Chapter 4 Classification
– Math 313 Statistics for Data Science

Guangliang Chen

Associate Professor

cheng@hope.edu

Hope College, Fall 2023

Presentation Overview

Section 4.1 An overview of classification

We have just covered in Chapter 3 the linear regression problem which deals with a continuous (or quantitative or numerical) response.

In other cases when the response is discrete (or qualitative or categorical), the task of predicting the response is called classification.

There are many classifiers that one might use to predict a categorical response. In this lecture we cover the following:

- logistic regression,
- linear discriminant analysis,
- quadratic discriminant analysis,
- naive Bayes, and
- K nearest neighbors (KNN)

Section 4.2 Why not linear regression?

When a response variable has three (or more) levels, say cat, dog, bird, simply encoding the levels by integers like 1, 2, 3 does not work well for the following reasons:

- The ordering is arbitrary and might be very wrong;
- Even when the introduced ordering might make sense, the gap between them is assumed to be equal (not necessarily true).

In the case of a binary qualitative response (with two levels), we can use a 0/1 valued dummy variable to convert it to quantitative. We can then attempt to perform regression with the 0/1 coding of the response, but need to be cautious about the interpretation.

Section 4.3 Logistic regression

The regression approach to binary classification

We just said that in the case of binary response, classification can be approached as a regression problem with 0/1 coding:

$$\underbrace{y}_{\text{binary response}} \quad = \quad f\,(\underbrace{x_1, \ldots, x_d}_{\text{predictors}}).$$

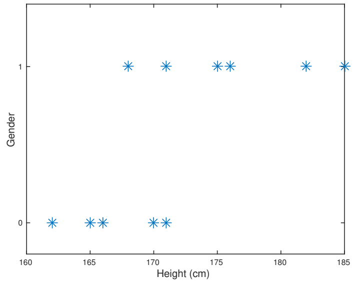
where $f: \mathbb{R}^d \mapsto \{0, 1\}$.

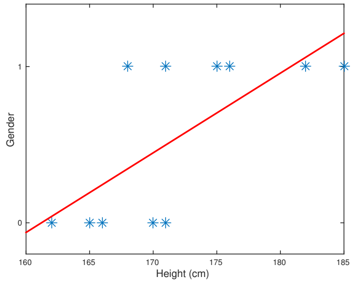
To explain the ideas, we start with a binary classification problem with only one predictor:

$$\underbrace{y}_{\text{binary response}} \quad = \quad f(\underbrace{x}_{\text{1 predictor}}).$$

Motivating example

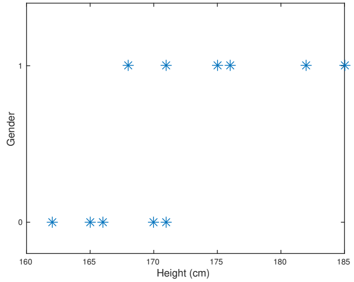
Consider a specific example where x represents a person's height while y denotes the person's sex (0 = Female, 1 = Male).

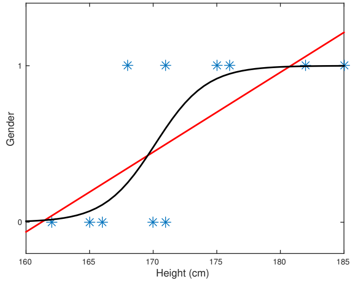




Simple linear regression is not appropriate in this case.

Consider a specific example where x represents a person's height while y denotes the person's sex (0 = Female, 1 = Male).



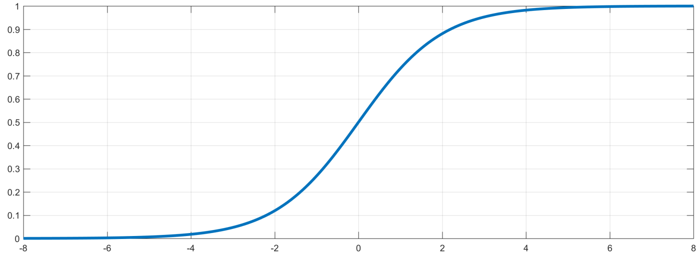


A better option is to use an S-shaped curve to fit the data.

