarlayarak modeli güncelleyebilir ve değiştirebiliriz. Seçenek adı ve seçenek değeri için el girdiler sağlayarak seçenekleri ayarlayın.

(54) `model = fitc (tbl, "response", "optionName", optionValue)`

model: Sınıflandırıcı model nesnesi

tbl: Eğitim verisi

"response": Yanıt değişkeni adı

"optionName": Seçenek adı - Örneğin "Cost"

optionValue: Belirtilen seçeneğe ilişkin değer - Örneğin [0 10 20]

Matlab'da sınıflandırıcı oluşturmak için, `fitc` yerine `fitcknn` fonksiyonunu kullanın. Sınıflandırıcının adını yazarak `fitc` fonksiyonunu kullanın. Girdiler ~~veri~~ tabloları ve yanıt değişkeni olmalıdır. Örneğin, aşağıdaki söz dizimini bir k-én yakın komşusu (KNN) sınıflandırıcı modeli oluşturur.

(55) `mdl = fitcknn (tableData, ---
"responseVariable")`

"predict" fonksiyonu yeni gözlemlerin tahmin edilen sınıfını belirler.

(56) `predClass = predict(model, testData);`

↳ Bu TAHMİNİ VERİR

Bir modelin verileri ne kadar iyi sınıflandırdığını değerlendirmek için kayıp fonksiyonunu kullanabiliriz.

(57) `err = loss(model, testData)`

↳ Bu KAYIPI VERİR

Çıktı, test verileri üzerindeki sınıflandırmaya modelin yanlışlığından oluşur, daha düşük bir kayıp daha iyi bir tahmini gösterir.

Algoritmanın gözlemleri nasıl sınıflandığını görselleştirmek için "confusionchart" kullanabiliriz.

(58) `confusionchart(trueLabels, predictions)`

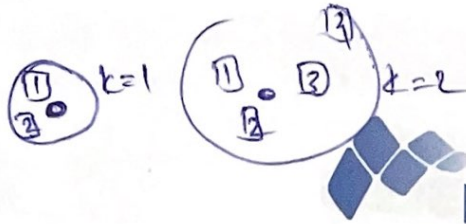
dataTest: ^{TrueLabel}

↳ preds

Burada sonuç, gerçek ve tahmin edilen sınıfın her bir kombinasyonu için gözlem sayısını gösteren bir kareküt matrisidir.

FN YAKIN KÖŞE SINIFLANDIRMA

Yeni bir örneği kategorize etmenin en basit yollarından biri, yeni örneğe benzer bir önceki örnekleri bulmak ve yeni örneği aynı sınıfa atamaktır. Bu, k-en yakın köşeli sınıflandırmanın ardındaki temel fikirdir. Bu yöntemi kullanırken, verilerin temel dağılımı hakkında herhangi bir varsayımda bulunmamız gerekir.



k=1
k=2

FIGES

NumNeighbors'at
göre seçil alır.

...../...../20.....

k-en yakın komşu sınıfından oluşturmak için, girdi olarak veri tablosuyla birlikte fitcnn isleminde kullanılır. Yanıt değişkeninin adını belirtir. Varsayılan olarak sınıf sayısı belirtilmek için "NumNeighbors" özelliği kullanılır.

(59)

`knnModel = fitcnn (tableData, --
"ResponseVariable",
"NumNeighbors", 4);`

Örnekler için daha fazla komşu kullanmak azırı uyum riskini azaltabilir.

fitcnn fonksiyon çağırısını değiştirmek yerine, model oluşturduktan sonra komşu sayısını değiştirebiliriz.

Model oluşturduktan sonra k-sayısını değiştirmek için "NumNeighbors" özelliği ile nokta gösterimi kullanılır.

(60)

`model.PropertyName = newPropertyValue`

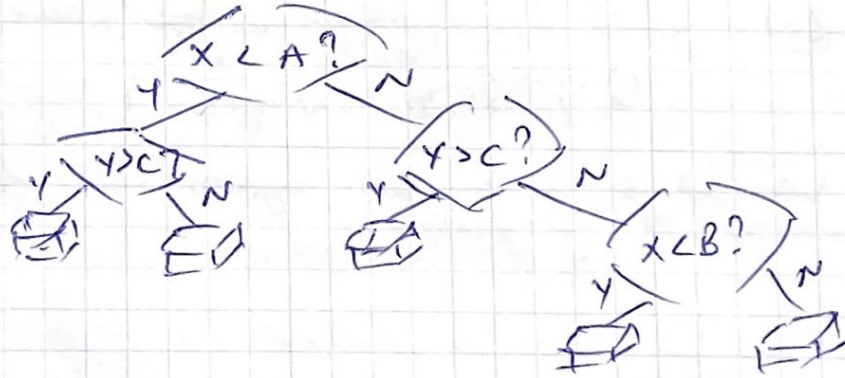
Örneğin
`model.KNN.NumNeighbors=7`

Veri setine daha iyi uyması için daha fazla parametre ile oynayabiliriz.

KARAR AĞAÇLARI

Gözetimler bir karar ağacına göre sınıflandırılır. K-en yakın Komşular (kNN), mesafe tabanlı sınıflandırma modelidir. Bu nedenle tüm tahmin edicilerin sayısal veya tüm tahmin edicilerin kategorik olmasını gerektirir. Birçok veri seti hem sayısal hem de kategorik tahmin ediciler içerir. Karar tahmin edicilerinin varsa, verilerin tamamını bir kNN modeli ile kullanamaz.

Burada karar ağacını yetiştirmeyi ve özelleştirmeyi kullanacağız. Karar Ağaçları, kNN modelleri gibi veriler hakkında herhangi bir varsayımda bulunmaz. Ancak tahmin edicileri bağımsız olarak ele aldıkları için sayısal ve kategorik tahmin edicilerin bir karışımına izin verilir.



"Split Criterion": Her seviyede optimum bölünmeleri belirlemek için kullanılan formül.

"MinLeafSize": Her yaprak düğümündeki minimum gözetim sayısı.

"Maximum Splits": Karar ağacında izin verilen maksimum bölme sayısı.

"NumNeighbors": Uygun bir zili karar ağacı modeli için kullanılır.

Simetri - ortalam
Simetri - left - ortanca (median)

veri doldurma



...../...../20.....

Bir sınıfta bir ağacı oluşturmak için, girdi olarak bir tablo ile "fitetree" fonksiyonu kullanılır ve çıktı değişken adı belirtilir.

(61) `treemodel = fitetree (tableData, ---
"ResponseVariable")`

Kara ağacı "view" fonksiyonu ile görselleştirilebilir.

(62) `view (treemodel, "node", "graph")`

"prune" fonksiyonu kullanarak sınıfta bir ağacın dallarının seviyesini değiştirebiliriz.

(63) `prunedTreemodel = prune (treemodel, ---
"Level", integer)` Sayısal Değer
Modeli değerlendirmek için de "evalModel" kullanılır.

(64) `evalModel (TestData, mdl)`

NAIVE BAYES

Naive Bayes kullanarak gözlemler sınıflandırılır. KNN Modelleri ve Karar Ağaçları, altta yatan verilerin dağılımı hakkında herhangi bir varsayımında bulunmaz.

Veri setinin temel bir dağılımda geldiğini varsayarsak, verileri istatistiksel bir örneklem olarak ele alabiliriz. Bu, aykırı değerlerin modelimiz üzerindeki etkisini azaltabilir.

Naive Bayes sınıflandırıcı, her bir sınıf içindeki tahmin edicilerin bağımsız olduğunu varsayar. Bu sınıflandırıcı nispeten basit problemler için iyi bir seçimdir.

"Distribution Names": Olasılıkları hesaplamak için kullanılacak dağılım.

