Pattern Recognition

Lecture 14. Logistic Regression

Dr. Shanshan ZHAO & Hong Seng GAN

School of AI and Advanced Computing
Xi'an Jiaotong-Liverpool University

Academic Year 2023-2024

Table of Contents

Outline

1. Introduction
2. Logistic Regression
3. Learning in Logistic Regression
4. Multinomial Logistic Regression

Linear Regression

- house prices
- stock
- food production

Predicted value is **continuous**

Classification

- spam email or not
- customer's intention
- car models

Predicted value is **categorical**

Classification Types

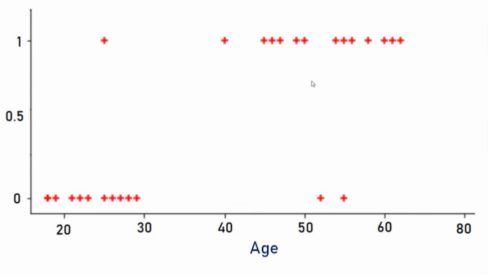
- **Binary Classification**
  e.g., Will the customer buy life insurance?
  1. yes
  2. no
- **Multiple Class calssification**
  e.g., dog breeds
  1. Golden Retriever
  2. Huskie
  3. Corgi

Logistic Regression

Example

age	have_insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0
27	0
29	0
49	1

(a)



(b)

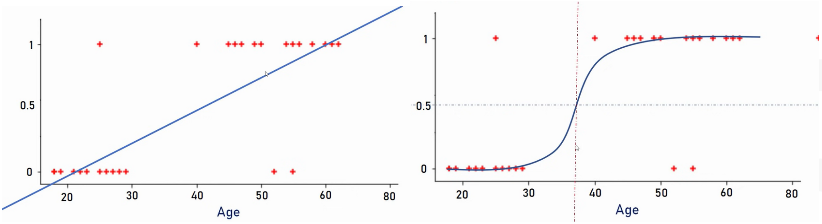
Introduction

- **Logistic Regression** is one the techniques for classification.
- Logistic regression can be used to classify an observation into one of **two classes** (like 'have insurance' and 'no insurance'), or into one of many classes.

Problem Statement

- Consider a single **input** observation x, which we will represent by a vector of features $[x_1, x_2, ..., x_n]$.
- The classifier **output** y can be 1 (meaning the observation is a member of the class) or 0 (the observation is not a member of the class).
- We want to know the probability $P(y = 1|x)$ that this observation is a member of the class.

Logistic Regression



Outline

Logistic Regression

Logistic regression solves this task by learning, from a training set, a vector of weights and a bias term. Each weight w_i is a real number, and is associated with one of the input features x_i.

$$z = \sum_{I=1}^{n} w_i x_i + b$$

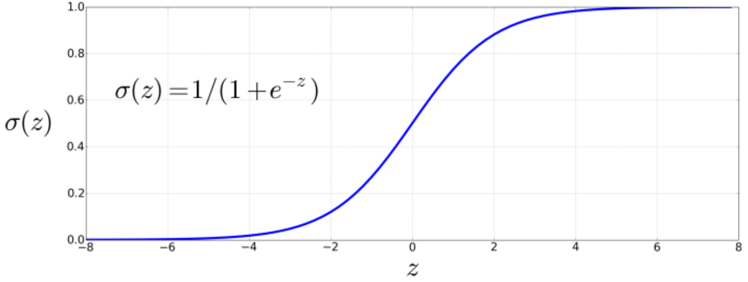
The sums can be represented with the dot product notation from linear algebra. Thus

$$z = w \cdot x + b$$

Sigmoid / Logit Function

$$z = w \cdot x + b$$

The **sigmoid function** $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $[0, 1]$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

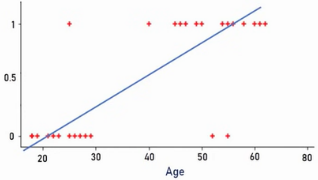


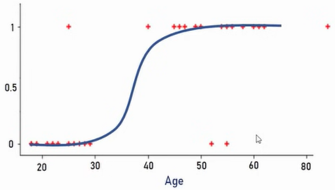
Logistic Regression

Example

$$y = m * x + b$$

$$y = \frac{1}{1 + e^{-(m*x+b)}}$$





Logistic Regression

If we apply the sigmoid to the sum of the weighted features, we get a number between 0 and 1. To make it a probability, we just need to make sure that the two cases, $P(y = 1)$ and $P(y = 0)$, sum to 1. We can do this as follows:

$$
\begin{aligned}
P(y=1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1+\exp(-(w \cdot x + b))} \\
P(y=0) &= 1 - \sigma(w \cdot x + b) \\
&= 1 - \frac{1}{1+\exp(-(w \cdot x + b))} \\
&= \frac{\exp(-(w \cdot x + b))}{1+\exp(-(w \cdot x + b))}
\end{aligned}
$$

Logistic Regression

The sigmoid function has the property

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

Can you prove it?⇑

so we could also have expressed $P(y = 0)$as $\sigma(-(w \cdot x + b))$.