Which functions have such shapes?

An example of such a curve is the **sigmoid** function:

$$g(z) = \frac{1}{1+e^{-z}}, \quad -\infty < z < \infty$$



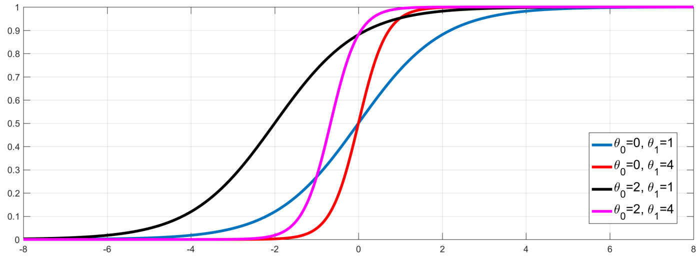
Can you think of another function that has a similar shape?

Making the logistic function more flexible

We generalize the basic sigmoid function to a **location-scale family** of similar curves (to provide abundant flexibility):

$$g(\theta_0 + \theta_1 x) = \frac{1}{1 + e^{-(\theta_0+\theta_1 x)}}$$

where $\theta_0 \in \mathbb{R}$ is a location parameter and $\theta_1 > 0$ is a scale parameter.



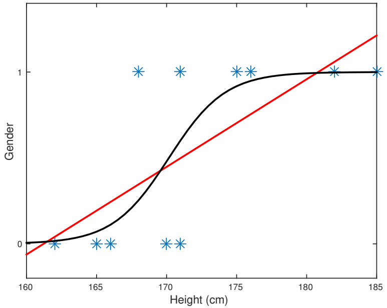
The logistic regression problem

Once we fix the template function $g(z)$ (with flexible choices of location and scale), the logistic regression problem reduces to parameter estimation based on a set of examples.

Problem. Given training data $\{(x_i, y_i) \mid 1 \le i \le n\}$, with $y_i = 0, 1$, find θ_0, θ_1 such that the curve

$$y = g(\theta_0 + \theta_1 x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

fits the data “as closely as possible”.



How to define the optimality

One naive choice is to use the least squares criterion based on the square loss $\ell(y, \hat{y}) = (y - \hat{y})^2$:

$$\min_{\theta_0, \theta_1} \sum_{i=1}^{n} \underbrace{(y_i - \hat{y}_i)^2}_{\text{square loss}}, \qquad \text{where} \quad \hat{y}_i = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

It does not work well in this binary setting due to two reasons:

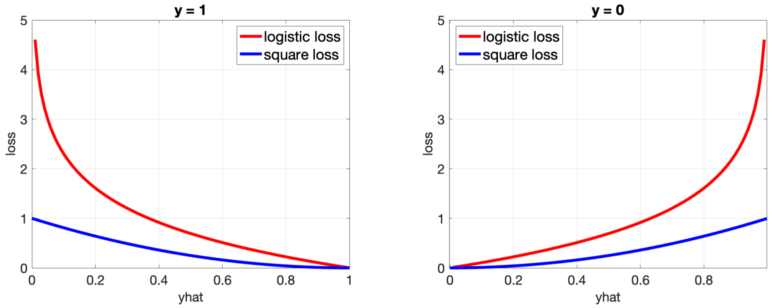
- The penalty of being totally wrong is not big enough (only 1);
- It is known to suffer from a learning slowdown when solving the optimization problem numerically through gradient descent.

There is a better choice of loss function, called the logistic loss or the cross-entropy loss, which we introduce next.

The logistic loss function

The logistic loss is defined as follows:

$$\ell(y,\hat{y}) = -y\log\hat{y} - (1-y)\log(1-\hat{y}) = \begin{cases} -\log\hat{y}, & y=1; \\ -\log(1-\hat{y}), & y=0 \end{cases}$$



It poses much heavier penalties on very wrong predictions than the square loss. It also has a very meaningful statistical interpretation.

