Advanced Application of Artificial Intelligence and Digital Transformation

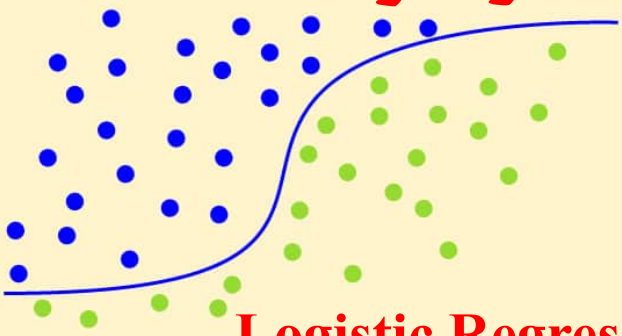
K.N. Toosi University of Technology

Artificiel Intelligence and Soft computing

K.N.TOOSI University of Technology

طبقه بندی

رگرسیون لجستیک



Logistic Regression

نمونه مسایل طبقه بندی

Classification- Binary decision

مسایل زیادی تصمیم گیری دودویی هستند و منجر به ایجاد یک طبقه بندی می‌شوند نظیر:

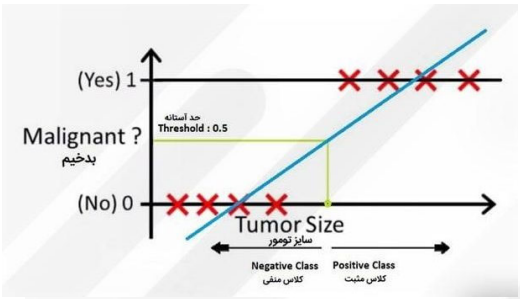
- طبقه‌بندی ایمیل‌ها به دو دسته‌ی ایمیل‌های اسپم یا غیر اسپم
- طبقه‌بندی معاملات آنلاین به دو دسته‌ی کلاهبرداری یا غیر کلاهبرداری
- طبقه‌بندی تومورهای بدخیم یا خوش‌خیم.

$y \in \{0, 1\}$

0: “Negative Class” (e.g., benign tumor)
1: “Positive Class” (e.g., malignant tumor)

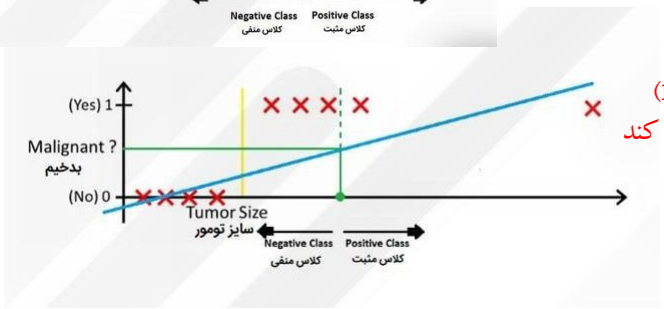
Artificial Intelligence

طبقه بندی و رگرسیون



If $h_\theta(x) \geq 0.5$, predict “y = 1”

If $h_\theta(x) < 0.5$, predict “y = 0”

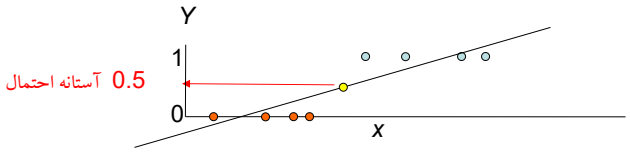


داده‌ی پرت (Rxwdhu) حدس را دچار خطا می کند

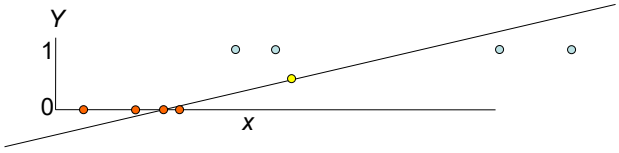
Adv. App. of AI and DT

4

طبقه بندی و رگرسیون



آستانه احتمال 0.5



- تغییر شیب با تغییر تعداد نمونه‌ها
- برونیابی غیر واقعی

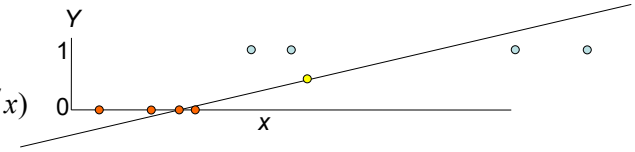
$$P(Y=0|x)+P(Y=1|x)=1 \quad or \quad P(Y|x)=1$$

احتمال بین صفر و یک است و خارج از آن مفهوم ندارد

طبقه بندی و رگرسیون

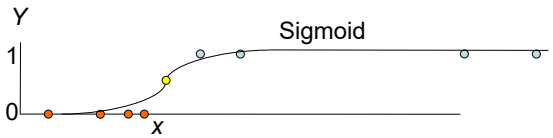
برچسب

$$P(Y=1|x,\theta)=\sigma(\theta^T x)$$
$$+$$
$$P(Y=0|x,\theta)=1-\sigma(\theta^T x)$$