"Width": Dağılımı seçme penceresinin genişliği ("Distribution Names", "kernel" olarak ayrıldı.)

"kernel": Kullanılacak kernel tipi

Bir Naive Bayes sınıflandırıcı oluşturmak için "fitcnb" fonksiyonu kullanılır.

(55) `nbmodel = fitcnb(tableData, --- "ResponseVariable")`
 Genellikle trainData

Varsayılan olarak, sayısal tahmin ediciler tahmini ortalaması ve standart sapmayla birlikte normal (Gauss) dağılımlar olarak modeller. Ancak, normalite varsayımı her zaman uygun olmayabilir. "DistributionNames" özelliği kullanarak dağılımı değiştirebiliriz.

(66) `nbmodel = fitcnb(trainingData, --- "ResponseVariable", --- "DistributionNames", distName);`

Bazı farklı değişkenler için farklı dağılımlar kullanabiliriz. Bu durumda, bir dizi dağılım adı kullanarak "DistributionNames" özelliğini ayarlayabiliriz.

(67) `nbmodel = fitcnb(trainingData, "response", ---
"DistributionNames", ["kernel", "mvn", "mvn"])`

Tekrarlanan ~~değer~~ değerlere sahip bir dizi oluşturmak için "repmat" işlevini kullanabilirsiniz.

(68) `d = repmat("kernel", 1, 2)
d =
"kernel" "kernel"`

DISKRİMİNANT ANALİZİ

Naive Bayes'e benzer şekilde diskriminant analizi, her bir tahmin sınıfındaki gözlemlerin normal bir olasılık dağılımıyla modellenilebileceğini varsayarak çalışır. Ancak her bir tahmin edicide bağımsızlık varsayımı yoktur. Bu nedenle, her sınıfa ait değişkenler bir normal dağılım uydurulur.

LINEAR DISKRİMİNANT ANALİZİ

Versayılar olarak her yantı sınıfının kovaryansının aynı olduğu varsayılır. Bu, sınıflar arasında doğrusal sınırlara neden olur.

⑦⑦ `daModel = fitcdiscr (dataTrain, "ResponseVarName")`

KUADRATİK DISKRİMİNANT ANALİZİ

Her kovaryans varsayımının kaldırılması, sınıflar arasında ikinci dereceden bir sınıra neden olur. Bunu yapmak için "DiscrimType" seçeneği kullanılır.

⑦⑦ `daModel = fitcdiscr (dataTrain, "ResponseVarName", ---
"DiscrimType", "quadratic")`

"DiscrimType": Kullanılan sınır türü

"Delta": Tahmin edicileri doğrusal bir sınıra dahil etmek için katsayı eğri (varsayılır: 0)

"Gamma": Doğrusal DA için kovaryans matrisini tahmin etmek kullanılacak düzeltilme.

Discriminant analizi için, her öngörünün her sınıfı için bir Gauss dağılımı varsayılır.

Bir diskriminant analizi sınıflandırıcısı oluşturmak için, girdi olarak tablo ile "fitcdiscr" fonksiyonunu kullanın ve yanıt değişkeninin adını belirtin. Varsayılan olarak, sınıflar arasındaki sınırlar doğrudur.

(72) `discrmodel = fitcdiscr(tableData, ...
"ResponseVariable");`

SUPPORT VECTOR MACHINES (SVM)

~~Veri seti sınıfları ayırmak istediğiniz varsayılan~~

Bir Destek Vektör Makinesi (DVM, SVM) algoritmasını, tüm veri noktalarını ayıran "en iyi" hiper düzlemi bularak verileri sınıflandırır.

"KernelFunction": Uygulanacak değişken dönüşümü.

"KernelScale": Çıktıdaki dönüşümünde önce uygulanan ölçeklerdir. (Varsayılan değer: 1)

"BoxConstraint": Yanıt sınıflandırma cetasını kontrol eden düzeltileştirme parametresi (varsayılan: 1)

Bir Destek Vektör Makinesi (SVM) oluşturmak için "fitcsvm" fonksiyonu kullanın.

~~svmmodel = fitcsvm(tableData, ...)~~

(73) `svmmodel = fitcsvm(tableData, ...
"ResponseVariable");`

DVM optimizasyonunun optimum hiper düzlemi, girdi verilerinin ölçeğinden etkilenir. Bu nedenle, bu algoritmayı kullanırken verilerin standartlaştırılması önerilir. Bir normalize fonksiyonu kullanarak eğitilmeden önce yapabiliriz. Alternatif olarak modeldeki "standardize" özelliğini "true" olarak ayarlayabiliriz.

(74) `svmmodel = fitcsvm(tableData, ...
"ResponseVariable", ...
"Standardize", true);`

Varsayılan olarak, "fitcsvm" doğrusal bir destek vektörü kullanır. "kernelFunction" özelliğini değiştirerek bunu değiştirebiliriz.

(75) `mdlsvm = fitcsvm(tableData, "faultCode"faultCode, "Standardize", true, ...
"kernelFunction", "polynomial")`

Bir SVM modeli oluşturup kayıtları bulmak için;

(76) `mdl = fitcsvm(trainPoints, "group", "kernelFunction", "polynomial");
mdlLoss = loss(mdl, testPoints)`

ÇOK SINIFLI DESTEK VEKTÖR MAKİNELERİ OLUŞTURMA

Çok sınıflı bir SVM modeli oluşturmak için adımlar
bir süreçtir.

ADIM 1: ikili SINIFLANDIRICI İLE BİR ŞABLON OLUŞTURMA
"templateSvm" fonksiyonu kullanarak ikili bir SVM
için bir şablon oluşturur.

(77) `template = templateSvm("PropertyName", PropertyValue)`

"template": Bir SVM sınıflandırıcı
"PropertyName": opsiyonel ~~değişken~~ ^{özellik adı} (varsayılan "kernelFunction")

"PropertyValue": Özellik değeri (örneğin "polynomial"). Birçok
özellik adı - değer çifti sağlayabiliriz.

ADIM 2: ÇOK SINIFLI BİR SVM SINIFLANDIRICI OLUŞTURMA.

(78) `ecocModel = fitcecoc(dataTrain, "y", "Learners", template)`

"ecocModel": ECOC sınıflandırıcı

"dataTrain": Eğitim verisi

"Learners": ikili sınıflandırıcıları belirlemek için özellik seti

"template": ikili sınıflandırıcı

Çok sınıflı bir Destek Vektör Makinesi (SVM) oluşturmak için "fitcecc" fonksiyonunu kullanın.

(79) `msvmModel = fitcecc(tableData, "ResponseVariable");`

SVM model seçeneklerini ayarlamak için "templateSVM" fonksiyonuna özellik ad-değer çiftlerini altıncı bir sablon öğrenici oluşturun.

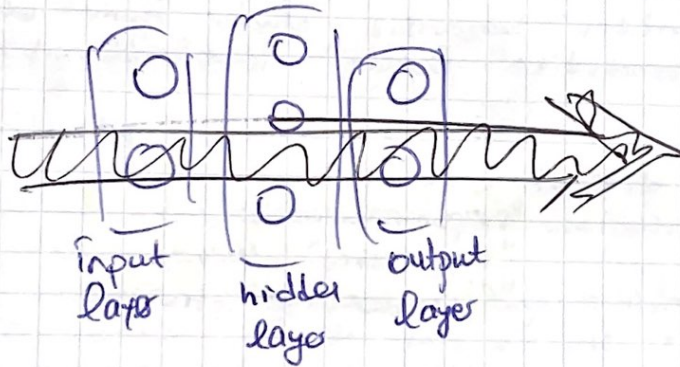
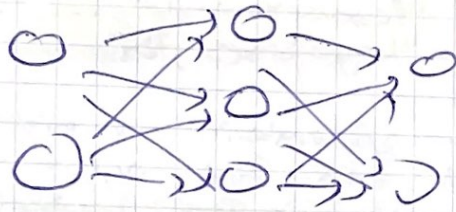
(80) `t = templateSVM("PropertyName", PropertyValue)`

Ardından sablon öğreniciyi "fitcecc" fonksiyonuna "Learners" özelliğinin değeri olarak verin.

