Анализ данных

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

Линейная регрессия
двоичная классификация

.

Иваново-2023

План

Линейная регрессия в бинарной классификации

В этом случае правильные ответы принимают лишь два значения: -1 и 1. А линейная регрессия возвращает вещественное число $y(x) = w_0 + w_1 x_1 + \dots w_k x_k$.
Вводим некоторую границу $b(Border)$ и полагаем, что ответ -1, если $y(x) \leq b$, иначе 1.

Классификация анкет на выдачу кредитной карточки

UCI_Australian.7z
Данные на 690 анкет, в каждой 15 полей:
Target, N0, N1, ..., N14
N0, N1, ..., N14 — некоторые числовые признаки
Target — целевое значение, 0 или 1. (надо сделать -1 и 1)

На Питоне

```
def lin_reg1(A,b):
    # добавление столбца из единиц в начало матрицы
    A = np.hstack((np.ones(len(A)).reshape((-1,1)), A))
    AtA = A.T.dot(A)
    AtA += 1e-8*abs(AtA).max()*np.eye(len(AtA))
    Atb = A.T.dot(b.reshape(-1))
    return np.linalg.solve(AtA,Atb)
```

Чтение данных:

```
csv = np.loadtxt("UCI_Australian.csv",
                 skiprows=1, delimiter=',')
Y = csv[:, 0]
Y = 2*Y-1
X = csv[:, 1:]
w = lin_reg1(X,Y)
print(w, '=w')
```

Матрица ошибок

y — верный ответ, $a(x)$ — наше предсказание. (border!)

	$y = 1$	$y = -1$
$a(x) = 1$	$True\ Positive(TP)$	$False\ Positive(FP)$
$a(x) = -1$	$False\ Negative(FN)$	$True\ Negative(TN)$

Доля правильных ответов:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Точность:

$$precision = \frac{TP}{TP + FP}$$

полнота:

$$recall = \frac{TP}{TP + FN}$$

ROC-кривая

ROC-кривая, (Receiver Operating Characteristic, рабочая характеристика приёмника)
Рассмотрим двумерное пространство, одна из координат которого соответствует доле неверно принятых объектов (False Positive Rate, FPR), а другая — доле верно принятых объектов (True Positive Rate, TPR):

$$FPR = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{TP + FN}$$

	$y = 1$	$y = -1$
$a(x) = 1$	$True\ Positive(TP)$	$False\ Positive(FP)$
$a(x) = -1$	$False\ Negative(FN)$	$True\ Negative(TN)$

Свойства

- TPR(border), FPR(border) — функции от border.
- $0 \leq FPR, TPR \leq 1$.
- Если $a(x) == -1$, то $FPR = TPR = 0$
- Если $a(x) == 1$, то $FPR = TPR = 1$
- Если border увеличить, то FPR, TPR будут расти.
- Если ответ случайный, то $FPR \approx TPR$.

	$y = 1$	$y = -1$
$a(x) = 1$	$True\ Positive(TP)$	$False\ Positive(FP)$
$a(x) = -1$	$False\ Negative(FN)$	$True\ Negative(TN)$