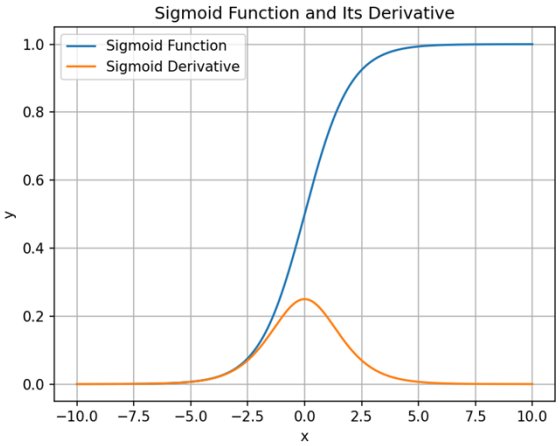
$$P(Y|x,\theta)=1$$

$$\sigma(\theta^T x)=?$$



تابع سیگموید یک تابع مناسب مسایل تصمیم گیری دوگانه است و در رگرسیون لجستیک از این تابع استفاده می شود.

تابع سیگموئید



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} = \sigma(x)\left(1-\sigma(x)\right)$$

مثال: توزیع برنولی یک توزیع گسسته است که مقادیر یک (در صورت موفقیت آزمایش) و صفر را (در صورت شکست) می گیرد. احتمال موفقیت آزمایش برابر p و احتمال شکست آن برابر $q=1-p$ است (مثال احتمال در شیر یا خط).

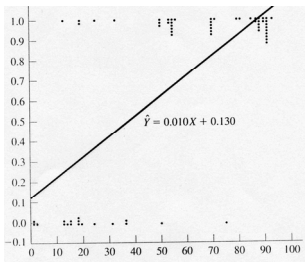
تابع سیگموئید برای توزیع برنولی مناسب است

$$\hat{y} = P(Y=1|x) = p(x) = \sigma(x)$$

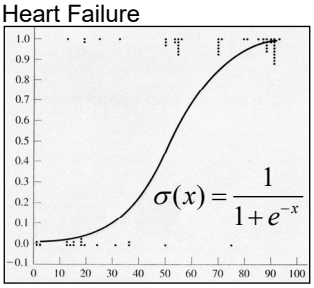
مثال رگرسیون لجستیک

احتمال مرگ ناشی از سکته قلبی

Heart Failure



Heart Failure



- Linear regression Extrapolates outside 0-1 and not as good empirically
- It is not appropriate for probability

- Sigmoidal is bounded between 0 and 1
- It is differentiable

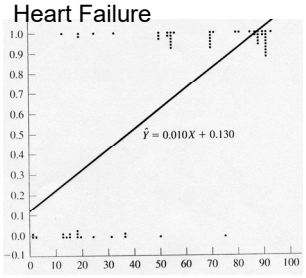
رگرسیون لجستیک

تابع فرضی

رگرسیون خطی با یک ویژگی $h_\theta(x) = \theta_0 + \theta_1 x$

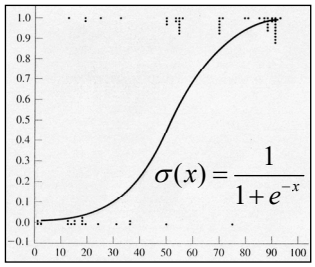
رگرسیون لجستیک با یک ویژگی $h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$

Heart Failure



- Linear regression Extrapolates outside 0-1 and not as good empirically
- It is not appropriate for probability

Heart Failure



مثال: احتمال مرگ ناشی از سکته قلبی

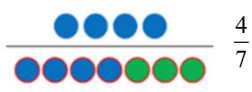
- Sigmoidal is bounded between 0 and 1
- It is differentiable

نسبت احتمال- بخت

نسبت احتمال- بخت (Odds)

محاسبه بخت بر اساس نسبت احتمال وقوع به احتمال عدم وقوع انجام می شود. بنابراین مقدار بخت، بر خلاف مقدار احتمال می تواند بزرگتر از یک نیز باشد. بطور مثال فرض کنید احتمال خوش خیم بودن غده سرطانی برای یک شخص $\frac{4}{7}$ برآورد شده است، بنابراین مقدار بخت برای این فرد $\frac{4}{3}$ می باشد.