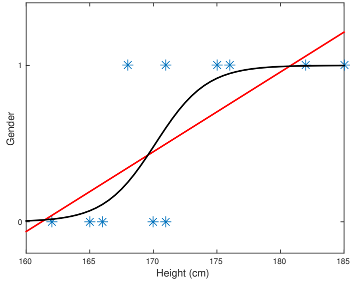
Statistical modeling of logistic regression

Key idea: To model the response Y as a Bernoulli random variable with a probability of success dependent on location x and parametrized by $\boldsymbol{\theta} = (\theta_0, \theta_1)$ (in the form of the sigmoid function):

$$Y \mid x, \boldsymbol{\theta} \sim \text{Bernoulli}(p), \quad \text{where} \quad p(x, \boldsymbol{\theta}) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

That is,

$$P(Y = 1 \mid x, \boldsymbol{\theta}) = p(x, \boldsymbol{\theta}),$$
$$P(Y = 0 \mid x, \boldsymbol{\theta}) = 1 - p(x, \boldsymbol{\theta})$$



Note that the pmf of Y when given x and $\boldsymbol{\theta}$ can be written as

$$P(Y = y \mid x, \boldsymbol{\theta}) = p^y (1-p)^{1-y}, \quad \text{for } y = 0, 1$$

Given independently sampled training examples $(x_i, y_i), 1 \leq i \leq n$, the likelihood of the sample is

$$L(\boldsymbol{\theta} \mid (x_i, y_i), 1 \leq i \leq n) = \prod_{i=1}^{n} P(Y = y_i \mid x_i, \boldsymbol{\theta}) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}$$

where

$$p_i = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_i)}}, \quad 1 \leq i \leq n$$

The log likelihood is

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i \log p_i + (1 - y_i) \log(1 - p_i))$$

Its maximizer, called the Maximum Likelihood Estimator (MLE) of $\boldsymbol{\theta}$ and denoted $\hat{\boldsymbol{\theta}}$, gives the optimal parameter values for the model.

This is a very important tool for machine learning (whenever parameter estimation for a distribution is involved).

Connection to optimization

Mathematically, the MLE formulation

$$\max_{\boldsymbol{\theta}} \log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

is equivalent to the minimization problem

$$\min_{\boldsymbol{\theta}} -\log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (-y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i))$$

The terms in the above summation for $-\log L(\boldsymbol{\theta})$, i.e.,

$$\ell(y_i, \hat{y}_i) = -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$

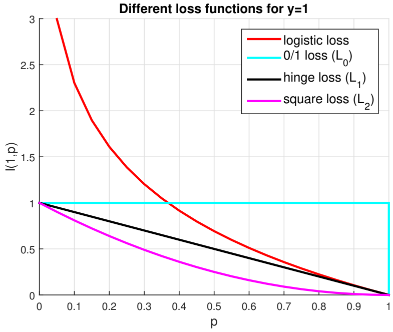
can be thought of as measuring the error between y_i and its approximation $\hat{y}_i$ (the logistic loss).

More loss functions

It turns out that by changing the choice of the loss function, we can invent more classifiers by the optimization approach.

Here are a few loss functions (with their graphs on the right):

- 0/1 loss: $\ell(y, \hat{y}) = 1_{y \neq \hat{y}}$
- Square loss: $\ell(y, \hat{y}) = (y - \hat{y})^2$
- Hinge loss: $\ell(y, \hat{y}) = |y - \hat{y}|$
- Logistic loss: $\ell(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$.



Other commonly used loss functions include the log loss and exponential loss.