На Питоне

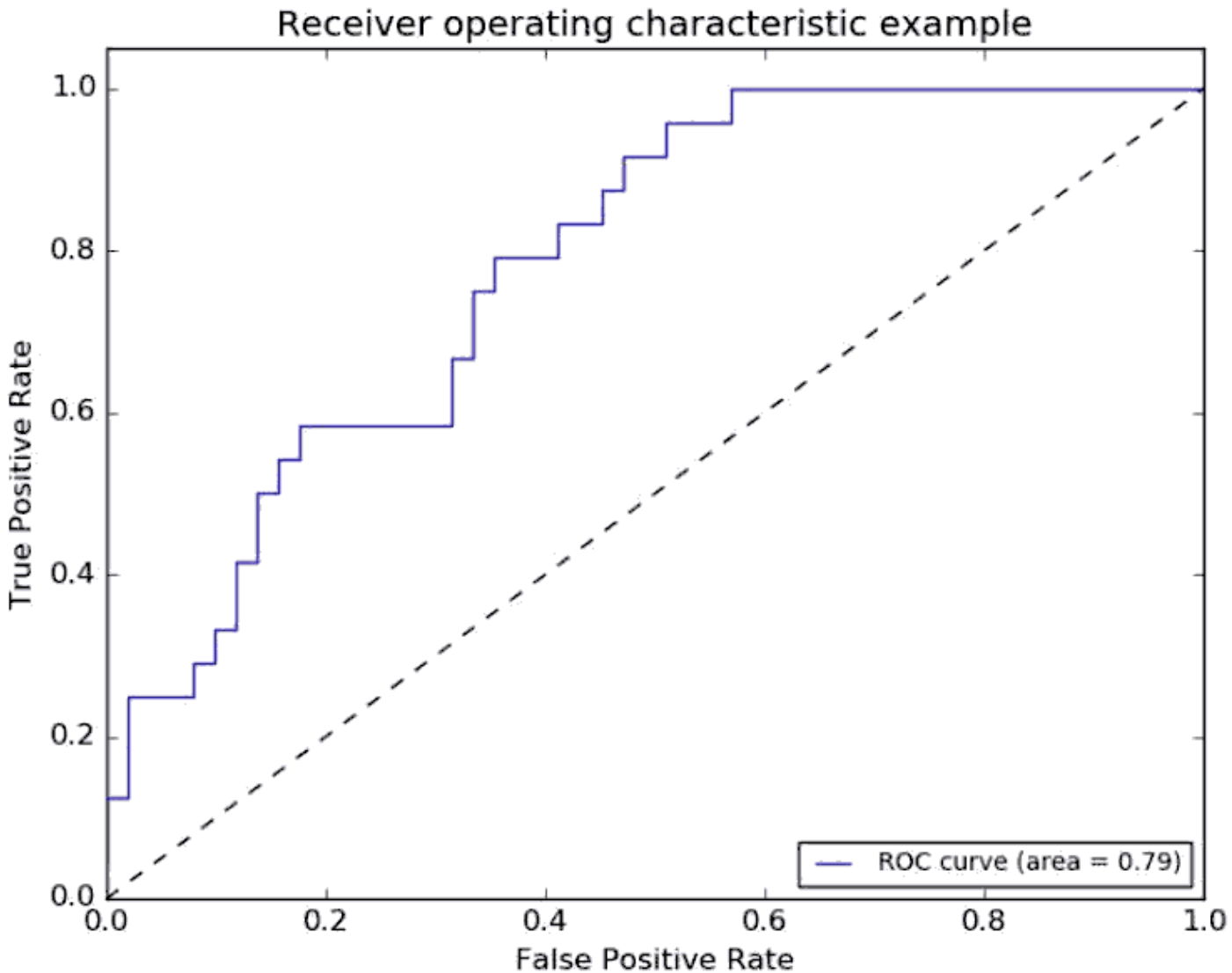
```
def TP_FP_FN_TN(Y, X, w, border):
  #param Y: вектор ответов (0 или >0)
  #param X: обучающие вектора (len=n)
  #param w: веса, вектор длины n+1
  #param border: заданная граница w0+w.Xi
  TP = FP = FN = TN = 0
  for i,x1 in enumerate(X):
    a_x = w[0] + np.dot(x1, w[1:])
    if y[i]>0:
      if a_x >= border: TP += 1
      else:             FN += 1
    else:
      if a_x >= border: FP += 1
      else:             TN += 1
  return TP, FP, FN, TN
```