Probability vs Odds

	Risk	Odds
Mathematically	$P(p) = \frac{p}{p+q}$	$O(p) = \frac{\frac{p}{p+q}}{\frac{q}{p+q}} = \frac{p(p+q)}{q(p+q)} = \frac{p}{q}$
Graphically	 $\frac{4}{7}$	 $\frac{4}{3}$

$$\hat{y} = P(Y=1|x) = p(x)$$

$$p(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}} \qquad Odds(\pi(x)) = \frac{p(x)}{1-p(x)} = e^{(\theta_0+\theta_1 x_1)}$$

لگاریتم طبیعی بخت

تبدیل لوجیت (Logit Transformation)

لگاریتم طبیعی بخت به عنوان تبدیل لوجیت یا (Log-Odds) شناخته می شود. در رگرسیون لجستیک با استفاده از تبدیل لوجیت بر روی احتمال وقوع x (برآورد مقدار y) می توان رابطه خطی رگرسیون را به شکل زیر نمایش داد.

$$Logit\left(p(x)\right) = Ln\left(Odds(p(x))\right) = Ln\left(\frac{p(x)}{1-p(x)}\right) = Ln\left(e^{(\theta_0+\theta_1 x_1)}\right) = \theta_0 + \theta_1 x_1$$

با تبدیل لوجیت یک رابطه خطی می توان نوشت

$$\hat{y} = E(Y|X=x) = P(Y=1|X=x) = p(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1)}}$$

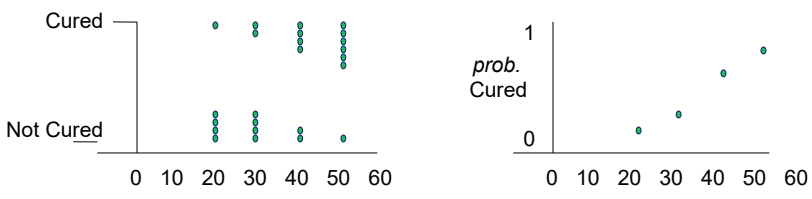
مقدار احتمال از لوجیت می تواند محاسبه شود

$$Logit\left(p(x)\right) = Ln\left(\frac{p(x)}{1-p(x)}\right) \Rightarrow \frac{p(x)}{1-p(x)} = e^{Logit(p(x))} \Rightarrow p(x) = \frac{e^{Logit(p(x))}}{1+e^{Logit(p(x))}}$$

Adv. App. of AI and DT 11

Non-Linear Pre-Process to Logit (Log Odds)

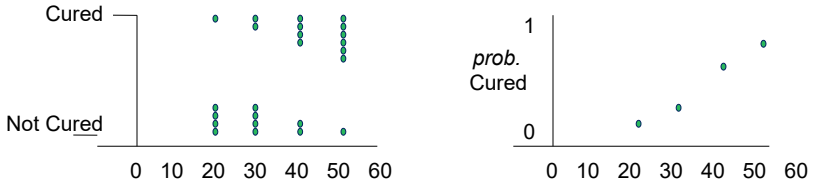
Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients
20	1	5	.20
30	2	6	.33
40	4	6	.67
50	6	7	.86



Adv. App. of AI and DT 12

Non-Linear Pre-Process to Logit (Log Odds)

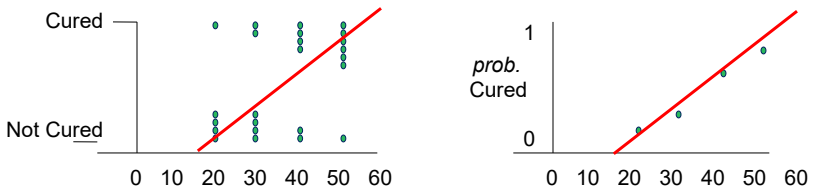
Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients
20	1	5	.20
30	2	6	.33
40	4	6	.67
50	6	7	.86



Adv. App. of AI and DT 13

Non-Linear Pre-Process to Logit (Log Odds)

Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients
20	1	5	.20
30	2	6	.33
40	4	6	.67
50	6	7	.86

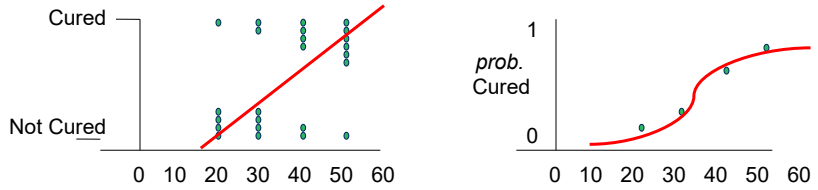


Adv. App. of AI and DT 14

Non-Linear Pre-Process to Logit (Log Odds)

آیا از رگرسیون خطی بدون برون یابی نامطلوب می توان استفاده کرد؟

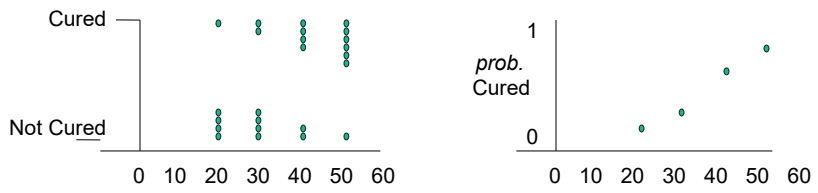
مشابه رگرسیون غیرخطی می توان از رگرسیون خطی بر روی لوجیت استفاده نمود.



