Lecture 5
Linear Models

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2023

Outline

- Linear model
  - Linear regression
  - Logistic regression
  - Softmax regression

Linear regression

- Our goal in linear regression is to predict a target continuous value y from a vector of input values $\boldsymbol{x} \in R^d$; we use a linear function h as the model
- At the training stage, we aim to find $h(\boldsymbol{x})$ so that we have $h(\boldsymbol{x}_i) \approx y_i$ for each training sample $(\boldsymbol{x}_i, y_i)$
- We suppose that h is a linear function, so

$$h_{(\boldsymbol{\theta},b)}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} + b, \boldsymbol{\theta} \in R^{d \times 1}$$

Rewrite it,

$$\boldsymbol{\theta}' = \begin{pmatrix} \boldsymbol{\theta} \\ b \end{pmatrix}, \boldsymbol{x}' = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix}$$

$$\boldsymbol{\theta}^T \boldsymbol{x} + b = \boldsymbol{\theta}'^T \boldsymbol{x}' \equiv h_{\boldsymbol{\theta}'}(\boldsymbol{x}')$$

Later, we simply use $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}, \boldsymbol{\theta} \in R^{(d+1)\times 1}, \boldsymbol{x} \in R^{(d+1)\times 1}$

Linear regression

- Then, our task is to find a choice of $\boldsymbol{\theta}$ so that $h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ is as close as possible to y_i

The cost function can be written as,

$$L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\boldsymbol{x}_i - y_i\right)^2$$

Then, the task at the training stage is to find

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\boldsymbol{x}_i - y_i\right)^2$$

For this special case, it has a closed-form optimal solution

Here we use a more general method, **gradient descent** method

Linear regression

- Gradient descent
  - It is a first-order optimization algorithm
  - To find a local minimum of a function, one takes steps proportional to the negative of the gradient of the function at the current point
  - One starts with a guess $\boldsymbol{\theta}_0$ for a local minimum of $L(\boldsymbol{\theta})$ and considers the sequence such that

$$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})_{|\boldsymbol{\theta}=\boldsymbol{\theta}_{n-1}}$$

where α is called as **learning rate**

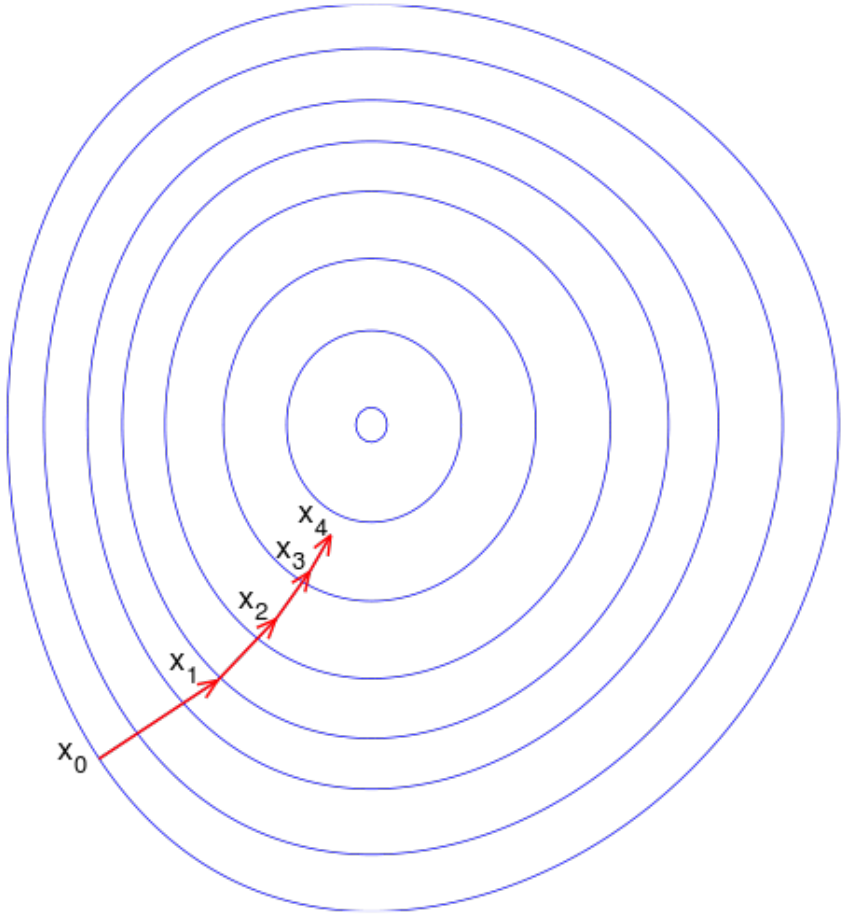
Why will the function value decrease when we move a small step along the opposite direction of the gradient?