- Now we have an algorithm that given an instance x computes the probability $P(y = 1|x)$.
- How do we make a decision?

Logistic Regression

Decision Boundary

$$\text{decision}(x) = \begin{cases} 1, \text{if } P(y=1|x) > 0.5 \\ 0, \text{otherwise} \end{cases}$$

Example: sentiment classification

Let's have an example. Suppose we are doing binary sentiment classification on movie review text, and we would like to know whether to assign the sentiment class $+$ or $-$ to a review document *doc*. We'll represent each input observation by the 6 features $x_1 \ldots x_6$ of the input shown in the following table; Fig. 5.2 shows the features in a sample mini test document.

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon words $\in$ doc)	3
x_2	count(negative lexicon words $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\log(\text{word count of doc})$	$\ln(66) = 4.19$

Example: sentiment classification(cont.)

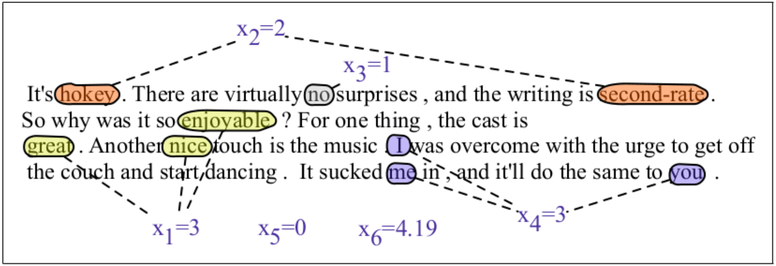


Figure 5.2 A sample mini test document showing the extracted features in the vector *x*.

Example: sentiment classification

Assuming 6 weights correspoinding to the 6 features are $[2.5, -5.0, -1.2, -0.5, 2.0, 0.7]$ while $b = 0.1$.
$P(+|x)$ and $P(-|x)$ can be computed using the equation

$$\begin{aligned}
P(+|x) =& P(y = 1|x) = \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, -0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\
&= \sigma(0.833) \\
&= 0.70 \\
P(-|x) =& P(y = 0|x) = 1 - \sigma(w \cdot x + b) \\
&= 0.3
\end{aligned}$$

Outline

Learning in Logistic Regression

- How are the parameters of the model, the weights w and bias b, learned?
- Logistic regression is an instance of supervised classification in which we know the correct label y (either 0 or 1) for each observation x.
- What the system produces $\hat{y}$, the system's estimate of the true y. We want to learn parameters (meaning w and b) that make $\hat{y}$ for each training observation as close as possible to the true y.
- **loss** or **cost function**

Cross Entropy Loss Function

We need a loss function that expresses, for an observation x, how **close** the classifier output ($\hat{y} = \sigma(w \cdot x + b)$) is to the correct output (y, which is 0 or 1).
We'll call this:

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y$$

cross-entropy loss ?

Cross Entropy Loss Function

- We'd like to learn weights that maximize the probability of the correct label $P(y|x)$.
- Since there are only two discrete outcomes (1 or 0), this is a Bernoulli distribution, and we can express the probability $P(y|x)$ that our classifier produces for one observation as the following

$$P(y|x) = \hat{y}^y (1-\hat{y})^{1-y}$$

Take the log of both side,

$$\begin{aligned} log(P(y|x) &= \log[\hat{y}^y (1-\hat{y})^{1-y}] \\ &= y \log(\hat{y}) + (1-y)\log(1-\hat{y}) \end{aligned}$$

It describes a log likelihood that should be maximized.

Cross Entropy Loss Function

In order to turn this into loss function (something that we need to minimize), we'll just flip the sign. The result is the cross-entropy loss L_{CE}:

$$L_{CE}(\hat{y}, y) = -\log(P(y|x)) = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$$

Cross Entropy Loss Function

- cross-entropy loss L_{CE}:

$$L_{CE}(\hat{y}, y) = -\log(P(y|x)) = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$$

- plug in the definition of $\hat{y} = \sigma(w \cdot x + b)$

$$L_{CE}(\hat{y}, y) = -[y\log(\sigma(w \cdot x + b)) + (1-y)\log(1 - \sigma(w \cdot x + b))] \quad (1)$$

Cross Entropy Loss Function

- Let's see if this loss function does the right thing the the example.
- We want the loss to be smaller if the model's estimate is close to correct, and bigger if the model is confused.
- So first let's suppose the correct label for the sentiment example is positive, i.e., $y = 1$.

Cross Entropy Loss Function

In this case our model is doing well, since it indeed gave the example a higher probability of being positive (.70) than negative (.30). If we plug $\sigma(wx + b) = .70$ and $y = 1$ into the equation (1), the right side of the equation drops out, leading to the following loss :
(we'll use log to mean natural log when the base is not specified)

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\ &= -[\log \sigma(w \cdot x + b)] \\ &= -\log(.70) \\ &= .36 \end{aligned}$$

Cross Entropy Loss Function

- By contrast, let's pretend instead that the example was actually negative, i.e., $y = 0$.
- In this case our model is confused and we'd want the loss to be higher.