Adv. App. of AI and DT 15

Non-Linear Pre-Process to Logit (Log Odds)

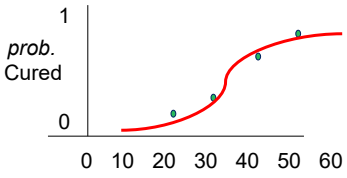
Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients	Odds: p/(1-p) = # cured/ # not cured	Logit Log Odds: ln(Odds)
20	1	5	.20	.25	-1.39
30	2	6	.33	.50	-0.69
40	4	6	.67	2.0	0.69
50	6	7	.86	6.0	1.79



Adv. App. of AI and DT 16

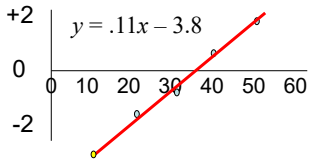
Non-Linear Pre-Process to Logit (Log Odds)

Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients	Odds: p/(1-p) = # cured/ # not cured	Logit Log Odds: ln(Odds)
20	1	5	.20	.25	-1.39
30	2	6	.33	.50	-0.69
40	4	6	.67	2.0	0.69
50	6	7	.86	6.0	1.79



مثال : مقدار احتمال p برای میزان مصرف ۱۰
(دقت کنید که مساله برون یابی است)

$$\text{Logit}(10) = .11(10) - 3.8 = -2.7$$



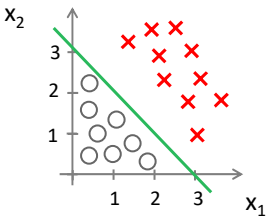
$$p(x) = \frac{e^{Logit(p(x))}}{1+e^{Logit(p(x))}} \quad \rightarrow p(x) = \frac{e^{-2.7}}{1+e^{-2.7}} = 0.06$$

با استفاده از فرمول مقدار احتمال از طریق لوجیت قابل محاسبه است

Adv. App. of AI and DT 17

مرز خطی تصمیم گیری

Linear Decision Boundary

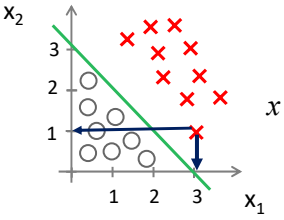


$$\text{Predict } y = 1 \quad if \quad -3 + x_1 + x_2 \geq 0$$

$$h(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = \theta^T x$$

معادله خط : $x_2 = 3 - x_1 \quad \Rightarrow 3 - x_1 - x_2 = 0$

$$\theta^T = [-3 \quad 1 \quad 1] \quad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \quad h(\theta) = -3 + x_1 + x_2$$



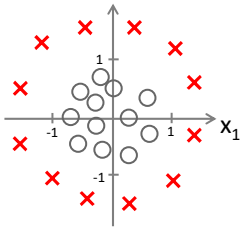
مثال: برای نقطه نمایش داده شده کلاس را مشخص کنید

$$x = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \rightarrow h(\theta) = -3 + 3 + 1 = 1 > 0 \Rightarrow y = 1$$

Adv. App. of AI and DT 18

مرزهای غیرخطی تصمیم گیری

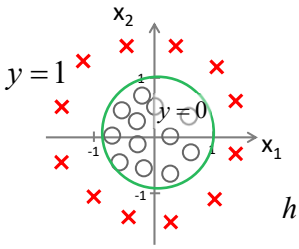
Non-linear decision boundaries



اگر مرز داده ها خطی نباشد چگونه عمل کنیم؟

مرزهای غیرخطی تصمیم گیری

مثال:



$$\text{Predict} \quad y=1 \quad if \quad -1+x_1^2+x_2^2 \geq 0$$

استفاده از تابع چند جمله ای با مرتبه دو

$$h(\theta)=\theta^T x$$

$$h(\theta)=\theta^T x=\theta_0+\theta_1 x_1+\theta_2 x_2+\theta_3 x_1^2+\theta_4 x_2^2+\theta_5 x_1 x_2$$

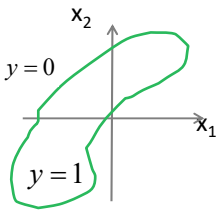
معادله دایره:

$$x_1^2+x_2^2=1 \Rightarrow -1+x_1^2+x_2^2=0$$

$$h(\theta)=\theta^T x=-1+0\times x_1+0\times x_2+1\times x_1^2+1\times x_2^2+0\times x_1 x_2$$

$$\theta^T=[-1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0] \qquad x=\begin{bmatrix}1\\ x_1\\ x_2\\ x_1^2\\ x_2^2\\ x_1 x_2\end{bmatrix}$$