(81) `mdl = fitcecc(tableData, "Response", ---
"Learners", t)`

NEURAL NETWORKS :

Sinir ağı bir dizi bağlı nöron aracılığıyla insan beyninin davranışını simüle eder. Bu ağılar, deneme-yanılma yoluyla nöronlar arasındaki bağlantıları ayarlayarak örüntüleri tanımayı öğrenir. Bu yöntem veriler hakkında varsayımlarda bulunmaz ancak verilerin normalize edilmesi gerektirir.



"Layer Sizes" : Tam bağlı katmanların boyutları (varsayımlar=10)

"Activations" : Tam bağlı katmanlar için aktivasyon fonksiyonları (varsayımlar)

"Validation Data" : Eğitim performans tespiti için değerlendirme verileri

MATLAB'da ~~bir sınıf ağı sınıflandırması~~ bir sınıflandırma için derin ağ modeli kurmak için girdi olarak tabloyla birlikte "fitnet" fonksiyonunu kullandıran ve yantı değişkeninin adını belirtir. Versaylar olarak, 10 nöronlu oluşan bir gizli katmana sahip bir sınıf ağı oluşturur. Ağıdaki gizli nöron sayısını belirtmek için "LayerSizes" özelliğini kullanın.

(82) `nnmodel = fitnet(tableData, ---
"ResponseVariable", ---
"LayerSizes", 24);`

Test setindeki tüm gözlemlerin tek bir sınıfta sınıflandırıldığına dikkat edin. Sınıf ağıları verilerin standartlaştırılması gerekebilir. Aksi takdirde bir ağına yakınsamakta başarısız olur. Bunu ele almak için "Standardize" özelliği true olarak ayarlanabilir.

(83) `nnmodel = fitnet (tableData, ---
"ResponseVariable", ---
"LayerSizes", 24, ---
"Standardize", true);`

(84) Bir sınıf ağında ağı aynı ölçekte için, doğruluk kaybı iyileşmeyi durdurduğunda eğitimi durdurmak için bir doğruluk kimesi kullanmak iyi bir uygulamadır. Doğruluk için kullanılan verileri belirtmek için "ValidationData" özelliği kullanılabilir. "ValidationPatience" özelliğinin ayarlanması, ağı belirli sayıda yinelemede sonra doğruluk kaybı iyileşmediyse eğitimi durdurmasını söyler.

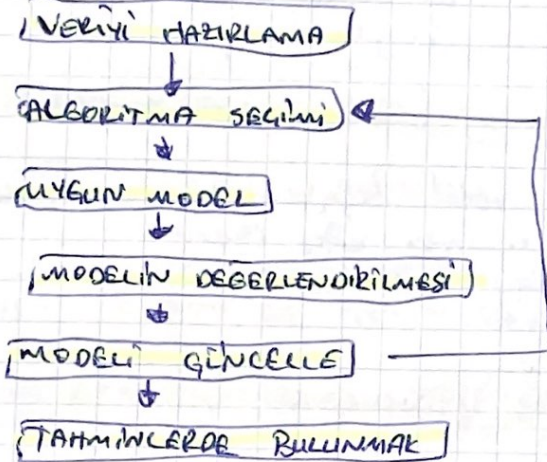
`nnmodel = fitnet (dataTrain, "FaultCode", "LayerSizes", 48, "Standardize", true, ---
"ValidationData", dataVal, "ValidationPatience", 30);`

(85)

Modeli önce oluşturup sonrasında hata tahmini için aşağıdaki kodu kullanabiliriz.

86) $mdl = \text{fitc}(\text{dataTrain}, \text{"Bağımlı Değerler"}, \text{--})$
 $\text{errRate} = \text{loss}(mdl, \text{dataTest})$
 → kurulacak olan model adı
 ↳ Bağımlı Değerler

TAHMİNİNE DAYALI MODELLERİN GELİTİRİLMESİ



CİAPRAZ DOĞRULAMA (CROSS VALIDATION)

Bir Modelin Değerlendirilmesi:

Bir modelin performansının değerlendirilmesinin basit bir yolu, verileri modeli eğitmek için bir alt küme ve test etmek için başka bir alt küme bölmektir. Eğitim setine bir model yerleştiririz ve ardından test setini kullanarak kaybı hesaplarız.

En iyi öğrenme algoritmasını veya yöntemini seçmek istediğinizde, her yöntem için kaybı hesaplayabilir ve en düşük kayba sahip yöntemi seçebilirsiniz.

Ancak kayıp verilerin belirli bir alt kümesi için yapılır, hesaplanır. Bir öğrenme algoritmasının söz konusu alt kümede iyi performans göstermesi, ancak diğer verilere iyi generalizasyon yapması mümkündür.

CROSS VALIDATION (Çapraz Doğrulama)

Çapraz doğrulama kullanarak belirli test alt kümesinin model değerlendirme sürecindeki etkisini azaltabiliriz.

Çapraz doğrulamanın arkasındaki fikir, model değerlendirme sürecini birden çok kez tekrarlamaktır. Süreci her seferinde farklı eğitim ve test alt kümeleri kullanır ve eğitim kümesine bir model uygular ve ilgili test kümesini kullanarak kaybı hesaplar.

CAPRAZ DOĞRULAMAYI KULLANMA (CROSS VALIDATION USING)

Bir model oluşturduğumuzda, model değişkeni hem matematiksel model hem de veri seti hakkında bilgi içerir.

Çapraz doğrulama için bölümler hakkındaki bilgi de model değişkeninde saklanacaktır.

CAPRAZ DOĞRULAMA MODELİNİN UYDURULMASI

Çapraz doğrulanabilir bir model oluşturmak için, model oluşturma sürecinde aşağıdaki seçeneklerden birisini seçeriz.

⑧ `mdl = fitcknn(data, "Response", "PropertyName", "PropertyValue")`

Property Name	Property Value	Doğrulama Tipi
"CrossVal"	"on"	10 kat çapraz doğrulama
"Holdout"	0'dan 1'e kadar skaler	Doğrulama için ayrılmış verilen kesitle uztama
"Kfold"	(k) skaler	K-katlı çapraz doğrulama
"Leaveout"	"on"	Birini dışarıda bırakma (Leave - one - out) çapraz doğrulama

• Daha büyük bir k değeri, daha büyük zaman gerektirir fakat zaman maliyeti düşürür.

FIGES

...../...../20.....

Eğer "cvpartition" fonksiyonu kullanarak oluşturulmuş bir bölünümü varsa, bunun yerine "fitting" fonksiyonuna bunu sağlayabiliriz.

(88)
$$cvpt = cvpartition(data.Response, "kfold", k)$$
$$mdl = fitcknn(data, "Response", "cvpartition", cvpt)$$

KAYBİN HESAPLANMASI (Calculating the loss)

Açpatı doğrulamak için bir modeli değerlendirmek için "kfoldloss" fonksiyonunu kullanın.

(89)
$$mdlLoss = kfoldloss(mdl)$$

Verilen k-kat açpatı doğrulama bölünümü oluşturmak için "cvpartition" işlevini kullanabiliriz.

(90)
$$pt = cvpartition(table.Response, \dots, "kfold", k);$$

data.faultCode

"cvpartition" seçeneğini kullanarak ~~fitcknn~~ ve ~~fitcdiscr~~ gibi model oluşturma işlevlerine bir bölüm aktarabiliriz.

