3^{rd}-week-1

Linear Regression

Wang Jianfang
王建芳(in Chinese)

Linear Regression

- The least square method
- The GD

Agenda

- Concept
- Example
- Code

concept

- • Gradient
  - The direction vector that represents the fastest rate of cha nge of a function at a point (understood as the derivative/p artial derivative at that point)
- • Samples
  - actual observed data sets, including inputs and outputs (th e number of samples in this paper is represented by m and the element subscript i)

concept

- • Features
  - Input to the sample (the number of features in this paper is d enoted by *n* and the subscript *j* of the element)
- Hypothesis function
  - The function used to fit the sample is denoted as h_θ (X) (θ is the parameter vector and X is the eigenvector).
- Loss function
  - Used to evaluate the degree of model fitting, the goal of trai ning is to minimize the loss function, denoted as J(θ).

Linear hypothesis function

- Linear hypothesis function

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \sum_{j=0}^{n} \theta_j x_j$$

where X is the eigenvector, θ_j is the model paramete r, and x_j is the *j-th* element of the eigenvector (let x_0 =1).

Linear hypothesis function

- The classical square deviation loss function is as follows:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(X_i) - y_i)^2$$

- Where *m* is the number of samples, X_i is the *i-th* element of th e sample feature set (is a vector), y_i is the *i-th* element of the sa mple output, and $h_\theta(X_i)$ is the hypothesis function.
- Note: When the input has multiple features, a sample feature is a vector. Suppose the input to the function is an eigenvector in stead of a member of the eigenvector.

Gradient Descent

- The goal of the gradient descent method is to update the parameter θ of the hypothesis function h_θ in a reasonable way so that the loss function J(θ) is minimized for all samples.
- The steps of this reasonable approach are as follows:

• Design the hypothesis function, the loss function, and the initial val ue of all θ for the hypothesis function as a rule of thumb.

• Take the partial derivatives of all θ for the loss function: ∂J(θ)/∂θj

• The sample data is used to update the θ of the hypothesis function a nd the update formula is:

• $$\theta_j = \theta_j - \alpha \cdot \frac{\partial J}{\partial \theta_j}$$

• Where α is the update step size (adjust the sensitivity of parameters, the sensitivity is too high and easy to oscillate, and the sensitivity is too low an d convergence is slow).

Derivation Process

- The formula of linear hypothesis function is as follows (define d artificially according to experience or existing data) :

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \sum_{j=0}^{n} \theta_j x_j$$

- The formula of loss function is as follows (defined artificially according to experience or existing data) :

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(X_i) - y_i)^2$$

  - One half 1/2 is for ease of calculation (it cancels out when multiplied by the derivative of the square).

Derivation process

- The partial derivative of the loss function of a single sample to θ is calculated as follows:

$$
\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\frac{1}{2}(h_\theta(X)-y)^2 \\
&= 2\cdot\frac{1}{2}(h_\theta(X)-y)\cdot\frac{\partial}{\partial \theta_j}(h_\theta(X)-y) \\
&= (h_\theta(X)-y)\cdot\frac{\partial}{\partial \theta_j}(\theta_0 x_0+\cdots+\theta_j x_j+\cdots+\theta_n x_n) \\
&= (h_\theta(X)-y)\cdot x_j
\end{aligned}
$$

Derivation process

- For all samples, the partial derivative of the loss function with respect to is equal to the sum of all individual samples. The formula is as follows:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(X_i) - y_i) \cdot X_{ij}$$

  - where X_{ij} represents the *j-th* feature of the *i-th* sample.
- The formula for updating the hypothetical function θ is as follows (the initial value θ should be given empirically):

$$\theta_j = \theta_j - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(h_\theta(X_i) - y_i) \cdot X_{ij}$$

- Repeat the above procedure with all samples as input until the value of the loss function meets the requirement.

Example

1. So far, let's summarize the steps to solve the linear regression problem:
2. Initialize a model, such as h =2+3x, our initial parameters are θ_0=2, θ_1=3.
3. Given a sample pair, such as (2,4), the predicted value is obtained by plugg3.ing it into the model, i.e., h=2+3*2=8.
4. If you plug it into the loss function formula, you get the loss value, which is J=1/2*(8-4)^2=8.
5. Substitute into the partial layer number formula, and find the partial derivative of θ_0,θ_1:

$$\frac{\partial}{\partial\theta_0}J(\theta) = (h_\theta(x) - y) = 2 + 32 - 4 = 4$$

$$\frac{\partial}{\partial\theta_1}J(\theta) = (h_\theta(x) - y)x_1 = (2 + 32 - 4) * 2 = 8$$

Example

6.Assuming that the learning rate is 0.1, then the gradient descent formula of the feed can be obtained as follows:

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta) = 2 - 0.1 \times 4 = 1.6$$
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta) = 3 - 0.1 \times 8 = 2.2$$

7.Gainning a new parameter, θ_0=1.6,θ_1=2.2, So the new model is: h=1.6+2.2x,The new prediction is h=1.6+2.2*2=6, Calculate the value of the loss function again:J=1/2*(6-4)^2=2.

