Тема 2: Линейна регресия

Обща методика за анализ на (големи) данни

Данните с които ще работим са структурирани в таблична форма (екселска). Редовете се идентифицират с конкретни наблюдения. Данните са записани във файл, който ще наричаме множество от данни (множество от наблюдения). Броят на редовете задават броя на наблюденията. Колоните са характеристиките (сойствата) на данните, наричат се още променливи. Съществува една колона – една променлива – която се нарича зависима, защото всяка стойност от колоната зависи от стойностите по реда. А всички останали колони се наричат независими променливи. Ако в зависимата колона стойностите са различни , то това означава, че се търси модел да прогнозира функционална зависимост между независимисите променливи и зависимата променлива. Това е регресионна задача. Ако в зависимата променлива има няколко стойности (две, три, четири), това означава във множеството от данни има няколко групи от наблюдения (класове). Стойността на зависимата променлива в случая се нарича етикет и той показва към клас принадлежи даденото наблюдение. Целта е да се построи модел за прогнозиране принадлежността на конкретно наблюдение към даден клас. Задачата се нарича класификационна.

Нека да припомним понятия, които ще използваме активно и е важно да разпознаваме обектите свързани с тях, както и да разбираме техния смисъл.

Модел – абстрактен обект , съответстващ на реален процес или обект. При анализа на данни моделът се представя чрез математически обекти – функции (линейни и/или нелинейни), оптимизационни задачи, представени чрез линейни и/или нелинейни уравнения и неравества.

Моделиране – процес за създаване на модел.

Метод – описание на процеса на решаване на даден модел. Методът може да решава не само модели – чрез него се решават математически обекти, като уравения, неравенства, търсене на екстремуми и други.

Алгоритъм – точна последователност от конкретни математически операции, които водят до определен резултат. Даден алгоритъм се реализира чрез програмен файл и изпълнение от компютър. Промяната на един ред в него води до друг алгоритъм. Да отбележим, даден метод може се реализира чрез различни алгоритми.

Методика – съвкупност от методи, които реализират отделни етапи от даден процес. Всяка методика се реализира от конкретен алгоритъм. По дадена методика може да има различни алгоритми, променяйки методите в методиката.

Структура на общата методика за анализ на данни

1.Четене на данните. Използват се различини команди в зависимост от вида и структурата на данните във файла.
2.При необходимост се прави предварителна обработка на данните – обработка на липсващи стойности в данните, скалиране на данните и други.
3.Избор на модел и параметри на модела. Решават се две основни задачи – регресионни и класификационни и в зависимост от това се избира модела. Команди от софтуера, с който се провежда анализа на данните (в случая Python) реализират чрез алгоритъм метод за решаване на модела. Командата всъщност стартира изпълнението на алгоритъм за решаване на модела.
4.Множеството от наблюдения се разделя на две подмножества, наречени тренировъчно (train) и тестово (test) подмножество. Върху тренировъчното подмножество се прилага командата от точка 3, т.е. върху тези данни се рашава модела. В резултат се намират търсените неизвестни на модела. Командата за решаване на модела е "**fit**" . Терминът е трениране на модела. Съществуват различни начини на избор на тренировъчното множество и съответно различни методи и алгоритми за провеждане на този избор.
5.Оценка на модела. Прилагат се стандартни методи за оценка на вече построения модел от точка 4. Моделът се оценява върху тестовото подмножество. Подробности в следващи лекции.

При реализация на тази обща методика се избират различни методи за реализация на процесите в точки 2, 3 и 4. Целта е да се изберат такива методи за реализация на точки 2,3 и 4 от общата методика, че оценките изведени в точка 5 да бъдат "възможно по-добри" в търсения смисъл.

Регресионен модел.

Започваме с един кратък пример, с който ще демонстрираме как се построява модел на линейна регресия в Python.