(91)
$$mdl = fitcdiscr(data, "Response", "cvpartition", pt)$$

Açpatı doğrulamak için bir modeli değerlendirmek için model üzerinde "kfoldloss" fonksiyonunu kullanırız. Kter adının büyük / küçük harfe duyarlı olduğunu unutmayalım.

(92)
$$modelLoss = kfoldloss(mdl)$$

Ayrıca bir bölüm oluşturmadan da açpatı doğrulamak için bir model uygulatabiliriz. Aşağıdaki kod bir modeli tek başına doğrulamak ile

Az sayıda gözlem içeren bir veri seti göz önüne alındığında, ~~bu~~ dağılımı yöntemi olarak Leave-one-out validation tercih edilebilir.

TEMEL BİLEŞEN ANALİZİ PRINCIPAL COMPONENT ANALYSIS (PCA)

Temel Bileşen Analizi, n -boyutlu bir özellik yayını ortogonal bileşenlerde oluşan yeni bir n -boyutlu yayına dönüştürür. Bileşenler, verilerde açıklanan varyansa göre sıralanır.

$$[pcs, scrs, N, N, pctExp] = pca(X)$$

pcs: Temel bileşenler katsayıları. Her bir tahmin edicinin n -temel bileşene katkısı ($n \times n$ matris)

scrs: Temel bileşen skorları. X 'teki m gözlemin temel bileşen yayınıdaki gösterimleri ($n \times n$ matris)

pctExp: Her bir temel bileşen tarafından açıklanan verilerdeki varyans yüzdesi ($1 \times n$ vektör)

X : m -gözelemlilik ve n -tahmincilik veri matrisi ($n \times m$ matris)

Dönüştürülmüş değişkenler orijinal verilerle aynı miktarda veri içerir. Ancak verilerin bir miktar günlük içerdiği varsayıldığında açıklanan varyansın son birkaç yüzdesini içeren bileşenlerin bilgileri çok günlük temsil etmesi muhtemeldir.

TAHMİN EDİCİ SAYISINI AZALTMA

PCA, seçilen bir açıklanan varyans eşliğinin ötesindeki temel bileşenleri atarak boyut azaltma için kullanılabilir. Temel bileşenler tarafından açıklanan varyansı görselleştirmek için "pareto" fonksiyonu kullanılabilir.

Verilerin Ortalamaya olan yakınlık / dağılımı



Örnek
bileşen
pareto gradı kodu aşağıdadır. İlk 9 temel
bileşen ile görselleştirme yapıldığı gösterilmektedir. / / 20

```
(94) [pcs, scrs, N, N, pctExp] = pca(x);  
pareto(pctExp)  
xreduced = scrs(:, 1:9);
```

TEMEL BİLEŞENİN YORUMLANMASI

Temel bileşenlerin kendi başlarına fiziksel bir anlamı yoktur. Ancak deneysel dönüşümün katsayıları her bir değişkenin temel bileşene katkısını gösterir.

Örneğin, birinci temel bileşenin katsayıları 0.8, 0.05 ve 0.3 ise, birinci değişken en büyük katkıya sahiptir. Onu ikinci ve üçüncü değişken takip eder.

BI PLOT

Biplot fonksiyonunu kullanarak her bir değişkenin temel bileşenlere katkılarını görselleştirebiliriz. Genellikle veriler en büyük varyans miktarını açıklayan ilk iki temel bileşenle görselleştirilmek için kullanılır.

```
(95) biplot(pcs(:, 1:2), "VarLabels", varnames)
```

ISI HARİTASI (HEAT MAP)

Her bir değişkenin temel bileşenlere katkılarını bir ısı haritası olarak da görselleştirebiliriz.

Örneğin ısı haritasında ilk üç temel bileşenle gösterilmek için

```
(96) heatmap(abs(pcs(:, 1:3)), "VDisplayLabels", varnames);  
xlabel("Principal Component")
```


PARALEL KOORDİNATLAR ÇİZİMİ

PCA yanıt değişiminde bağımsız olarak gerçekleştirilebilir. Ancak, verilerin birden fazla kategoriye (örn. doğru-yanlış) sahip bir yanıt değişimi olduğunda, temel bileşen planlarının paralel koordinat grafiği yararlı olabilir.

(97) `parallelcoords(scrs, "Group", y, "Quantile", 0.25)`

PCA Analizini gerçekleştirmek için verilerin standartlaştırılması gerekir. Böylece tüm değişimler aynı standart sapmaya sahip olur. Bunu "zscore" fonksiyonunu kullanarak yapabiliriz.

(98) `normData = zscore(rawData)`

ÖRNEK OLARAK $T\{i, 1:end-1\} = zscore(T\{i, 1:end-1\});$

Dönüştürülmüş veriler ve her bir temel bileşen tarafından açıklanan varyans yitisi için "pca" fonksiyonunu kullanabiliriz.

(99) `[N, normData, N, N, pctExp] = pca(rawData)`
↓ örnek kod
`pca(T\{i, 1:end-1\})`

Temel bileşen tarafından açıklanan varyansı görselleştirmek için "pareto" fonksiyonunu kullanabiliriz.

(100) `pareto(pctExp)`

Pareto grafik ile örneğin ilk on temel bileşenin varyansın yaklaşık %90'ını açıkladığını görebiliriz. Varyansın el az %99'unu açıklamak için kaç temel bileşene ihtiyacımız var?

Görev: pctExp kümülatif toplamını hesapla.

Bu kümülatif toplamın %99'dan büyük olduğu poz durumlarını bulun.

(101) İnci kodları çıkan sonucu şu şekilde yorumlarız. / / 20

⇒ İlk (idx) tane temel bileşen varyansın %99'undan fazlasını açıklamaktadır.
Buradaki yorumda sonra (scrs'nin) ilk (idx) sütununu içeren bir PCApreds değişkeni oluştururuz.

(102) $PCApreds = scrs(:, 1:idx)$

DİTİDE SAKLANAN VERİLERİ KULLANARAK MODEL KURMA

Verilerimiz bir matrisde saklanıyorsa, uydurma işlevlerini kullanmak için bunları bir tabloya dönüştürmemiz gerekmektedir. Tahmin edicileri ve yanıt değişkenlerini sırasıyla bir matris ve vektör olarak sağlayabilirsiniz.

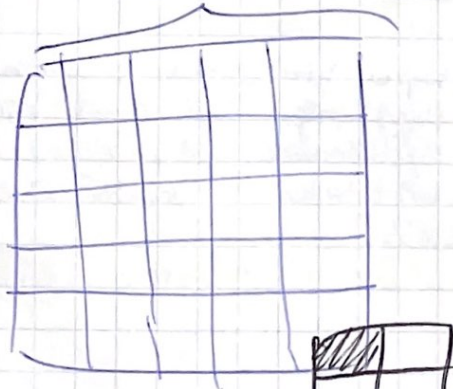
(103) $knnModel = fitcknn(X, y)$

knnModel: Sınıflandırma modeli hakkında bilgi içeren değişken

X: Sayısal matris olarak belirtilen tahmin edici değerler

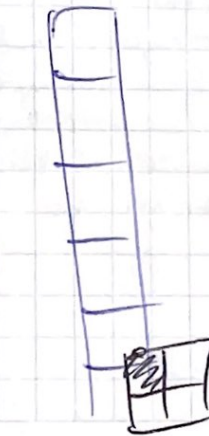
y: Sınıflandırma (yanıt) değerleri, vektör olarak belirtilir.

PREDICTORS



X

RESPONSE



y

pca: Özellik dönüştürme
gerektirir.



...../...../20.....

Oluşturulan bu PCA preds ile model kurmak ve kayıplarını gözlemlemek için örnek kod aşağıdadır.