مرزهای غیرخطی تصمیم گیری



استفاده از تابع چند جمله ای با مرتبه های بالاتر

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

مساله تعیین مرز تصمیم گیری یک مساله طبقه بندی است.
این مساله برچسب دار است و یادگیری ماشین با نظارت است.

$$h(\theta) = \theta^T x$$

چگونه θ تعیین می شود؟

رگرسیون لجستیک

رگرسیون لجستیک با چند ویژگی

تابع فرضی

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots)}}$$

تابع هزینه: (اثبات از ماکزیمم احتمال)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

To fit parameters: $\theta \quad \min_\theta J(\theta)$

حداقل کردن تابع هزینه و تعیین ضرایب

رگرسیون لجستیک

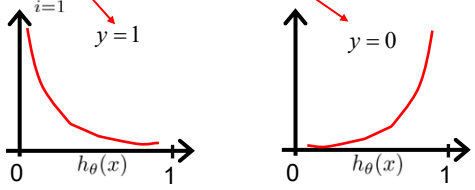
رگرسیون لجستیک با چند ویژگی

تابع فرضیه

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2+\theta_3 x_3+\ldots)}}$$

تابع هزینه

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))]$$



حداقل کردن تابع هزینه و تعیین ضرایب

To fit parameters: $\theta \quad \min_\theta J(\theta)$

تخمین برای شرایط جدید

To make a prediction given new: x

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

رگرسیون لجستیک

در رگرسیون لجستیک تابع هزینه لگاریتمی انتخاب می شود

- این تابع باعث می‌شود مدل، پیش‌بینی‌های خود را برای نمونه‌هایی که احتمال بالایی دارند به دقت انجام دهد.
- تابع لگاریتمی هزینه، **محدب** است (مشتق دوم آن یک مقدار مثبت است، به این معنا که تنها یک مینیمم دارد و در نتیجه، الگوریتم‌های بهینه‌سازی می‌توانند به مینیمم مطلق برسند و مدل به خوبی آموزش ببینند

```
from sklearn.linear_model import LogisticRegression


# Train the logistic regression model
model = LogisticRegression()
model.fit(X, y)
```

روشهای حداقل کردن تابع هزینه

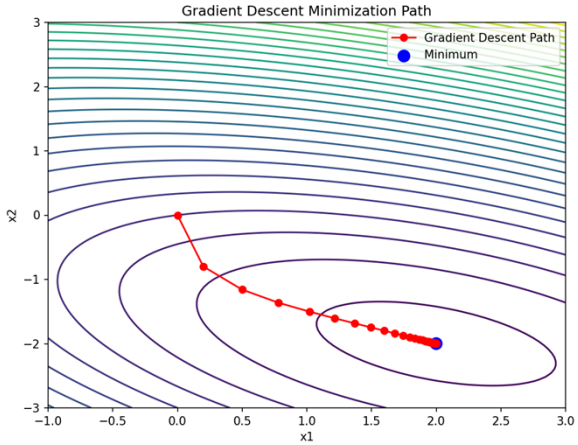
گرادیان کاهشی (Gradient Descent)
گرادیان کاهشی دسته‌ای (BGD: Batch Gradient Descent)
گرادیان کاهشی تصادفی (SGD: Stochastic Gradient Descent)
مختصات کاهشی (Coordinate Descent)
گرادیان مزدوج (CG:Conjugate Gradient)

روش‌های مبتنی بر بهینه‌سازی عددی مانند
روش نیوتن
روش شبه نیوتنی BFGS (Broyden–Fletcher–Goldfarb–Shanno)
روش شبه نیوتنی L- BFGS (Limited-memory BFGS)

- روش‌های دیگری نیز وجود دارد که در اینجا توضیح داده نمی شود و می توانید مطالعه نموده و از آنها استفاد کنید مانند:
Mini-Batch Gradient Descent, Momentum-Based Methods,Adam (Adaptive Moment Estimation), Trust Region Methods, Trust Region Methods
- در کتابخانه sklearn برای مدلهای مختلف رگرسیون روشهای مختلفی از حداقل کردن تابع هزینه گنجانده شده است.

روشهای حداقل کردن تابع هزینه

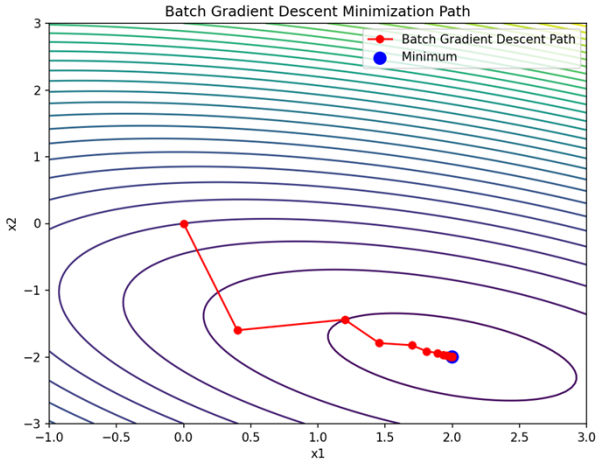
گرادیان کاهشی (Gradient Descent)



گرادیان کاهشی از تمام داده ها یکجا استفاده می کند و زمان محاسبات را در داده های حجیم خیلی زیاد خواهد شد.