Linear regression

- Gradient descent

Linear regression

- Gradient descent



Linear regression

- Gradient descent

Repeat until convergence ($L(\boldsymbol{\theta})$ will not reduce anymore)
{

$$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})_{|\boldsymbol{\theta}=\boldsymbol{\theta}_{n-1}}$$

}

GD is a general optimization solution; for a specific problem, the key step is how to compute gradient

Linear regression

- Gradient of the cost function of linear regression

$$L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i\right)^2$$

The gradient is,

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_{d+1}} \end{bmatrix} \quad \text{where,} \quad \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} = \sum_{i=1}^{m}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)x_{ij}$$

Linear regression

- Some variants of gradient descent
  - The ordinary gradient descent algorithm looks at every sample in the **entire** training set on every step; it is also called as **batch gradient descent**
  - **Stochastic gradient descent (SGD)** repeatedly run through the training set, and each time when we encounter a training sample, we update the parameters according to the gradient of the error w.r.t that single training sample only

Repeat until convergence
{
  for i = 1 to m (m is the number of training samples)
  {
    $$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \alpha\left(\boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_i - y_i\right)\boldsymbol{x}_i$$
  }
}

Linear regression

- Some variants of gradient descent
  - The ordinary gradient descent algorithm looks at every sample in the **entire** training set on every step; it is also called as **batch gradient descent**
  - **Stochastic gradient descent (SGD)** repeatedly run through the training set, and each time when we encounter a training sample, we update the parameters according to the gradient of the error w.r.t that single training sample only
  - **Minibatch SGD**: it works identically to SGD, except that it uses more than one training samples to make each estimate of the gradient

Linear regression

- More concepts
  - *m* Training samples can be divided into *N* minibatches
  - When the training sweeps all the batches, we say we complete one **epoch** of training process; for a typical training process, several epochs are usually required

epochs = 10;
numMiniBatches = *N*;
while *epochIndex*< *epochs* && *not convergent*
{
  reshuffle minibatches
  for *n* = 1 to *numMiniBatches*
  {
    //update the model parameters based on minibatch $\mathcal{B}_n$
    $\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \alpha \mathbf{g}_n\left(\boldsymbol{\theta}_{n-1}\right)$
  }
}

$$\mathbf{g}_n\left(\boldsymbol{\theta}_{n-1}\right) = \frac{dL\left(\mathcal{B}_n; \boldsymbol{\theta}_{n-1}\right)}{d\boldsymbol{\theta}_{n-1}}$$

Outline

- Linear model
  - Linear regression
  - Logistic regression
  - Softmax regression

Logistic regression

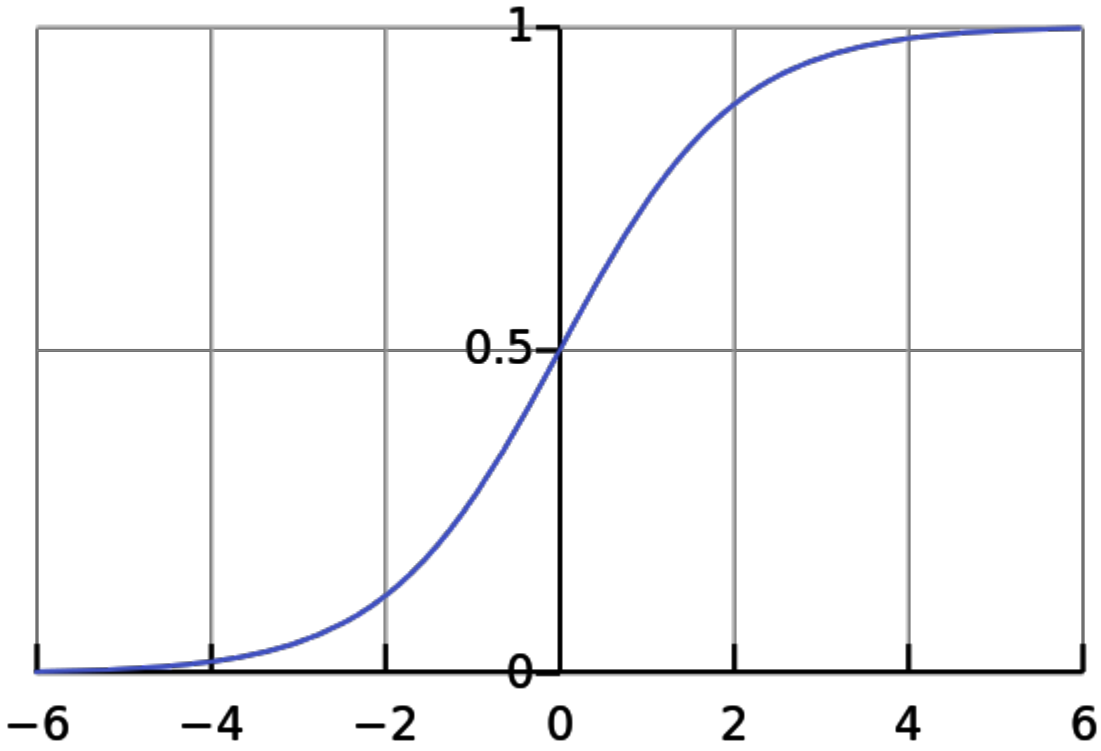
- Logistic regression is used for binary classification
- It squeezes the linear regression $\boldsymbol{\theta}^T \boldsymbol{x}$ into the range (0, 1) ; thus the prediction result can be interpreted as probability
- At the testing stage