На Питоне, ROC-кривая

Напомню, что X, Y, w уже готовы

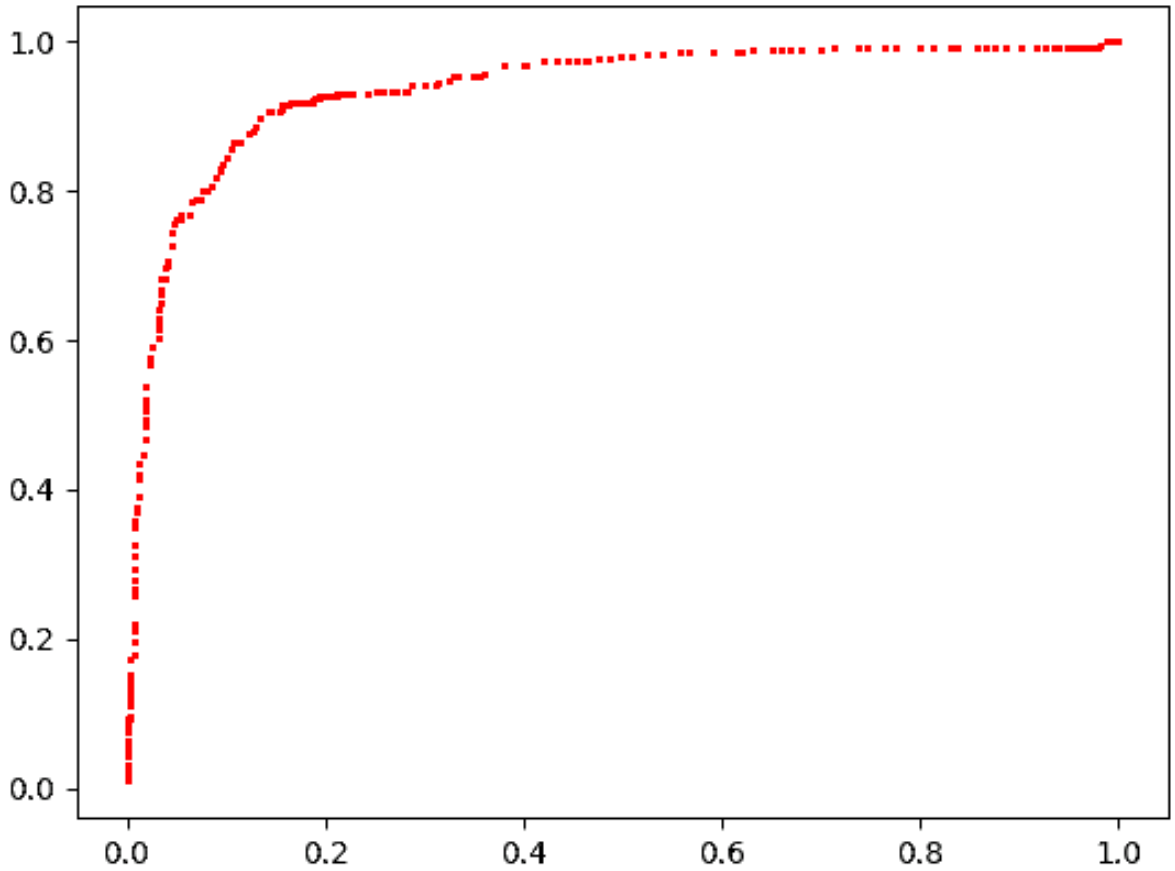
```
cx, cy = [], []
for ib in range(-150, 150):
    TP, FP, FN, TN = TP_FP_FN_TN(Y, X, w, ib/100)
    cx.append(FP/(FP+TN))
    cy.append(TP/(TP+FN))
plt.scatter(cx, cy, s=1, color='red')
plt.show()
# или, через tk_:
#tk_.plot(cx, cy, color='green', mx=600, my=600)
```

Пример ROC-кривой



Пример ROC-кривой

ROC-кривая из задачи UCI_Australian



На Питоне

```
def trianlge_area(x0,y0,x1,y1,x2,y2):#площадь треугольника
    x1 -= x0;  y1 -= y0
    x2 -= x0;  y2 -= y0
    return (x1*y2 - x2*y1)/2

def AUC(x, y): # Area Under Curve
  # (x,y) - векторы с координатами точек
  area = 0
  for i in range(1, len(x)):
    area+=trianlge_area(1,0,x[i-1],y[i-1],x[i],y[i])
  return area

print(f'AUC={AUC(cx, cy):7.3f}')  # AUC=  0.933
```

Задание

1. Проанализируйте, какие показатели больше влияют не результат.
2. Рассмотрим целевую функцию S(border) равную:
4*количество клиентов, кому надо дать карту, но её не дали
+количество клиентов, кому не надо дать карту, но её дали
При каком параметре border S(border) будет минимальна?
Напечатайте матрицу ошибок в этом случае.