Пример 2.1.

```
#
import numpy as np
import matplotlib.pyplot as plt
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
x = x.reshape(-1, 1)

from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
```

```
linreg.fit(x,y)
y_pred = linreg.predict(x)
sc=linreg.score(x, y)
print('score=',sc)
a=linreg.coef_
b=linreg.intercept_
print('y = %6.3f x ' %a ,' + %6.3f '  %b)

# Plot outputs
plt.scatter(x, y,  color='black')
plt.plot(x, y_pred, color='blue', linewidth=2)
plt.show()
```

Известни са две едномерни множества x и y. Търси се зависимост между x и y от вида y = ax +b. Трябва да намерим коефициентите a и b. След намирането на тези стойности считаме, че е намерен прогнозен модел за множеството x. Този модел се нарича линейна регресия, а новите стойности , които се получават по формулата y_pred = a x+b, където x са дадените стойности. Новото множество y_pred са прогнозните стойности, получени от модела, за известното множество x. Разликата y – y_pred показва близостта на прогнозните стойности до истинските.

Командите

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(x,y)
y_pred = linreg.predict(x)
sc=linreg.score(x, y)
print('score=',sc)
```

изпълняват следното: дефинират процедурата LinearRegression чрез името linreg; моделът се тренира по данните x и y чрез командата linreg.fit(x,y). Командата linreg.predict(x) преобразува множеството x в множеството y_pred, а близостта мужду реалните и прогнозните стойности се пресмята чрез linreg.score(x, y) и след това се отпечатва. При изпълнение се получава

'score=', 0.952538038613988.

За смисъла на функцията score() може да се прочете от този сайт:

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Функцията score() връща стойността на R^2, пресметната за прогнозната стойност и реалната стойност (в нашия случай y_pred и y). Най-високата стойност за оценката е 1. Колкото по-близко до 1 е стойността, която пресмята функцията score(), толкова точността на модела е по-висока. Стремим се към модели, в които стойността на score() е висока, близка до 1. Много често R^2 се нарича коефициевнт на детерминация и в този смисъл score() връща коефициента на детерминация за линейната регресия. За извеждане на характеристиките на линейната регресия може да се използва и функцията r2_score() , която връща модифицирания R^2 за този регресионен модел.

За смисъла на функцията r2_score() може да се прочете от този сайт:

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

Следващите редове

```
a=linreg.coef_
b=linreg.intercept_
print('y = %6.3f x ' %a ,' + %6.3f '  %b)
```

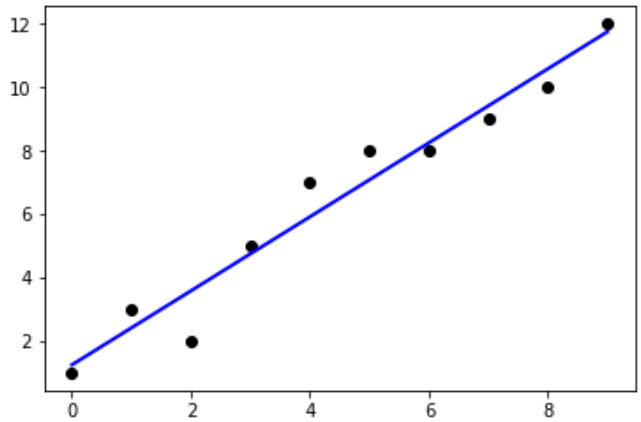
пресмятат регресионните коефициенти и отпечатват:

('y = 1.170 x ', ' + 1.236 ').

Продължаваме с редовете:

```
plt.scatter(x, y,  color='black')
plt.plot(x, y_pred, color='blue', linewidth=2)
plt.show()
print('SCORE : %.4f' %sc)
```

които редставят графично множествата x и y като точки в равнината, както и една права линия, която всъщност е линейната регресия (Фигура 2.1):



Фигура 2.1.

Продължаваме с пример на построяване на модел на линейна регресия върху файл с данни Advertising.csv :
https://www.kaggle.com/datasets/bumba5341/advertisingcsv
https://github.com/erkansirin78/datasets/blob/master/Advertising.csv
https://search.r-project.org/CRAN/refmans/glmtoolbox/html/advertising.html (плюс кратка характеристика на данните)
базов учебник https://www.statlearning.com/
данни https://www.statlearning.com/resources-first-edition

Файлът и описанието на данните могат да бъдат намерени в интернет. Накратко, файлът съдържа няколко колони: разходите за реклама на някакво изделие в телевизионните канали, радиото и вестниците, а последната колона съдържа информация за продажбите на това изделие. Целта е да се намери зависимост между разходите за реклама в различните медии и продажбите. В следващия пример ще търсим зависимост между разходите за реклама в телевизионните канали и продажбите на това изделие.

Пример 2.3.