روشهای حداقل کردن تابع هزینه

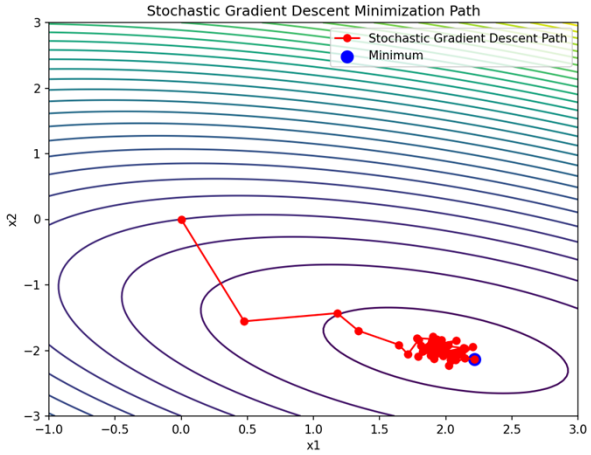
گرادیان کاهشی دسته‌ای (BGD: Batch Gradient Descent)



گرادیان کاهشی دسته‌ای از تمام داده ها یکجا استفاده نمی کند و زمان محاسبات را کاهش می دهد

روشهای حداقل کردن تابع هزینه

گرادیان کاهشی تصادفی (SGD: Stochastic Gradient Descent)



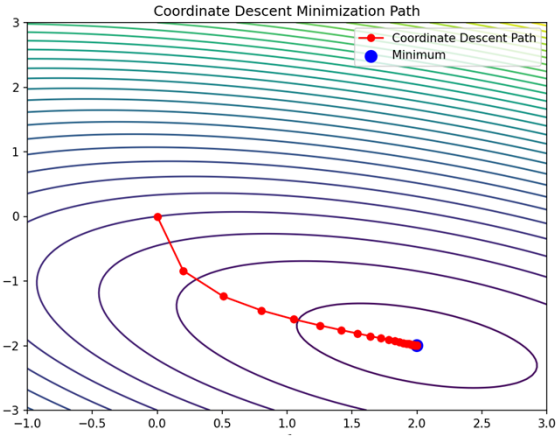
روش مشابه

```
solver=
‘sag’, ‘saga'
```

گرادیان کاهشی تصادفی از داده ی تصادفی استفاده می کند و در هر مرحله محاسبات کم است اما تعداد سعی و خطا زیاد است.
برای داده های زیاد مناسب است

روشهای حداقل کردن تابع هزینه

مختصات کاهشی (Coordinate Descent)



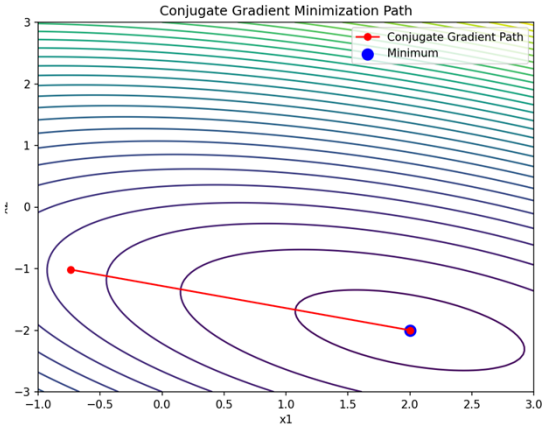
روش مشابه

```
solver=
'liblinear'
```

مشابه گرادیان کاهشی است ولی در هر گام تمام متغیر همزمان بروز نمی شوند در هر گام یک متغیر بروز می شود در حالیکه بقیه متغیرها ثابت فرض می شوند. برای مدل‌های خاصی (مانند رگرسیون لَسو) بسیار کارآمد است.

روشهای حداقل کردن تابع هزینه

گرادیان مزدوج (CG:Conjugate Gradient)

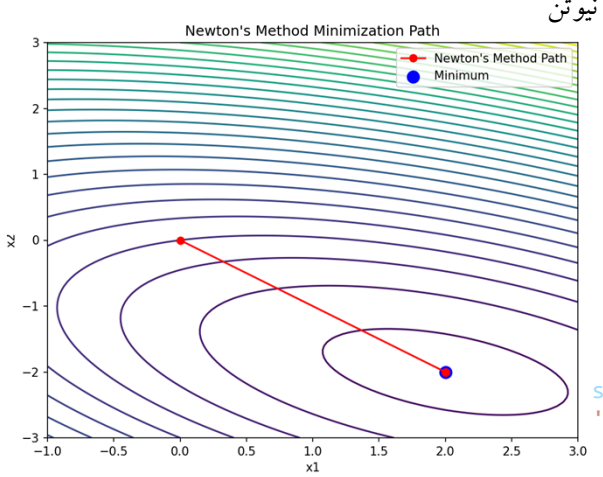



```
solver=
'newton-cg'
```

روش گرادیان مزدوج برای توابع کوادراتیک مناسب است. از جهت عمود بر تابع هزینه استفاده می کند و برای توابع کوداراتیک بسیار سریع است