How to classify new observations

After we fit the logistic model to the training set (and obtain the MLE θ),

$$\hat{y} = \frac{1}{1 + e^{-(\hat{\theta}_0 + \hat{\theta}_1 x)}}$$

we may use the following decision rule for a new observation x:

$$\text{Assign label } y = 1 \quad \text{if and only if} \quad \hat{y} > \frac{1}{2}.$$

Remark. Logistic regression is also a Bayes classifier because it is based on the maximum posterior probability:

$$\hat{y} > \frac{1}{2} \quad \text{if and only if} \quad \underbrace{P(Y = 1 \mid x, \theta)}_{\hat{y}} > \underbrace{P(Y = 0 \mid x, \theta)}_{1-\hat{y}}$$

The general binary classification problem

When there are multiple predictors $x_1, \ldots, x_d$, we let

$$y = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d)}} = \frac{1}{1 + e^{-\boldsymbol{\theta} \cdot \mathbf{x}}}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_d)$ and $\mathbf{x} = (1, x_1, \ldots, x_d)$, and use the log-likelihood function to find the MLE $\hat{\boldsymbol{\theta}}$:

$$\hat{y} = \frac{1}{1 + e^{-\hat{\boldsymbol{\theta}} \cdot \mathbf{x}}}$$

The classification rule for a test point $\mathbf{x}$ remains the same:

$$y = 1_{\hat{y} > 0.5}$$

We call this classifier the binary Logistic Regression (LR) classifier.

What kind of classifier is LR?

The decision boundary consists of all points $\mathbf{x} \in \mathbb{R}^d$ such that

$$\hat{y} = \frac{1}{1 + e^{-\hat{\boldsymbol{\theta}} \cdot \mathbf{x}}} = \frac{1}{2}$$

or equivalently,

$$\hat{\boldsymbol{\theta}} \cdot \mathbf{x} = \hat{\theta}_0 + \hat{\theta}_1 x_1 + \cdots + \hat{\theta}_d x_d = 0$$

which is a hyperplane in $\mathbb{R}^d$.

This shows that LR is a linear classifier.

LR is a generalized linear model (GLM)

The LR model

$$p(\mathbf{x}, \boldsymbol{\theta}) = \hat{y} = \frac{1}{1 + e^{-\boldsymbol{\theta}\cdot\mathbf{x}}}$$

can be rewritten as

$$\text{link function (logit)} \longrightarrow \quad \log \frac{p}{1-p} = \boldsymbol{\theta} \cdot \mathbf{x}$$

where the response Y is a Bernoulli random variable with mean

$$\mathrm{E}(Y \mid \mathbf{x}; \boldsymbol{\theta}) = p(\mathbf{x}, \boldsymbol{\theta})$$

Interpretation:

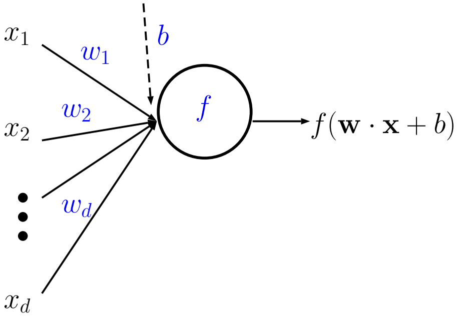
- p: probability of “success” (i.e. $Y = 1$)
- $\frac{p}{1-p}$: odds of “winning”
- $\log \frac{p}{1-p}$: logit (a link function)

LR is a single-neuron network

In logistic regression, we already see a lot of the concepts that are used in the design and training of a neural network, such as

- Activation function (sigmoid, and more)
- Loss function (square, logistic)
- Gradient descent (for solving the optimization problem)

To learn more on this, see Nielson's online book on neural networks and deep learning[a] (very easy and enjoyable to read).