(194) `mdl = fitcnb(PCA preds, T-faultCode, "k fold", 15);
mdlloss = kfoldloss(mdl)`

Classification Learner App'ta manuel olarak PCA Enable aktif ederek PCA analizi yapılabilir. Varsayılan olarak varsayın %95'ini açıklayan PCA bileşenleri kullanılır.

Özellik Seçimi

Veri setleri genellikle yanıtla herhangi bir ilişkisi olmayan tahmin ediciler içerir. Örneğin, tıbbi bir veri seti bir hasta kimlik numarası içerir. Ancak bu kimliği inceleyen tıbbi durumla herhangi bir ilişkisi olmayacaktır. Kimlik bir modele dahil edilmemelidir. Ancak model, bunu diğer tahmin ediciler gibi ele alacaktır.

Veri setleri ayrıca yüksek korelasyonlu öngörücüler içerir ve bu öngörücülerden yalnızca birinin modele dahil edilmesi gerekir.

Özellik seçim teknikleri, modele dahil edilecek tahmin edicilerin bir alt kümesini seçer.

Yapısal Özellik Seçimi

Bazı sınıflandırıcıların kendi yerleşik özellik seçim yöntemleri vardır.

Örneğin, karar ağacı modelleri doğru bir model oluşturmak için önemli olan tahmin edici değişkenleri belirlemek için kullanılabilir. "predictorImportance" yöntemine sahiptir. "predictorImportance" çıktısı, her bir tahmin edici için önem değerlerinin bir vektörüdür.

Aşağıda örnek bir kod verilmiştir.

...../...../20.....

(105) `tmodel = fitctree (T, "faultCode");`
`p=predictorImportance(tmodel);`
`tokeep = p > 0.001;`
`bar (p(tokeep))`
`x ticklabels (varNames (tokeep))`
`x tickangle (45)`

→ Çıktıdaki grafikler daha yüksek önem düzeyini görebiliriz.

Her bir öngörücü değişkenin önemini hesaplamak için bir karar ağacı modelinde "predictorImportance" kullanabiliriz.

(106) `predImp = predictorImportance (treeModel)`

NOT: "predictorImportance" işlevini ağırlık doğrulanmış karar ağacı modelleriyle kullanmamızdır.

Önem için bir eşik değeri belirleyerek hangi tahmin edicinin saklanacağını görebiliriz.

(107) `tokeep = p >= 0.001`

tokeep değerlerini görselleştirmek için bar grafiği kullanırız.

(108) `bar (p(tokeep))`
~~`x ticklabels (varNames (tokeep))`~~
`x ticklabels (T.Properties.VariableNames (tokeep))`

Yalnızca p değeri 0.001'den büyük olan tahmin ediciler ve faultCode yanıtı içeren seçili bir tablo oluşturmak için tokeep ögesini kullanırız.

(109) `selected = T(:, [tokeep true])`

Tahminci seçilen tahmin edicileri kullanarak "mdlPort" adında bir kütüphane öğrenme değeri olan bir sınıflandırma ağacı modeli oluşturup kayıtları hesaplayan örnek kod.

(110) `mdlPort = fitctree(selected, "faultCode", "Kfold", 5);`
`portLoss = kfoldloss mdlPort`

ÖZELLİK SIRALAMA ALGORİTMALARI

Özellik sıralama algoritmaları, belirli bir metriğe göre modelde ne kadar alakalı olduklarına bağlı olarak özelliklere puan atar. Bazı yaygın algoritmalar arasında Chi-Square, Minimum Redundancy Maximum Relevance (MRMR) ve Neighborhood Component Analysis bulunur.

Özellik seçme algoritmalarını verilere uygulamak için "fs" fonksiyonu kullanılır. Gözde tutulan özellik sıralama algoritması, en yüksekleri en doğru sıralanmış tüm özelliklerin konumlarını ve puanlarını döndürür.

(111) `[idx, scores] = fs(tblData, ResponseVarName)`

idx: Tahmincinin öğrenimine göre sıralanmış tahminci indeksleri
scores: Tahmin puanları

tblData: Tahmin değişkeni ve yanıt değişkeni içeren tablo verileri

ResponseVarName: Yanıt değişkeni adı.

Bu verilerde özelliklerin bir alt kümesini seçmek ve bunları modelle eğitmek için doğrusal indexlemeyi kullanabiliriz.

(112) `toKeep = idx(1:nfeatures);`
`selected = [tblData(:, toKeep), tblData(:, "ResponseVarName")];`
`mdl = fitctm(selected, "ResponseVarName");`

MATLAB'DA ÖZELLİK SIRALAMA ALGORİTMALARI

ALGORİTMA

FONKSİYON

TİPİ

Chi-Square

→

fschi2

→

Classification

Minimum Redundancy Maximum
(MRMR)

→

fsmmr / fsmmr

→

Classification / Regression

Neighborhood Component Analysis
(NCA)

→

fscna / fscna

→

Classification / Regression

f-Tests

→

fsftest

→

Regression

Laplacian

→

fslaplacian

→

Clustering (k-means)

Relief / ReliefF

→

relieff

→

Classification / Regression

113 `[idx, score] = fscorr2(tbl, "Response")`
DATA BAĞIMLI DEĞİŞKEN

⑪④ $\text{bar}(\text{score}(\hat{\text{idx}}))$

⑪ 5

```
takekeep = idx(1:15);  
selected = T(:, [takekeep, end]);  
DATA
```

(116) $\text{mdlPart} = \text{fitcTree}(\text{select}, \text{"response"}, \text{"kfold"}, n)$ Model oluşturma.
 $\text{partLoss} = \text{kfoldLoss}(\text{mdlPart})$ hata payı hesapla
 $\text{confusionchart}(\text{T_faultCode}, \text{kfoldPredict}(\text{mdlPart}))$ karışıklık matrisi hesapla

www.figes.com.tr

SIRALI ÖZELLİK SEÇİMİ

Herhangi bir öğrenme algoritması ile kullanılabilir bir özellik seçimi yaklaşımı, sıralı özellik seçimidir. Sıralı özellik seçimi, belirli bir tahmin modeli verildiğinde, tahmin edici değişkenleri sırasıyla ekleyerek ve kaldırarak, ardından modelin kalitesi üzerindeki etkisi değerlendirilerek yinelenen bir prosedürdür.

- Sıralı özellik seçimi, bir model oluşturur ve tahmin hatasını hesaplayan bir hata fonksiyonu gerektirir.
- Sıra, modelin oluşturulması ve tahmin edicilerden birini kullanarak hatanın hesaplanmasıyla başlar.
- İkinci bir tahminci eklenir ve hata tekrar hesaplanır.
- Hata azalır ise ikinci tahminci korunur. Hata azaldıkça modele daha fazla tahminci eklenir.

"sequentialfs" fonksiyonu sıralı özellik seçimlerini gerçekleştirir.

(117) `lokeep = sequentialfs(fun, X, y)`

lokeep: Nihai modele hangi tahmin edicilerin dahil edildiğini gösteren mantıksal vektör.

fun: Bir modele uygun ve kayıtları hesaplayan bir fonksiyon için fonksiyon tanıtıcısı

X: M-görüşüm ve n-tahminci içeren sayısal matris

y: n-yanıt ~~vektörünün~~ değerinin sayısal vektörü

! İlk girişin hata fonksiyonunun tanıtıcısı olduğunu unutmayın!

(118) `lokeep = sequentialfs(@errorfun, X, y)`

Çapraz doğrulama yöntemini belirtmek için isteğe bağlı "cv" özelliği kullanılabiliriz. Örneğin 7-katlı çapraz doğrulama kullanılabiliriz.