روشهای حداقل کردن تابع هزینه

روش عددی نیوتن



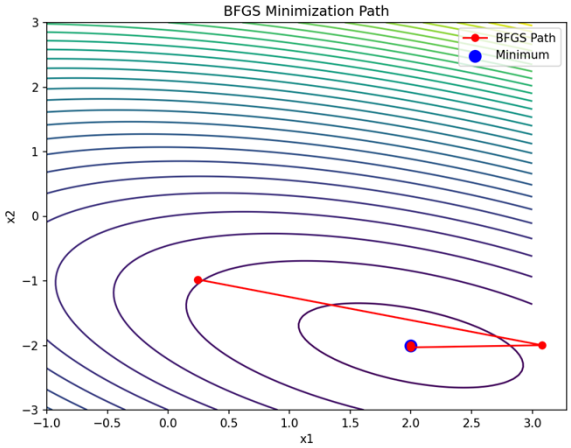
روش مشابه

```
solver=
'newton-cholesky'
```

روش عددی نیوتن برای داده های کم مناسب است. این روش در هر مرحله ماتریس مشتقات مرتبه دوم (هسیان) را حساب می کند و زمان زیادی می برد

روشهای حداقل کردن تابع هزینه

روش عددی شبه نیوتنی BFGS (Broyden–Fletcher–Goldfarb–Shanno)



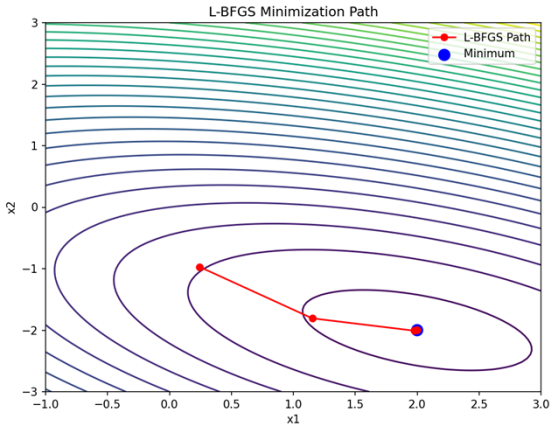
روش مشابه

```
solver=
'lbfgs'
```

روش عددی شبه نیوتنی BFGS برای داده های حجیم مناسب است. در این روش ماتریس مشتقات مرتبه دوم (هسیان) تخمین زده می شود و دقیق حساب نمی شود ممکن است تعداد سعی و خطا زیاد شود ولی زمان برا ی داده های حجیم کاهش می یابد.

روشهای حداقل کردن تابع هزینه

روش عددی شبه نیوتنی L- BFGS (Limited-memory BFGS)




```
solver='lbfgs'
```

روش عددی شبه نیوتنی L-BFGS مشابه روش BFGS بوده و برای داده های کم و متوسط مناسب است.
چون در این روش از حافظه کمتری استفاده می شود برای داده های با ابعاد زیاد نظیر طبقه بندی چند کلاسی مناسب است.

روشهای حداقل کردن تابع هزینه

گرادیان کاهشی (Gradient Descent)
گرادیان کاهشی دسته‌ای (BGD: Batch Gradient Descent)
گرادیان کاهشی تصادفی (SGD: Stochastic Gradient Descent)
مختصات کاهشی (Coordinate Descent)
گرادیان مزدوج (CG:Conjugate Gradient)
روش نیوتن
روش شبه نیوتنی BFGS (Broyden–Fletcher–Goldfarb–Shanno)
روش شبه نیوتنی L- BFGS (Limited-memory BFGS)

- روش‌های دیگری نیز وجود دارد که در اینجا توضیح داده نمی شود و می توانید مطالعه نموده و از آنها استفاد کنید مانند:
Mini-Batch Gradient Descent, Momentum-Based Methods,Adam (Adaptive Moment Estimation), Trust Region Methods, Trust Region Methods, Coordinate Descent

- در کتابخانه sklearn برای رگرسیونهای مختلف روشهای متفاوتی از حداقل کردن تابع هزینه گنجانده شده است. برای استفاده بهینه می توانید به توضیحات مربوطه مراجعه نمایید.