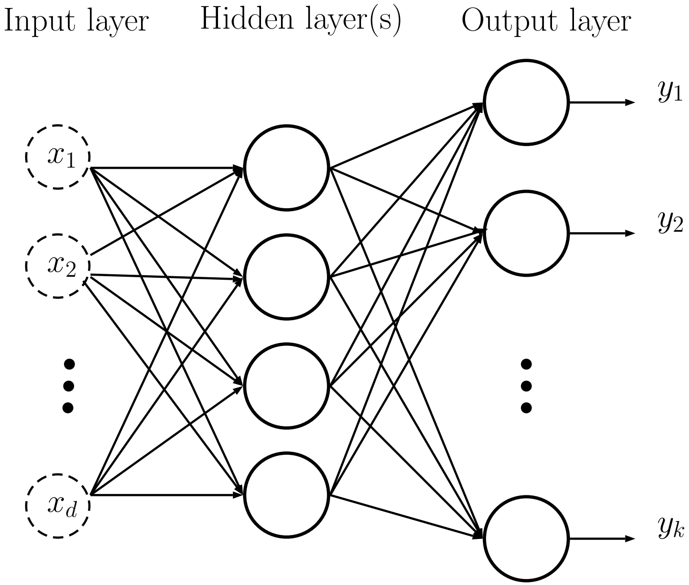


In the above,

- w_i: **weights**, b: **bias**, and f: **activation function**

[a] https://neuralnetworksanddeeplearning.com/

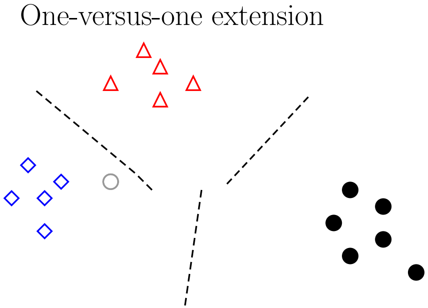
A neural network

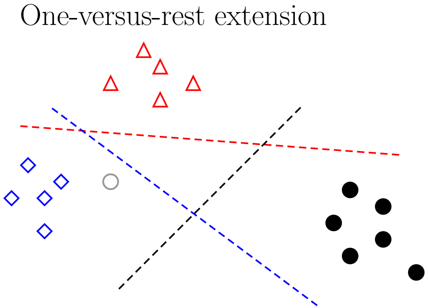


Multiclass extensions of LR

1. **Union of binary models**
   - *One versus one*: construct a LR model for every pair of classes
   - *One versus rest*: construct a LR model for each class against the rest of the training set

   In either case, the prediction of the label of a test point is collectively determined by all the models (see next slide).





How to determine the label of a test point $\mathbf{x}_0$:

- For the one-versus-one extension, the final prediction for a test point is the majority vote by all the pairwise models;
- For the one-versus-rest extension,
    - Use each class in turn as a reference class exactly once
    - Each time the chosen reference class (say class j) receives new label 1 (and the rest are given label 0). Fix a binary LR model with parameters $\hat{\boldsymbol{\theta}}^{(j)}$.
    - For each binary model with class j as the reference class, compute the probability

$$\hat{y}_0^{(j)} = P(Y_0 = 1 \mid \mathbf{x}_0, \hat{\boldsymbol{\theta}}^{(j)}) = \frac{1}{1 + e^{-\hat{\boldsymbol{\theta}}^{(j)} \cdot \mathbf{x}_0}}$$

    - The final prediction is

$$\hat{y}_0 = \arg\max_j \ \hat{y}_0^{(j)}$$

2. **Softmax regression**. It is based on the so-called **softmax function**:

$$\mathrm{softmax}(\alpha_1, \ldots, \alpha_c, j) = \frac{e^{\alpha_j}}{\sum_{1 \le \ell \le c} e^{\alpha_\ell}}, \quad j = 1, \ldots, c$$

   The softmax function is a smooth function trying to approximate the indicator function (whether α_j is the largest of all):

$$1_{\alpha_j = \max(\alpha_1, \ldots, \alpha_c)} = \begin{cases} 1, & \text{if } \alpha_j = \max(\alpha_1, \ldots, \alpha_c) \\ 0, & \text{otherwise} \end{cases}.$$

   See
   `https://neuralnetworksanddeeplearning.com/chap3.html`.