```
#
# data  Advertising.csv
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
# url with dataset
# download the file
data_adv = pd.read_csv('Advertising.csv')
# advertising costs on TV, Radio, and Newspapers
# The sales are in column 4

X = data_adv.values[:,1:]

#we have assumed a linear relationship between advertising
#costs on TV and sales
X_TV=X[:,0]     # 0 = advertising costs on TV
                # 1 = advertising costs on Radio
y=data_adv.values[:,4]    # sales

# y = a X_TV + b
linreg = LinearRegression()
# Train the model using the all set
X_TV = X_TV.reshape(-1, 1)
linreg.fit(X_TV,y)
p = linreg.predict(X_TV)
sc=linreg.score(X_TV, y)
print('score=',sc)

a=linreg.coef_
b=linreg.intercept_
print('sales = %6.3f X_TV ' %a ,' + %6.3f '  %b)

# Plot outputs
plt.scatter(X_TV, y,  color='black')
plt.plot(X_TV, p, color='blue', linewidth=2)
plt.xticks(())
plt.yticks(())
plt.show()
```

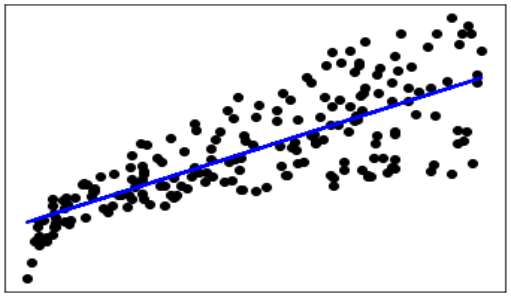
Данните за разходите в телевизионните канали са записани в променливата X_TV, докато данните от продажбите се намират в y. Дефинираме модела линейна регресия в променливата linreg и след това трениране модела върху пълните множества от данни X_TV и y. На Фигура 2.2. се виждат точките от множеството (по абсцисата X_TV, а по ординатата y), както и правата линия която е образ на регересионния модел, т.е. точки с координати от двете множества X_TV и p. Пресмятаме грешката от модела върху цялото множество чрез командата linreg.score(X_TV, y).

Коефициентът на детерминация е

SCORE : 0.6119.

В хода на работа на програмния код от Пример 2.3 се получава и вида на регресията, т.е. коефициентите на регресията:
'sales = 0.048 X_TV ', ' + 7.033 '.

Получава се следната 2D графика



Фигура 2.2.

Ще променим програмния код за Пример 2.3 с цел да намалим броя на наблюденията, по които се построява модела, т.е. ще въведем тренировъчно и тестово множество, и в същото време ще получим максимално възможен коефициент на детерминация, който ще отчитаме чрез функцията score().

Пример 2.4.

```
#
# data set  Advertising.csv
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
# url with dataset
# download the file
data_adv = pd.read_csv('Advertising.csv')
# advertising costs on TV, Radio, and Newspapers
# column 4 is sales

X = data_adv.values[:,1:]
y=data_adv.values[:,0] # нулевата колона е пореден номер
#we have assumed a linear relationship between advertising
#costs on TV and sales
X_TV=X[:,0]              # advertising costs on TV
y=data_adv.values[:,4] # sales
X_TV = X_TV.reshape(-1, 1)
# y = a X_TV + b
linreg = LinearRegression()

from sklearn.cross_validation import KFold
kf = KFold(len(X_TV), n_folds=10)
k=0
sm=0
for train_index, test_index in kf:
   #print('k=',k)
   linreg.fit(X_TV[train_index],y[train_index])
```

```
    score_test = linreg.score(X_TV[test_index], y[test_index])
    #print('score_train=',linreg.score(X_TV[train_index], y[train_index]))
    #print('score_test =',score_test)
    if sm < score_test :
        #print('k=',k)
        sm=score_test
        train_minindex = train_index
        test_minindex =  test_index

    k+=1
    print

# That is the model
linreg.fit(X_TV[train_minindex],y[train_minindex])
pred_xtest =linreg.predict(X_TV[test_minindex])
print('SCORE on test set: ',linreg.score(X_TV[test_minindex],
                                  y[test_minindex]))
# SCORE on all set
sctt=linreg.score(X_TV, y)
print('SCORE on all set: ',linreg.score(X_TV,y) )
a=linreg.coef_
b=linreg.intercept_
print('sales = %6.3f X_TV ' %a ,' + %6.3f '  %b)