Сокращение размерности

Метод главных компонент.
На плоскости (в пространстве) дано множество векторов.
Будем считать, что их среднее арифметическое равно 0. Как найти их основное направление?
Квадратичная форма на плоскости:

$$f(x, y) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2$$

или, в матричном виде:

$$f(v) = f(x, y) = (x, y) \cdot \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Сокращение размерности

В трёхмерном пространстве это выглядит так:

$$f(v) = f(x, y, z) = (x, y, z) \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

причем матрица обязательно симметричная. В матричном виде это записывается так:

$$f(v) = v^t \cdot A \cdot v,$$

где A — матрица квадратичной формы.

Сокращение размерности

Definition

Квадратичная форма называется положительно определённой, если $f(v) > 0$ для всех ненулевых векторов v.
Квадратичная форма называется неотрицательно определённой, если $f(v) \geq 0$ для всех ненулевых векторов v.

Theorem

Квадратичная форма является положительно определённой, если все главные миноры её матрицы > 0.
Квадратичная форма является неотрицательно определённой, если все главные миноры её матрицы ≥ 0.

Сокращение размерности

Theorem

Любую квадратичную форму можно привести ортогональные преобразованием к диагональному виде.
Иначе: для любой квадратичной форму существует ортонормальный базис, в котом её матрица имеет диагональный вид:

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0, & \lambda_2 & \dots & 0 \\ & & \dots & \\ 0, & 0 & \dots & \lambda_n \end{pmatrix}.$$

Не ограничивая общности будем полагать:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

Такие базисные вектора будем называть главными осями, а числа λ_i — главными значениями.

Сокращение размерности

Можно убедиться, что главные оси

$$v_i = \begin{pmatrix} v_{i1} \\ v_{i2} \\ \dots \\ v_{in} \end{pmatrix}$$

являются собственными векторами матрицы A:

$$A \cdot \begin{pmatrix} v_{i1} \\ v_{i2} \\ \dots \\ v_{in} \end{pmatrix} = \lambda_i \cdot \begin{pmatrix} v_{i1} \\ v_{i2} \\ \dots \\ v_{in} \end{pmatrix}$$

Сокращение размерности

Как находить собственные вектора и значения матрицы? Если матрица A невырожденная, то система линейных уравнений $Av = w$ имеет единственное решение (теорема Крамера). Система уравнений $Av = 0$ всегда имеет нулевое решение, и если есть ещё и ненулевое, это означает, что матрица A вырождена, то есть $\mathrm{detA} = 0$.
Пусть $Av = \lambda v$, тогда

$$(A - \lambda E)v = 0,$$

где E — единичная матрица. Таким образом, если v — собственный вектор с собственным значением λ, то оператор $A - \lambda E$ вырождена и, следовательно $\det(\mathrm{A} - \lambda \mathrm{E}) = 0$. Этот определитель является многочленом от λ степени n (характеристический многочлен матрицы). Т.е. надо найти характеристический многочлен, его корни и для каждого корня найти ненулевые решения системы уравнений $Av = \lambda v$.
Для симметричной матрицы A есть более простые алгоритмы.

Сокращение размерности

```
def optimal_basis(V):# оптимальный базис для строк матрицы
    '''
    В строках (my) матрицы V лежат вектора (длины mx).
    Возвращает пару (w,B), где
    B - оптимальный базис для этих векторов,
        то есть матрицу mx*mx, в СТОЛБЦАХ которой лежат эти
    w - собственные значения
    '''
    my, mx = V.shape
    X2 = np.zeros((mx, mx))  # sum(x_i^2)
    for v1 in V:    # по строкам матрицы V
        X2 += np.outer(v1,v1)
    X2 /= my
    w, B = np.linalg.eigh(X2)
    return w[::-1], B[:, ::-1]
```

Метод главных компонент

```
def AUC1(Y, X):
    w = lin_reg(Y, X)
    #print(w, '=w')
    cx, cy = [], []
    for ib in range(-150, 150):
        TP, FP, FN, TN = mlu.TP_FP_FN_TN(X, Y, w, ib / 100)
        cx.append(FP / (FP + TN))
        cy.append(TP / (TP + FN))
    plt.scatter(cx, cy, s=1, color='red')
    n_col = X.shape[1]
    plt.savefig(f't:\\ROC_{n_col:02d}.png', dpi = 100)
    plt.clf()
    return  AUC(cx, cy)
```