The softmax regression fits the following model to the data:

$$P(Y = j \mid \mathbf{x}; \Theta) = \operatorname{softmax}(\boldsymbol{\theta}^{(1)} \cdot \mathbf{x}, \ldots, \boldsymbol{\theta}^{(c)} \cdot \mathbf{x}, j) = \frac{e^{\boldsymbol{\theta}^{(j)} \cdot \mathbf{x}}}{\sum_{1 \le \ell \le c} e^{\boldsymbol{\theta}^{(\ell)} \cdot \mathbf{x}}},$$

with (redundant) parameters $\Theta = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(c)}\}$.

The distribution of Y is multinomial with probabilities $P(Y = j \mid \mathbf{x}; \Theta), 1 \le j \le c$. Therefore, softmax regression is also called **multinomial logistic regression**.

Remark. Multinomial LR can be thought of as a “fixed vs rest” extension of the binary LR classifier.

$$\frac{P(Y = j \mid \mathbf{x})}{P(Y = 1 \mid \mathbf{x})} = e^{(\boldsymbol{\theta}^{(j)} - \boldsymbol{\theta}^{(1)}) \cdot \mathbf{x}}, \quad j = 2, \ldots, c$$

Given training data $\{(\mathbf{x}_i, y_i)\}_{1\le i\le n}$, softmax regression estimates the parameters by maximizing the likelihood of the training set:

$$L(\Theta) = \prod_{i=1}^{n} P(Y = y_i \mid \mathbf{x}_i; \Theta) = \prod_{i=1}^{n} \frac{e^{\boldsymbol{\theta}^{(y_i)}\cdot\mathbf{x}_i}}{\sum_{1\le j\le c} e^{\boldsymbol{\theta}^{(j)}\cdot\mathbf{x}_i}}$$

Prediction for a new point $\mathbf{x}_0$ is based on the largest posterior probability:

$$\begin{aligned}\hat{y}_0 &= \operatorname{argmax}_{1\le j\le c} P(Y = j \mid \mathbf{x}_0; \widehat{\Theta})\\ &= \operatorname{argmax}_j \frac{e^{\hat{\boldsymbol{\theta}}^{(j)}\cdot\mathbf{x}_0}}{\sum_{1\le \ell\le c} e^{\hat{\boldsymbol{\theta}}^{(\ell)}\cdot\mathbf{x}_0}}\\ &= \operatorname{argmax}_j \hat{\boldsymbol{\theta}}^{(j)}\cdot\mathbf{x}_0\end{aligned}$$

Feature extraction/selection for logistic regression

Logistic regression tends to overfit the data in the setting of high dimensional data (i.e., many predictors). Two ways to fix it:

- Use a **dimensionality reduction** method (such as PCA) to project data into low dimensions
- Add a **penalty** term to the objective function of the binary logistic regression classifier

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (-y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)) + \lambda \cdot \text{penalty}$$

where the penalty term is normally set to

- $\|\boldsymbol{\theta}\|_2^2 = \theta_0^2 + \theta_1^2 + \cdots + \theta_d^2$ (ℓ_2 regularization) or
- $\|\boldsymbol{\theta}\|_1 = |\theta_1| + \cdots + |\theta_d|$ (ℓ_1 regularization).

The constant $\lambda \geq 0$ is called a regularization parameter; larger values of λ would lead to more zeros in $\boldsymbol{\theta}$ (simpler models).

Logistic regression in Python (via sklearn)[1]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'.

Note that regularization is applied by default.

# with default values
logreg = linear_model.LogisticRegression(penalty= 'l2', C=1.0, multi_class='auto')
# **penalty**: may be set to None, or 'l1' (which can produce more zeros in $\boldsymbol{\theta}$ than 'l2')
C $(= 1/\lambda)$: inverse of regularization strength (smaller values of C specify stronger regularization).
# **multi_class**: may be changed to 'multinomial' (no 'ovo' option)

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Other implementations of LR

See the Lab (Section 4.7 of the lab) and also the following:

`https://library.virginia.edu/data/articles/logistic-regression-four-ways-with-python`