# Plot test set
plt.scatter(X_TV[test_minindex], y[test_minindex],  color='black')
plt.plot(X_TV[test_minindex],          linreg.predict(X_TV[test_minindex]),          color='blue',
linewidth=2)
plt.xticks(())
plt.yticks(())
plt.show()

# Plot all set
plt.scatter(X_TV, y,  color='black') # the all test
plt.plot(X_TV[test_minindex], linreg.predict(X_TV[test_minindex]),
        color='blue', linewidth=6)
#plt.plot(X_TV, linreg.predict(X_TV), color='red', linewidth=2)
linreg.fit(X_TV,y)
p = linreg.predict(X_TV)
plt.plot(X_TV, p, color='red', linewidth=2)
plt.xticks(())
plt.yticks(())
plt.show()
```

Пример 2.4 започва, както Пример 2.2, с четене на данните от същия файл. Преди да построим модела на линейната регресия се появяват следните редове:

```
from sklearn.cross_validation import KFold
kf = KFold(len(X_TV), n_folds=10)
k=0
sm=0
for train_index, test_index in kf:
   linreg.fit(X_TV[train_index],y[train_index])
   score_test = linreg.score(X_TV[test_index], y[test_index])
   if sm < score_test :
      sm=score_test
      train_minindex = train_index
      test_minindex =  test_index

   k+=1
   print
```

Целта при тях е да използваме допълнителните функция KFold, за които може да се прочете от тук
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

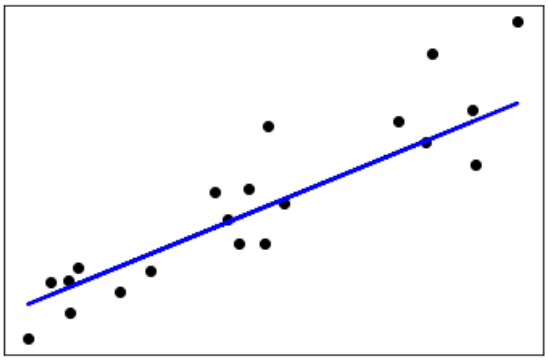
С командата
kf = KFold(len(X_TV), n_folds=10)
множеството X_TV се разделя на n_folds=10 части, подмножества, или по-точно целите числа от 0 до дължината на множеството X_TV се разделят на десет равни части, като 9/10 от тези цели числа (ще наричаме индекси) се запазват като инедкси за тренировъчното множество, а 1/10 от тези индекси се запазват за тестовото множество. С изпълнението на този фрагмент в примера се реализира елемент на самообучение на алгоритъма, т.е. алгоритъмът сам определя наблюденията, които да образуват тренировъчното и тестовото подмножество, така че построеният модел да достига максимално възможна стойност на score коефициента.

След това дефинираме цикъл по променливите train_index, test_index, които последователно вземат стойности от множеството kf. При първото потворение на цикъла train_index съдържа индекси от 20 до 199, а test_index съдържа индекси от 0 до 19. При следващите повторения на цикъла тези стойности се променят последователно. В тялото на цикъла се изпълняват следните команди. Първо моделът се тренира на множеството X_TV[train_index],y[train_index] чрез командата linreg.fit(X_TV[train_index],y[train_index]). След това пресмятаме оценката на модела чрез тестовото множество с командата score_test = linreg.score(X_TV[test_index], y[test_index]). Следва оператор if, който намира онези индекси, за които оценката score_test е най-малка. Индексите, за които се достига тази най-малка оценка се записват в двете променливи train_minindex и test_minindex. След приключване на цикъла в променливите train_minindex и test_minindex се намират индексите на тренировъчното и тестово множество, които простройват линейна регресия с възможно най-малка оценка score_test. След това моделът се тренира на избраното множество:

linreg.fit(X_TV[train_minindex],y[train_minindex])
и се пресмята оценката върху тестовото множество:
pred_xtest =linreg.predict(X_TV[test_minindex])
print('SCORE on test set: ',linreg.score(X_TV[test_minindex],
y[test_minindex]))
В нашия случай тя е:
('SCORE on test set: ', 0.78975122907881135)

И веднага пресмятаме оценката за цялото множество чрез така намерения модел:
sctt=linreg.score(X_TV, y)
print('SCORE on all set: ',linreg.score(X_TV,y))
Получава се:
('SCORE on all set: ', 0.61161122238071319).
Следват коефициентите на линейната регресия:
a=linreg.coef_
b=linreg.intercept_
print('sales = %6.3f X_TV ' %a ,' + %6.3f ' %b)
които са
('sales = 0.047 X_TV ', ' + 7.091 ') .

Следващите програмни редове
# Plot test set
plt.scatter(X_TV[test_minindex], y[test_minindex], color='black')
plt.plot(X_TV[test_minindex], linreg.predict(X_TV[test_minindex]), color='blue', linewidth=2)
plt.xticks(())
plt.yticks(())
plt.show()
построяват 2D графика (Фигура 2.3.), на която са показани точките от тестовото множество, и също така е показана и правата линия, която изобразява линейната регресия:

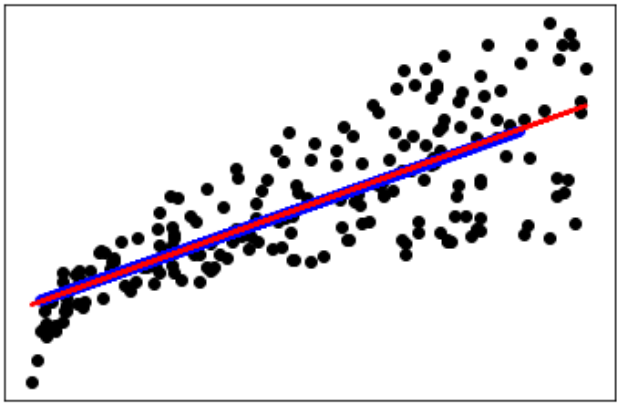


Фигура 2.3.

Последната група програмни редове
Plot all set
plt.scatter(X_TV, y, color='black') # the all test
plt.plot(X_TV[test_minindex], linreg.predict(X_TV[test_minindex]),
color='blue', linewidth=6)
linreg.fit(X_TV,y)