نمونه رگرسیون لجستیک

مثال : برای داده های زیر که احتمال حمله قلبی را بر اساس ضربان قلب نشان می دهد رگرسیون لجستیک را بدست آورید
احتمال حمله قلبی شخصی با ضربان قلب ۱۰۰ چقدر است

Heart Rate	Heart Attack
50	Y
50	N
50	N
50	N
70	N
70	Y
90	Y
90	Y
90	N
90	Y
90	Y

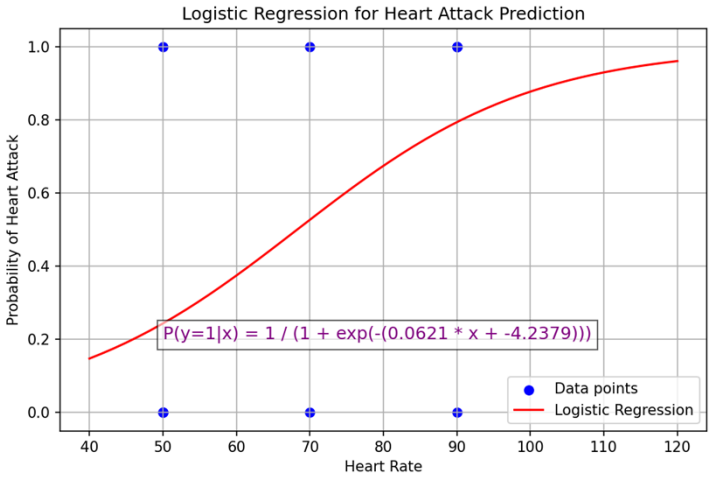
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

# Create the dataset
data = {
 'Heart Rate': [50, 50, 50, 50, 70, 70, 90, 90, 90, 90, 90],
 'Heart Attack': [1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1]
}
df = pd.DataFrame(data)

# Prepare the data
X = df[['Heart Rate']]
y = df['Heart Attack']
```

فایل: 7 regression logistic.py

نمونه رگرسیون لجستیک

```python
# Train the logistic regression model
model = LogisticRegression(solver='lbfgs') # Optimization method L-BFGS

model.fit(X, y)

# Generate values for prediction
x_values = np.linspace(40, 100, 200).reshape(-1, 1)
y_pred = model.predict_proba(x_values)[:, 1]
prediction_100 = model.predict_proba(np.array([[100]]))[0, 1]
print(f"heart attack probability for heart rate 100:{prediction_100:.2f}")


# Retrieve the coefficients for the logistic function formula
coef = model.coef_[0][0]          # Teta1
intercept = model.intercept_[0]  # Teta0

# Plotting
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color='blue', label='Data points')
plt.plot(x_values, y_pred, color='red', label='Logistic Regression')
plt.xlabel('Heart Rate')
plt.ylabel('Probability of Heart Attack')
plt.title('Logistic Regression for Heart Attack Prediction')
plt.legend()

# Display formula on the plot
formula_text = f'P(y=1|x) = 1 / (1 + exp(-({coef:.4f} * x + {intercept:.4f})))'
plt.text(50, 0.2, formula_text, fontsize=12, color='purple',
bbox=dict(facecolor='white', alpha=0.6))

plt.grid(True)
plt.show()
```

نمونه رگرسیون لجستیک



با رگرسیون لجستیک طبقه بندی به راحتی صورت می گیرد

```
heart attack probability for heart rate 100: 0.88
```

$$p(y=1|x) = p(x) = \frac{1}{1+e^{-(-4.2379+0.0621x)}} = 0.88$$

تمرین برنامه نویسی

تمرین هفتم :

یک برنامه به زبان پایتون بنویسید که یک فایل داده را خوانده و رگرسیون لجستیک برای یک ویژگی را بدست آورد.

۱- منحنی رگرسیون را روی داده ها رسم کنید

۲- از سه روش مختلف حداقل نمودن تابع هزینه استفاده کرده و زمان آنها را با هم مقایسه کنید.

۳- برای یک حالت با ویژگی مشخص، احتمال را تعیین کنید

ماتریس مشتقات

ماتریس مشتق مرتبه اول (برای آنهایی که بیشتر می خواهند بدانند)

Jacobian Matrix:

Suppose we have a vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m, \mathbf{f}(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) \end{bmatrix}$

Jacobian Matrix: $J(\mathbf{f}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$

Example

Consider a simple function $\mathbf{f}(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix}$ $f_1(x, y) = x^2 + y^2$ $f_2(x, y) = xy$

Jacobian Matrix: $J(\mathbf{f}) = \begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix}$

ماتریس مشتقات (برای آنهایی که بیشتر می خواهند بدانند)

ماتریس مشتق مرتبه دوم (برای آنهایی که بیشتر می خواهند بدانند)

Hessian Matrix:

Suppose we have a scalar-valued function $f(x_1, x_2, \ldots, x_n)$

Hessian Matrix:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Example

Consider a simple function $f(x, y) = x^2 + y^2$

Hessian Matrix:

$$H(f) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Adv. App. of AI and DT 41