  The probability that the testing sample $\boldsymbol{x}$ is positive is represented as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \dfrac{1}{1+\exp(-\boldsymbol{\theta}^T \boldsymbol{x})}$

  The probability that the testing sample $\boldsymbol{x}$ is negative is represented as $1\text{-}h_{\boldsymbol{\theta}}(\boldsymbol{x})$

Function $\sigma(z) = \dfrac{1}{1+\exp(-z)}$ is called as **sigmoid** or **logistic** function

Logistic regression



The shape of sigmoid function

One property of the sigmoid function

$$\sigma'(z) = \sigma(z)(1-\sigma(z))$$

Can you verify?

Logistic regression

- The hypothesis model can be written neatly as

$$P(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \left(h_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^{y} \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^{1-y}, y \in \{0,1\}$$

- Our goal is to search for a value $\boldsymbol{\theta}$ so that $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ is large when $\boldsymbol{x}$ belongs to "1" class and small when $\boldsymbol{x}$ belongs to "0" class

Thus, given a training set with binary labels $\{(\boldsymbol{x}_i, y_i) : | \, i = 1, \ldots, m, y_i \in \{0,1\}\}$, we want to maximize,

$$\prod_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^{y_i} \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^{1-y_i}$$

Equivalent to maximize,

$$\sum_{i=1}^{m} y_i \log\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) + (1 - y_i) \log\left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)$$

Logistic regression

- Thus, the cost function for the logistic regression is (we want to minimize),

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{m} y_i \log\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)$$

To solve it with gradient descent, gradient needs to be computed,

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \sum_{i=1}^{m} \boldsymbol{x}_i \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)$$

Assignment!

Outline

- Linear model
  - Linear regression
  - Logistic regression
  - Softmax regression

Softmax regression

- Softmax operation
  - It squashes a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range (0, 1). The function is given by,

$$\sigma(\mathbf{z})_j = \frac{\exp(\mathbf{z}_j)}{\sum_{k=1}^{K} \exp(\mathbf{z}_k)}$$

  - Since the components of the vector $\sigma(\mathbf{z})$ sum to one and are all strictly between 0 and 1, they represent a categorical probability distribution

Softmax regression

- For multiclass classification, given a test input $\boldsymbol{x}$, we want our hypothesis to estimate $p(y = k \mid \boldsymbol{x})$ for each value $k=1,2,\ldots,K$

Softmax regression

- The hypothesis should output a K-dimensional vector giving us K estimated probabilities. It takes the form,

$$h_\phi(\boldsymbol{x}) = \begin{bmatrix} p(y=1 \mid \boldsymbol{x};\phi) \\ p(y=2 \mid \boldsymbol{x};\phi) \\ \vdots \\ p(y=K \mid \boldsymbol{x};\phi) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp\left((\boldsymbol{\theta}_j)^T \boldsymbol{x}\right)} \begin{bmatrix} \exp\left((\boldsymbol{\theta}_1)^T \boldsymbol{x}\right) \\ \exp\left((\boldsymbol{\theta}_2)^T \boldsymbol{x}\right) \\ \vdots \\ \exp\left((\boldsymbol{\theta}_K)^T \boldsymbol{x}\right) \end{bmatrix}$$

where $\phi = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K] \in R^{(d+1)\times K}$