(119) `lokeep = sequentialfs(@errorfun, X, y, 'cv', 7)`

HATA FONKSİYONU YAPILI

Sıralı İntellekt seçimi, bir model oluşturur ve tahmin hatasını hesaplayan bir hata fonksiyonu gerektirir. Hata fonksiyonu aşağıdaki yapıya sahip olmalıdır.

- Dört Sayısal Girdi

- Modeli eğitmek için iki girdi (tahmin matrisi ve yanıt vektörü)
- Modeli değerlendirmek için iki tane (tahmin edici matris ve yanıt vektörü)

- Tahmin hatasını temsil eden bir skaler çıktı.

ÖRNEK HATA FONKSİYONU

```
function error = error_fun(x_train, y_train, x_test, y_test)
```

```
    mdl = fitcsvm(x_train, y_train);  
    ypred = predict(mdl, x_test);  
    error = nnz(ypred ~= y_test);
```

end.

NOT: Eğitim ve test veri kümelerini oluşturmanın gereği yoktur, "sequentialfs" işlevi, hata işlevini çağırmadan önce verileri dahili olarak bölümlendirir.

"predictorImportance" kullanımına Classification Trees algoritmasının itici vardır.

Hata fonksiyonunu yerel bir fonksiyon veya ayrı bir dosyaya yazmanın bir alternatifi de onemin bir fonksiyon oluşturmaktır.

Eğitim ve test verileri verildiğinde yanlı sınıflandırma sayısını hesaplayan anonim bir fonksiyon oluşturmak için aşağıdaki kalıp kullanılabilir.

(121)
$$\text{anonfun} = \lambda (X_{\text{Train}}, y_{\text{Train}}, X_{\text{Test}}, y_{\text{Test}}) \dots$$

$$\text{fitFunction}(X_{\text{Train}}, y_{\text{Train}}, X_{\text{Test}})$$

CREATING DUMMY VARIABLES (KUKLA DEĞİŞKENLER OLUŞTURMA)

Bu iyi bir yaklaşımdır, her kategori için bir kukla tahmin edici olmak üzere yeni kukla tahmin ediciler oluşturulmaktadır.

"dummyvar" fonksiyonu kullanarak bir kukla değişkenler matrisi oluşturabiliriz.

(122)
$$d = \text{dummyvar}(c)$$

Her satır bir gözleme karşılık gelir ve 1 değeri sahip bir elemana ve 0'a eşit diğer tüm elemanlara sahiptir. 1 değeri, o gözlemin atanmış kategorisine karşılık gelen sütunda görünür.

Bu matris artık kategorik vektör c yerine bir makine öğrenimi modelinde kullanılabilir; her sütun, c'nin o kategorisinin varlığını (1) veya yokluğunu (0) gösteren ayrı bir tahmin edici değişken olarak ele alınır.

Table verileri bazen bazı değişken içerebilir. Diğer sayısal ve bazı kategorik, grt.

Kategorik bir vektörü sayısal bir matrise dönüştürmek için "dummyvar" fonksiyonu kullanılabilir.

(123)
$$\text{numericMatrix} = \text{dummyvar}(\text{catVector})$$

$$\downarrow$$

 Kategorik değişken

HİPERPARAMETRE OPTİMİZASYONU

Model performansını iyileştirmeye çalışmak için bir makine öğrenimi modelinin özelliklerini değiştirebilirsiniz. Sıra gereklilikle modelin belirli özellik değerleriyle tekrar tekrar eğitilmesinde ve en iyi doğruluğu içeren kombinasyonun seçilmesinde oluşur.

HİPERPARAMETRE OPTİMİZASYONUNU GERÇEKLEŞTİRME

Hangi model özelliklerinin optimize edileceğini sağlamak için "OptimizeHyperparameters" özellik adı-değer çiftini kullanabilirsiniz.

(124) `mdl = fitcm(data, "ResponseName",
"OptimizeHyperparameters", params)`

`mdl`: Uygun model, optimize edilmiş model
`data, "ResponseName"`: Tahmin edilecek ve çıktı değerleri tablosu ve çıktı değeri adı.

`"OptimizeHyperparameters"`: Hiperparametre optimizasyonu için isteye bağlı özellik

`"params"`: En iyileştirilecek model özellikleri, bir string array veya cell array. "auto" ve "all" kullanımından önceki seçilmiş özellikleri optimize eder.

Optimizasyon sırasında, yinelenmeli güncellemelerin yanı sıra yinelenme sayısına karşı en iyi amaç fonksiyonu değerini içeren bir çıktı görüntülenir.

(125) `mdl = fitcm(data, "y", "OptimizeHyperparameters", "auto")`

"auto" olarak ayarlanması durumunda optimize eden özellikler model türüne bağlı olarak farklılık gösterebilir. Örneğin En Yakın Komşu sınıflandırması için optimize eden özellikler "Distance" ve "NumNeighbors"tır.

OPTİMİZASYON SEÇENEKLERİ

Varsayılan olarak, hiperparametre optimizasyonu Bayes Optimizasyonu kullanır ve 5 kat çapraz doğrulama kaybını en aza indirmeye çalışır.

Bir yapı kullanarak optimizasyon seçeneklerini belirtebiliriz. 10 katlı çapraz doğrulama kullanmak için bir çapraz doğrulama bölümü oluşturun ve ardından seçenekleri ad-değer çiftleri içeren bir yapıda oluşturun.

```
(126) port = cvpartition (y, "kfold", 10)
      opt = struct ("CVPartition", port);
      mdl = fitFcn (data, "y", "OptimizeHyperparameters", ---
                  "auto", "HyperparameterOptimizationOptions", ---
                  opt);
```

Yapıda birçok optimizasyon seçeneği ayarlanabilir. Örneğin, grafikler çizilebilir ve maksimum amaç fonksiyonu değeri/dürme sayısını ayarlayabiliriz.

```
(127) opt = struct ("ShowPlots", false, "MaxObjectiveEvaluation", 50);
```

TOPLULUK ÖĞRETİMİ

Bon machine öğrenim modelleri, yöntemleri zayıf öğreniciler olarak kabul edilir. Yani onları eğitmek için kullanılan verilere karşı oldukça duyarlıdır.

Örneğin, karar ağaçları zayıf öğrenicilerdir, biraz farklı iki eğitim verisi seti tamamen farklı iki ağaç ve sonuç olarak farklı tahminler üretebilir.

TOPLULUK MODELLERİNİN LİYDURULMASI (Ensemble models)

"fitcensemble" işlevi taşıyıcı öğrenicilerden oluşan bir sınıflandırma topluluğu oluşturur. Benzer şekilde "fitrensemble" fonksiyonu da bir regresyon topluluğu oluşturur. Her iki fonksiyon da aynı söz dizimine sahiptir.

(128) `mdl = fitcensemble(data, "ResponseName")`

`mdl`: Ensemble model değişkeni
`data`: Tahmin ediciler ve çıktı değişkenleri içeren tablo
`"ResponseName"`: Başlıkları Değişken

YAYGIN OLARAK KULLANILAN SEÇENEKLER

"Method" Bagging (bootstrap aggregation) ve boosting topluluk modellemesinde kullanılan en yaygın iki yaklaşımdır. "fitcensemble" işlevi her iki torbalama ve güçlendirme yöntemlerini sağlar. Örneğin rastgele bir orman oluşturmak için "Bag" yöntemini kullanın.

(129) `mdl = fitcensemble(data, "x", "method", "Bag")`

Varsayılan yöntem, ikili veya çok sınıflı sınıflandırma problemlerinin yanı sıra toplulukta öğrenicilerin tihine (learners)

"Learners" Toplulukta kullanılacak taşıyıcı öğrenici tihine belirtilebilir. "tree", "discriminant" ve "knn". Varsayılan öğrenici tihine belirtilen yöntemle bağlıdır. "Subspace" yöntemi varsayılan "knn" öğrenicisine sahiptir ve diğer tüm yöntemler varsayılan "tree" öğrenicisine sahiptir.