Метод главных компонент

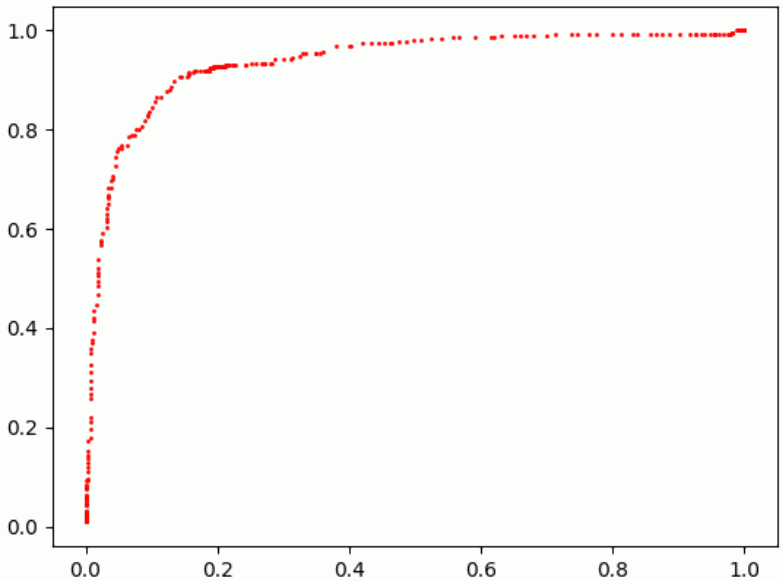
```
eval, basis = mlu.optimal_basis(X)
print(np.sqrt(eval), '=values')
X1 = X.dot(basis)
for i in range(len(eval)-1, 0, -1):
    print(f'i={i:2d}, AUC1={AUC1(Y, X1[:, :i]):7.3f}')
```

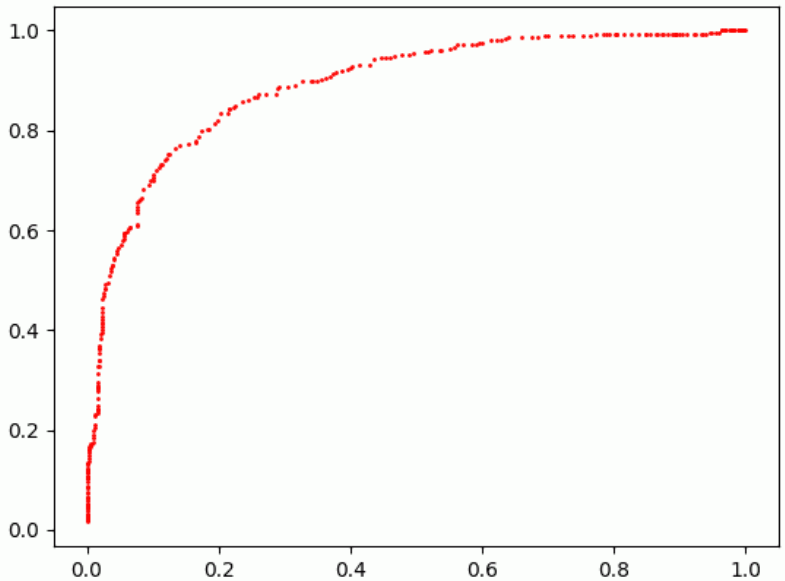
Метод главных компонент

```
i=13, AUC1=  0.926
i=12, AUC1=  0.926
i=11, AUC1=  0.884
i=10, AUC1=  0.849
i= 9, AUC1=  0.848
i= 8, AUC1=  0.844
i= 7, AUC1=  0.844
i= 6, AUC1=  0.844
i= 5, AUC1=  0.813
i= 4, AUC1=  0.809
i= 3, AUC1=  0.669
i= 2, AUC1=  0.639
i= 1, AUC1=  0.685
```

Пример ROC-кривой

$dim = 14$ $dim = 10$





Более подробные картинки см. в папке ./ROC_PCA/ROC*.png и их анимация в ./ROC_PCA/ROC_PCA.gif