Softmax regression

- In softmax regression, for each training sample we have,

$$p\left(y_i = k \mid \boldsymbol{x}_i; \phi\right) = \frac{\exp\left(\left(\boldsymbol{\theta}_k\right)^T \boldsymbol{x}_i\right)}{\sum_{j=1}^{K} \exp\left(\left(\boldsymbol{\theta}_j\right)^T \boldsymbol{x}_i\right)}$$

At the training stage, we want to maximize $p\left(y_i = k \mid \boldsymbol{x}_i; \phi\right)$ for each training sample for the correct label k

Softmax regression

- Cost function for softmax regression

$$L(\phi) = -\sum_{i=1}^{m}\sum_{k=1}^{K} 1\{y_i = k\} \log \frac{\exp\left((\boldsymbol{\theta}_k)^T \boldsymbol{x}_i\right)}{\sum_{j=1}^{K} \exp\left((\boldsymbol{\theta}_j)^T \boldsymbol{x}_i\right)}$$

where 1{.} is an indicator function

- Gradient of the cost function

$$\nabla_{\theta_k} L(\phi) = -\sum_{i=1}^{m} \left[\boldsymbol{x}_i \left(1\{y_i = k\} - p\left(y_i = k \mid \boldsymbol{x}_i; \phi\right)\right)\right]$$

Can you verify?

Cross entropy

- After the softmax operation, the output vector can be regarded as a discrete probability density function
- For multiclass classification, the ground-truth label for a training sample is usually represented in one-hot form, which can also be regarded as a density function

  For example, we have 10 classes, and the ith training sample belongs to class 7, then $y_i = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$

- Thus, at the training stage, we want to minimize

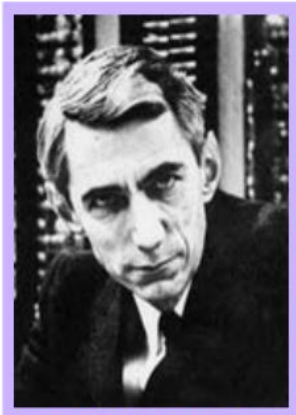
$$\sum_i dist(h(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

How to define *dist*? Cross entroy is a common choice

Cross entropy

- Information entropy is defined as the average amount of information produced by a probabilistic stochastic source of data $H(X) = -\sum_i p(x_i) \log p(x_i)$

  where X is a random variable and x_i is the event in the sample space represented by X



Claude Shannon
1916–2001

After graduating from Michigan and MIT, Shannon joined the AT&T Bell Telephone laboratories in 1941. His paper 'A Mathematical Theory of Communication' published in the *Bell System Technical Journal* in 1948 laid the foundations for modern information theory. This paper introduced the word 'bit', and his concept that information could be sent as a stream of 1s and 0s paved the way for the communications revolution. It is said that von Neumann recommended to Shannon that he use the term entropy, not only because of its similarity to the quantity used in physics, but also because "nobody knows what entropy really is, so in any discussion you will always have an advantage".

Cross entropy

- Information entropy is defined as the average amount of information produced by a probabilistic stochastic source of data $H(X) = -\sum_i p(x_i) \log p(x_i)$
- Cross entropy can measure the difference between two distributions

$$H(p, q) = -\sum_i p(x_i) \log q(x_i)$$

where p is the ground-truth, q is the prediction result, and x_i is the class index

- For multiclass classification, the last layer usually is a softmax layer and the loss is the 'cross entropy'

Cross entropy

Example:

Suppose that the label of one sample is [1 0 0]

For model 1, the output of the last softmax layer is [0.5 0.4 0.1]

Its cross entropy is (base 10),

$$H(p,q) = -\sum_i p(x_i)\log q(x_i) = -\left(1*\log 0.5 + 0*\log 0.4 + 0*\log 0.1\right) \approx 0.3$$

For model 2, the output of the last softmax layer is [0.8 0.1 0.1]

$$H(p,q) = -\sum_i p(x_i)\log q(x_i) = -\left(1*\log 0.8 + 0*\log 0.1 + 0*\log 0.1\right) \approx 0.1$$

Model 2 is better


Q+A