(130) `mdl = fitcensemble(data, "y", "Learners", "knn")`

"fitcensemble" işlevi her öğrenici türü için varsayılan ayarı kullanır. Öğrenici özelliklerini oluşturmak için bir sayı ile öğrenici sayısını kullanır.

(131) `mdl = fitcensemble(data, "y", "Learners", templateknn("NumNeighbors", 3));`

Birkaç farklı öğrenici türünden oluşan bir topluluk oluşturmak için öğrenicilerden oluşan bir hücre vektörü kullanabilirsiniz. Örneğin bir topluluk iki tür knn öğrenicisinden oluşabilir.

(132) `lrs = { templateknn("NumNeighbors", 3), templateknn("NumNeighbors", 5) }
mdl = fitcensemble(data, "y", "Learners", lrs)`

"NumLearningCycles": Her öğrenme döngüsünde, "Learners" içinde belirtilen her öğrenici için bir sayı ile öğrenici eğitilir. Varsayılan öğrenme döngü sayısı 100'dür. "Learners" yalnızca bir öğrenici içeriyorsa, varsayılan olarak 100 öğrenici eğitilir. Eğer "Learners" iki öğrenici içeriyorsa, varsayılan olarak 200 öğrenici eğitilir.

Bir ensemble model oluşturmak için "fitcensemble" kullanılır.

(133) `en = fitcensemble(table, "Response", ---
"Method", method)`

"fitcensemble" işlevi için bir öğreniciyi oluşturmak üzere "templateTree" işlevini kullanabilirsiniz.

(134) `t = templateTree(---
"PropertyName1", "PropertyValue1", ---
"PropertyName2", "PropertyValue2")`

DOĞRUSAL REGRESYON MODELLERİ

Doğrusal regresyon en basit ama en güçlü regresyon yöntemlerinden biridir.

Parametrik bir regresyon tekniğidir ve yantı, tahmin edici değişkenler açısından bilinen bir formül olarak modellerir.

Örneğin aşağıdaki formül gibi

$$\text{fuel economy} = c_0 + c_1 \cdot a + c_2 \cdot b + c_3 \cdot d$$

Uyum süreci, tahmin hatasını en aza indiren c_0, c_1, c_2 ve c_3 katsayılarını bulur.

LİNEER REGRESYON MODELİ OLUŞTURMA

"fitlm" fonksiyonunu kullanarak lineer regresyon modeli oluşturabiliriz.

$$\text{(135)} \quad \text{mdl} = \text{fitlm}(\text{data}, \text{"model spec"})$$

"mdl": Katsayıları ve modelle ilgili diğer bilgileri içeren bir regresyon modeli.

"data": Regresyon modeline uymak için kullanılan verileri içeren bir tablo.

"modelspec": Regresyon modeli özellikleri.

VERİ NEREDE DİZENLENİR?

Bir regresyon modelinde, yordayıcılar ile yantı arasındaki ilişki aşağıdaki formülle açıklanır.

$$y = \sum_{k=0}^n c_k \cdot f_k(x_1, x_2)$$

$\uparrow$ $\nwarrow$
 Response predictors

"fitlen" ilk girişi, tahmin ve yant içeren bir tablodur
 "fitlen" verisayısı olarak son sütunu yant olarak ve
 diğer tüm sütunları tahmin edici olarak kullanılır.



REGRESYON MODELİ BELİRLEME

Doğrusal bir regresyon modeli kullanırken, tahmin değiştirilene
 farklı izlevler uygulayabiliriz "fitlen" ikinci girişi olarak
 önceden tanımlanmış modellerden birini kullanabilir veya
 "Wilkinson-Rogers" notasyonunda bir formül sağlayarak
 bir model belirleyebiliriz

$$y = \sum_{k=0}^n c_k \cdot f_k(x_1, x_2)$$

model spesifikasyon
formülü

Ortak Önceden Tanımlanmış Modeller

MODEL NAME

MEANING

"linear":

Her tahminin için kesme
ve doğrusal terimler

"interactions":

Kesime, doğrusal terimler ve
farklı tahmin çiftlerinin tüm çarpımları

"quadratic"

Kesime, doğrusal terimler, etkileşim
ve kare terimler.

Wilkinson - Rages notasyonu : Responder ~ terimler

OPERATOR	MEANING	EXAMPLE
(+)	Bu terimi dahil et	" $y \sim x_1 + x_2$ ", kestirme terimi x_1 ve x_2 'yi içerir. $y = c_0 + c_1x_1 + c_2x_2$
(-)	Bu terimi hariç tut	" $y \sim x_1 + x_2 - 1$ " kestirme terimi hariç tutar. $y = c_1x_1 + c_2x_2$
(*)	Çarpım ve tüm alt düzey terimleri ekleyin	" $y \sim x_1 * x_2$ " kestirme terimleri x_1 , x_2 ve $x_1 * x_2$ 'yi içerir. $y = c_0 + c_1x_1 + c_2x_2 + c_3x_1x_2$
(:)	Yalnızca 1.5. terimini içerir.	" $y \sim x_1 : x_2$ ", yalnızca x_1, x_2 'yi içerir. $y = c_1x_1x_2$
(^)	Güç ve tüm alt düzey terimleri dahil edin.	" $y \sim (x_1^2) + (x_2^2)$ " kestirme noktasını x_1, x_2, x_1^2 ve x_2^2 'yi içerir. $y = c_0 + c_1x_1 + c_2x_2 + c_3x_1^2 + c_4x_2^2$

Aktarıdaki söz dizimini kullanarak bir doğrusal regresyon modeli oluşturabiliriz.

(136) `mdl = fitlm(trainingData)`

Test verileri ile yantı tahmin etmek için "predict" fonksiyonu kullanılabilir.

(137) `predictions = predict(linearModel, testData)`

(138) `mdl = fitlm(dataTrain, "modelspec")`
`yPred = predict(mdl, dataTest)`

Çizimde, aktarıcı grafikte aykırı değerler olabilir. "RobustOpt" özelliği "on" yaparak aykırı değerlerin model üzerindeki etkisini azaltabiliriz.

(139) `mdl = fitlm(dataTrain, "modelspec", ---`
`"RobustOpt", "on"`
`"PropertyName", PropertyValue)`
`"RobustOpt" "on"`

$y = a + b \cdot \sin(cx)$
 $y = at + bx + x^c$ } LİNEER DEĞİLDİR

DİZİDE SAKLANAN MODELLERİ KULLANARAK MODEL KURMA

Verilerimiz tek bir tablo yerine dizilerde saklanıyorsa, "fitlm" fonksiyonuna tahmin edici ve yanıt değerlerini sayısal diziler olarak sağlayabiliriz.

(140) `mdl = fitlm(X, y)`

mdl: Doğrusal regresyon modeli değişkeni

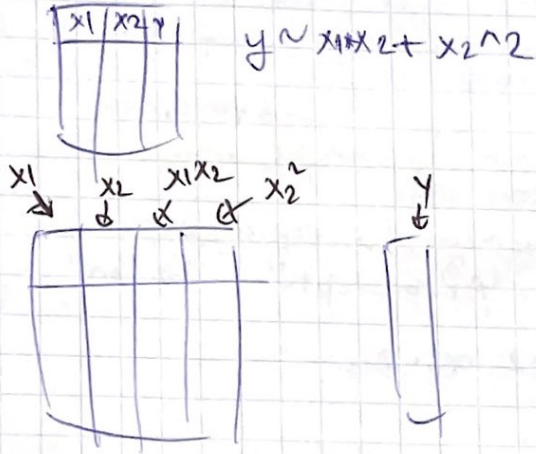
X: Sayısal matris olarak belirtilen tahmin değerleri

y: Vektör olarak belirtilen yanıt değerleri

REGRESYON MODELİ NASIL BELİRLENİR?