Now if we plug $y = 0$ and $1 - \sigma(w \cdot x + b) = .30$ into equation (1),the left side of the equation drops out:

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\ &= -[\log(1 - \sigma(w \cdot x + b))] \\ &= -\log(.30) \\ &= 1.2 \end{aligned}$$

The loss for the first classifier is less than the loss for the second classifier.

Gradient Descent

- Our goal with gradient descent is to find the optimal weights: minimize the loss function we've defined for the model.
- In equation below, we'll explicitly represent the fact that the loss function L is parameterized by the weights, which we'll refer to in machine learning in general as θ (in the case of logistic regression $\theta = w, b$).
- So the goal is to find the set of weights which minimizes the loss function, averaged over all examples:

$$\hat{\theta} = \arg_{\theta} \min \frac{1}{m} \sum_{i=1}^{m} L_{CE}(f(x^{(i)}; \theta), y^{(i)})$$

For logistic regression, this loss function is conveniently convex.
A convex function has just one minimum; there are no local minima to get stuck in, so gradient descent starting from any point is guaranteed to find the minimum.

Gradient Descent

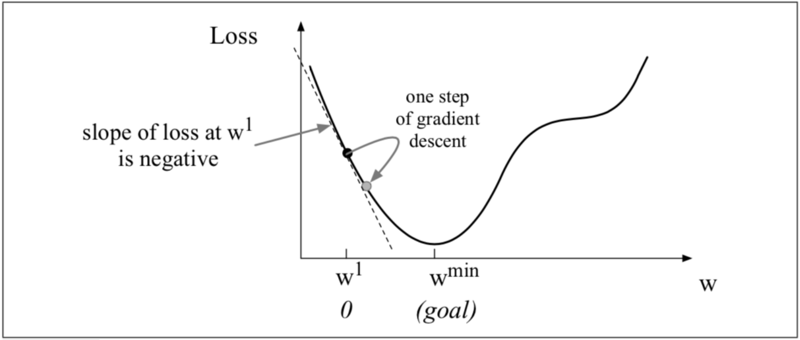


Figure 5.3 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 at the second step, and so on.

Outline

- To deal with multi-class problem, we use multinomial logistic regression, also called softmax regression.
- In multinomial logistic regression the target y is a variable that ranges over more than two classes;
- we want to know the probability of y being in each potential class $c \in C$, $P(y = c|x)$.

The multinomial logistic classifier uses a generalization of the sigmoid, called **the softmax function**, to compute the probability $P(y = c|x)$.

For a vector z of dimensionality k, the softmax is defined as:

$$softmax(z) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}, 1 \leq i \leq k$$

The softmax of an input vector $z = [z_1, z_2, \cdots, z_k]$ is thus a vector itself:

$$softmax(z) = \left[\frac{\exp(z_1)}{\sum_{j=1}^{k} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{j=1}^{k} \exp(z_i)}, \cdots, \frac{\exp(z_k)}{\sum_{j=1}^{k} \exp(z_i)} \right]$$

$$softmax(z) = \left[\frac{\exp(z_1)}{\sum_{j=1}^{k} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{j=1}^{k} \exp(z_i)}, \cdots, \frac{\exp(z_k)}{\sum_{j=1}^{k} \exp(z_i)} \right]$$

The denominator $\sum_{j=1}^{k} \exp(z_i)$ is used to normalize all the values into probabilities.

For example given a vector:

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

the resulting (rounded) $softmax(z)$ is

$$[0.055, 0.090, 0.006, 0.099, 0.74, 0.010]$$

- like the sigmoid, the input to the softmax will be the dot product between a weight vector w and an input vector x (plus a bias).
- difference: we need to separate weight vectors (and bias) for each of the K classes.

$$P(y = c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

Like the sigmoid, the softmax has the property of squashing values toward 0 or 1. Thus if one of the inputs is larger than the others, it will tend to push its probability toward 1, and suppress the probabilities of the smaller inputs.

Loss function for Multinomial Logistic Regression

The loss function for multinominal logistic regression

$$\begin{aligned} L_{CE} &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\ &= -\sum_{k=1}^{K} y_k \log \hat{p}(y = k|x) \end{aligned}$$

The vector y is a **one-hot vector**. Thus,

$$\begin{aligned} L_{CE} = (\hat{y}, y) &= -\sum_{k=1}^{K} \mathbb{1}\{y = k\} \log \hat{p}(y = k|x) \\ &= -\sum_{k=1}^{K} \mathbb{1}\{y = k\} \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)} \end{aligned}$$

where $\mathbb{1}\{y = k\}$ is the indicator function, which evaluates to 1 if the condition in the brackets is true and to 0 otherwise.

Thank You !

Q & A