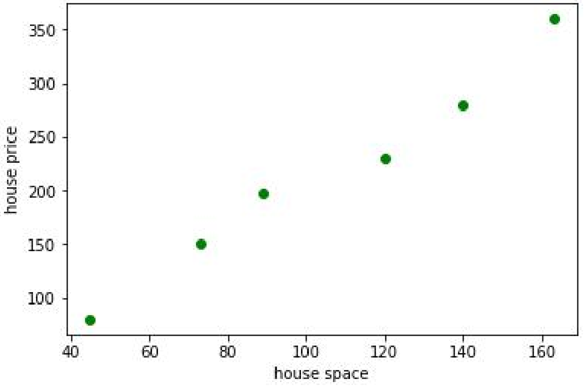
8.The loss value 2 obtained by the new model is 6 less than the generation value 8 obtained by the old model. The smaller the loss value is, the better the match between our model and the training set is. Therefore, through continuous gradient descent, we can get the model *h* which is most suitable for the training data.

Python

- Here is an example of a home price assessment using the gradient desc ent method.
- We know that the price of a house is related to many factors (such as a rea, book of rooms, location, etc.), and each factor is called a feature.
- Assuming that the area of the house is the only feature (ignoring other s for the purpose of simplifying the model), the known data are as foll ows:
- Room area: 45, 73, 89, 120, 140, 163 m^2
- Housing price: 80, 150, 198, 230, 280, 360 (RMB Yuan)
- From this data, you can use the Python code below to make a three-po int diagram of area and price.

Python

- %matplotlib inline
- import numpy as np
- import matplotlib.pyplot as plt
- spaces = [45, 73, 89, 120, 140, 163]
- prices = [80, 150, 198, 230, 280, 360]
- spaces, prices = np.array(spaces), np.array(prices)
- plt.scatter(spaces, prices, c='g')
- plt.xlabel('house space')
- plt.ylabel('house price')
- plt.show()
- ## A scatter plot showing the size and price of a house



Python

- According to the steps of the gradient descent method, we need to first give the hypothesis function h_θ and the loss function J(θ), and the initial value θ.
- The hypothetical function of house area and price is: h_θ (x)= θ_0+ θ_1x (a feature value).
- The loss function uses the mean variance function:

$$J(\theta) = \frac{1}{2*6}\sum_{i=1}^{6}(h_\theta(X_i) - y_i)^2$$

Python

- If the update step size is 0.00005, the update formula is:

$$\theta_j = \theta_j - 0.00005 \cdot \frac{1}{6} \sum_{i=1}^{6} (h_\theta(X_i) - y_i) \cdot X_{ij}$$

- where θ_j contains θ_0 and θ_1, $X_{i0} = 1$.
- Note: If the step size is not selected correctly, the result of the t heta parameter update will not be correct.
- Calculate θ and draw the h_θ function using the following Pytho n code:

Python

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

spaces = [45, 73, 89, 120, 140, 163]
prices = [80, 150, 198, 230, 280, 360]
spaces, prices = np.array(spaces), np.array(prices)
plt.scatter(spaces, prices, c='g')
plt.xlabel('house space')
plt.ylabel('house price')
plt.show()

## A scatter plot showing the size and price of a house
```

Python

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
## the initial value theta
theta0 = 0
theta1 = 0
## If the step size is not selected correctly, the result of the theta parameter update will be incorrect
step = 0.00005
x_i0 = np.ones((len(spaces)))
# hypothesis function
def h(x) :
  return theta0 + theta1 * x
# loss function
def calc_error() :
  return np.sum(np.power((h(spaces) - prices),2)) / 6
# Partial derivative of the loss function( theta 0)
def calc_delta0() :
  return step * np.sum((h(spaces) - prices) * x_i0) / 6
# Partial derivative of the loss function( theta 1)
def calc_delta1() :
  return step * np.sum((h(spaces) - prices) * spaces) / 6
# To iterate over the value of Theta and calculate the error, stop condition is
#  1. The error is less than some value
#  2. Loop count control
k = 0
while True :
  delta0 = calc_delta0()
  delta1 = calc_delta1()
  theta0 = theta0 - delta0
  theta1 = theta1 - delta1
  error = calc_error()
  # print("delta [%f, %f], theta [%f, %f], error %f" % (delta0, delta1, theta0, theta1, error))
  k = k + 1
  if (k > 10 or error < 200) :
    break
print(" h(x) = %f + %f * x" % (theta0, theta1))

# The price calculated using the hypothesis function is used to draw the fitting curve
y_out = h(spaces)
plt.scatter(spaces, prices, c='g')
plt.plot(spaces, y_out, c='b')
plt.xlabel('house space')
plt.ylabel('house price')
plt.show()
# The green dots are the area and price data
# The blue line is the curve that we fitted using the gradient descent method
```

Python

- By running the code above, we can see that the result of the gr adient descent method is related to the initial value of θ and th e step size.
- We need to empirically give θ and step sizes based on the char acteristics of the system.

Homework

- Download iris datasets
- and read and test the code.

Questions and Comments?

Thank you!!