```
p = linreg.predict(X_TV)
plt.plot(X_TV, p, color='red', linewidth=2)
plt.xticks(())
plt.yticks(())
plt.show()
```

построяват 2D графика (Фигура 2.4.), на която са показани точките от цялото множество, и също така са представени две прави линии. По-късата права линия е линейната регресия, построена върху тестовото множество (същата от Фигура 2.3), а по-дългата е линейната регресия, построена за цялото множество. Както се вижда двете регресии „почти“ съвпадат, оценките чрез score функцията са 0.6119 (Пример 2.3.) и 0.6116 (Пример 2.4.).



Фигура 2.4.

От изпълнението на Пример 2.3. и Пример 2.4. се вижда, че може да се построи линейна регресия на база по-малко данни отколкото данните от цялото множество и същевременно оценката за точността на регресията не намалява.

Упражнения:

За упражнение направете примери за построяване на линейна регресия

Линейна регресия!

Задълбочен анализ на данните 'Advertising.csv':

https://www.ritchieng.com/machine-learning-evaluate-linear-regression-model/ (октомври, 2024)

Пример 1. Използвайте файла с наблюдения AutoInsurSweden.txt за да създадете модел на линейна регресия с една променлива.

Пример 2. За 141 области от различни държави на Европейския съюз е направена следната статистика: брой на лекарите със собствена практика (означаваме с Y) и брой на общото население (X1, измерено в хиляди); земната площ на областта (X2, измерена в квадратни км); и среден личен доход на лекарите от областта (X3, измерен в хиляди евро). Да се изведе зависимост между броя на лекарите (Y) и независимите променливи брой население, земна площ и среден личен доход. (Използвайте метода на линейната регресия и създайте модели между Y и X1; Y и X2; Y и X3;) Приложен е файл SMSA-C.txt с необходимите данни на приложения диск.

Пример за реализацията на метода на линейната регресия може да се намери тук https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/. Използвани са данните от файла insurance.csv.

Пример

https://vitalflux.com/boston-housing-dataset-linear-regression-predicting-house-prices/ (2024)

Проф. Иван Иванов

Октомври 2024