Farklı bir regresyon formülü uygulamak için iki seçeneğimiz vardır. Tahmin ediciler ve yanıt bir tabloda saklayabiliriz ve model spesifikasyonunu ayrı olarak sağlayabiliriz. Alternatif olarak, regresyon formülündeki her terim için bir sütun içeren bir matris oluşturabiliriz. Bu matrise tasarım matrisi (design matrix) denir.

$$y = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_1 x_2 + c_4 x_2^2$$



Veri setini x ve y değişkenlerinde oluşturabiliriz.

$x \rightarrow x_{train}, x_{test}$

$y \rightarrow y_{train}, y_{test}$ olarak bölünebilir. Bu durumda model eğitimi aşağıdaki gibi olabilir.

(41) $model = fitlm(x_{Train}, y_{Train})$

REGRESYON SONUÇLARINI GÖZLEMLERİME

"stepwise" ile ~~veri~~ yalnızca temel model doğrusal regresyon olduğunda kullanılır.

Doğrusal olmayan regresyon ve sınıflandırma problemleri için "sequential fs" fonksiyonunu kullanırız.

KADEMELİ DOĞRUSAL REGRESYONUN UYDURULMASI

...../...../20.....

Kademeli doğrusal regresyonun modelini sigdirmek için "stepwiselm" fonksiyonu kullanılır.

(142)

`stepwisemdl = stepwiselm(data, "modelspec")`

MODEL NASIL BELİRLENİR?

"stepwiselm" sizin için model seçer. Ancak model seçimin sürecini kontrol etmek için aşağıdakiler kullanabilirsiniz.

"modelspec" - fonksiyonun ikinci giriş başlığı modelini belirtir. "stepwiselm" bu modelle başlar ve belirli kriterlere göre terimleri ekler veya çıkarır.

Yaygın olarak kullanılan başlığı değerleri: "constant", "linear", "interactions", (sabit, doğrusal ve etkileşim terimleri), "quadratic" (sabit, doğrusal, etkileşim ve ikinci dereceden terimler)

"Lower" ve "Upper" - Modelin karmaşıklığını sınırlamak istiyorsanız bu özellikleri kullanın.

Örneğin aşağıdaki modelde keskinlikle kesme ve doğrusal terimleri istemeyen ancak derecesi 2'ye veya daha fazla olan terimleri istemeyecektir.

(143) `mdl = stepwiselm(data, "Lower", "linear", "Upper", "quadratic")`

Varsayılan olarak, "stepwiselm" modelleri yalnızca sabit terim kadar basit ve etkileşim modeli kadar karmaşık olarak değerlendirilir.

MODEL NASIL SEÇİLİR?

"stepwiselm", değiştirilen model önceki iterasyonda daha iyiyse, başlangıç modeline iteratif olarak terim ekler veya çıkarır.

"Bette", "Criteration" (kriter) özelliğinin değerine göre değerlendirilir. Varsayılan değer "sse" dir - (kareler hatası toplamı) - iterinde bir f-testi (an f-test on the sum of squared error) - "Criteration" özelliğini ayarlayarak bu ölçüye değiştirilebiliriz.

(144) `mdl = stepwiselm(data, "Criteration", "rsquared")`

Mevcut değerler; "sse" (varsayılan), "aic", "bic", "rsquared", "adjrsquared")

"stepwiselm", sadece doğrusal regresyon modelleri için iyidir.

Kademeli doğrusal regresyon modeli uydurmak için "stepwiselm" fonksiyonu kullanılabiliriz.

(145) `lmmdl = stepwiselm(table)`

Test verilerindeki yeni tahmin etmek için "predict" fonksiyonu kullanılabiliriz.

(146) `prediction = predict(lmmdl, testdata)`

Model formülü veya kök ortalama kareler hatası (RMSE) gibi model özelliklerini incelemek için notta indexlemeyi kullanabiliriz.

(147) `lmmdl.formula`
`lmmdl.RMSE`

Farklı bir modele başlamak için bir formül verilebilir. İkinci bir girdi kullanarak başlangıç modelini belirtebiliriz.

(148) $lmmdl = \text{stepwiselm}(\text{table}, "modelspec")$

DÜZENLENMİŞ DOĞRUSAL REGRESYON

Bir ver seti çok sayıda tahmin ediciye sahip olduğunda, doğru parametrik regresyon modelini seçmek zor olabilir. Tüm tahmin edicilerin dahil edilmesi gerektirir yere komayık bir model oluşturabilir ve bazı tahmin ediciler korelasyonlu olabilir.

Bu tür verilere doğrusal bir regresyon modeli uydu- mak, büyük varyanslı katsayılarla sonuçlanabilir. Yarı eğitim verilerindeki küçük değişikliklerle katsayıların değerleri büyük miktarda değişebilir.



RIDGE ve LASSO REGRESSION

Doğrusal regresyonda katsayılar, ortalama karesel hatanın (MSE) en aza indirilmesiyle seçilir. Ortalama karesel hata gözlenen ve tahmin edilen yanıt değer arasındaki farkın karesidir.

RIDGE REGRESSION MODELİ

Bir Ridge Regresyon Modeli'ni sığdırmak için "ridge" fonksiyonunu kullanabiliriz.

(149) $b = \text{ridge}(y, X, \text{lambda}, \text{scaled})$

y: vektör olarak bilinen yanıt değeri

X: sayısal matris olarak belirtilen tahmin değeri

lambda: Düzeltme parametresi

scaled: b'deki katsayı tahminlerinin orijinal veri ölçeğine döndürülüp döndürülmediğini belirlemek için {0,1} değerli bir işaret.

Bir karar ağacı kullanarak parametrik olmayan bir model oluşturabiliriz.

(150) `mdl = fitrtree(table, "ResponseName")`

Modeli test verileri üzerinde değerlendirmek için `loss` fonksiyonunu kullanabiliriz. Varsayılan olarak, model tahminleri ile yantı değişimindeki ortalamaya karşılık hesaplar.

(151) `rloss = loss(mdl, testTable)`

Tahmin fonksiyonu yeni veriler için yantıları tahmin edebilir.

(152) `preds = predict(mdl, data)`

Bir SVM model oluşturmak için aşağıdaki fonksiyonu kullanabiliriz.

(153) `mdl = fitrsvm(table, "ResponseName")`

Neural Network Regression modeli oluşturmak için aşağıdaki kodu kullanabiliriz.

(154) `mdl = fitrnet(table, "ResponseName", ...
"Standardize", true)`

Eğer veriler normalize değilse `"true"` ile veriler normalizasyonu yapabiliriz. `"prune"` fonksiyonu aşırı uyumu azaltmak için kullanılabilir.

(155) `treeModel = prune(treeModel, "Level", n)`

GAUSSIAN PROCESS REGRESSION

Gauss süreci regresyonu (GPR) bir başka parametrik olmayan regresyon tekniğidir.

GPR Modelleri, verileri tahminci değerlerinin yarı değerini tahmin etmenin yanı sıra istatistiğe bağlı olarak standart sapma ve tahmin özelliklerini de gösterir.

GPR Model oluşturmak için aşağıdaki kodu kullanabilirsiniz.

(156) `mdl = fitrgp(data, "responseName")`

Farklı bir modelle test etmek için

(157) `[yPred, yStd, yInt] = predict(mdl, dataNew)`

Yarı kısaca aşağıdaki gibidir.

(158) `gpmdl = fitrgp(data, "responseName")
preds = predict(gpmdl, data)
mdlLoss = loss(gpmdl, data)`