Classification: Logistic regression, k nearest neighbours

Rui Zhu

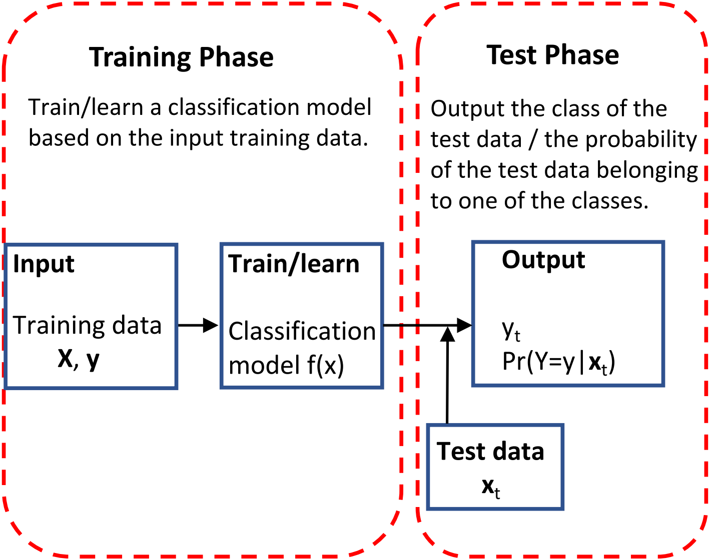
Overview

1. Why linear regression doesn't work for classification?
2. Logistic regression
3. k nearest neighbours (kNN)

Supervised learning: objectives

1. Training and test data
   - Training data: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$
   - Test data: $\mathbf{x}_t$ for one test instance, or $\mathbf{X}_t \in \mathbb{R}^{N_t \times p}$ if we have N_t test instances.
2. Given the training data, we aim to
   - Understand the association between outcomes and inputs.
   - Predict the response/class, $\hat{y}_t$, of the test data $\mathbf{x}_t$.
   - Assess the quality of the predictions and inferences.

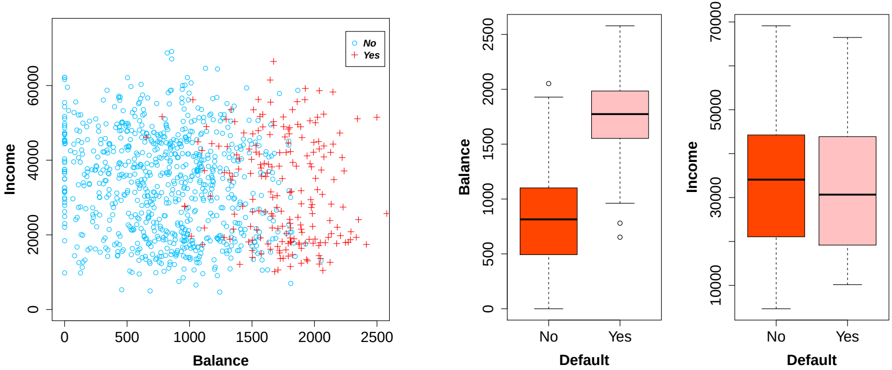
Classification



Linear regression as a classifier

Default data:

- $Y \in \{\text{Not default}, \text{Default}\}$
- Binary classification



Linear regression as a classifier

- We code the outcome measurement

$$Y = \begin{cases} 0, & \text{if No}, \\ 1, & \text{if Yes}. \end{cases}$$

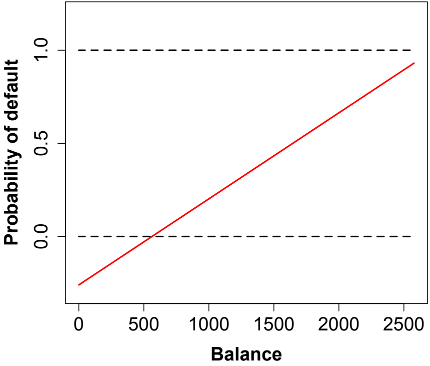
- We can perform a linear regression of Y on X

$$Y = \beta_0 + \beta_1 \text{Income} + \beta_2 \text{Balance} + \epsilon$$

and classify as Yes if $\hat{y}_t > 0.5$.

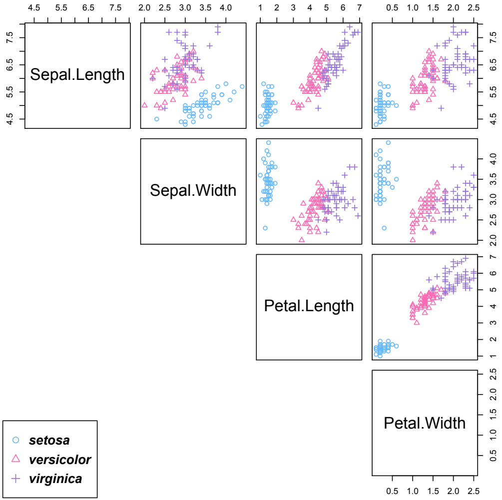
Problems of using linear regression as a classifier

- Probability of default $Pr(Y = 1|X = \mathbf{x})$: linear regression can produce negative estimates of probabilities.



Problems of using linear regression as a classifier

- Problem of using linear regression for multi-class classification.



Problems of using linear regression as a classifier

We aim to classify a flower to three species, setosa, versicolor or virginica, and code Y as follows:

$$Y = \begin{cases} 1, & \text{if setosa,} \\ 2, & \text{if versicolor,} \\ 3, & \text{if virginica.} \end{cases}$$

This coding of Y implies:

- an order of the three species,
- the difference between setosa and versicolor is the same as that between versicolor and virginica,

which are not appropriate.

Classifier

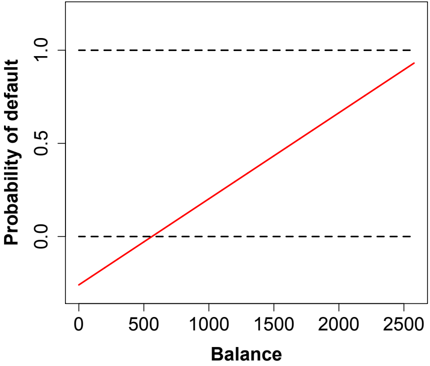
Therefore we need methods that are specificly designed for the classification tasks. We usually call these methods classifiers.

Two classifiers to learn today:

- Logistic regression
- k nearest neighbours

Problems of using linear regression as a classifier

- Probability of default $Pr(Y = 1|X = \mathbf{x})$: linear regression can produce negative estimates of probabilities.



Logistic regression

Let's write

$$Pr(Y = 1|X) = \frac{\exp(\alpha + \beta_1 X_1 + \dots \beta_p X_p)}{1 + \exp(\alpha + \beta_1 X_1 + \dots \beta_p X_p)}, \tag{1}$$

where $X = (X_1, \dots X_p)$ are p predictors/features. We can use the maximum likelihood method to estimate $\alpha, \beta_1, \dots, \beta_p$.

Logistic regression

	Coefficient	Std. error	Z-statistic	P-value
`Intercept`	-10.8690	0.4923	-22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	-0.6468	0.2362	-2.74	0.0062

Table: For the `Default` data, estimated coefficients of the logistic regression model that predicts the probability of default using `balance`, `income`, and `student` status. Student status is encoded as a dummy variable `student[Yes]`, with a value of 1 for a student and a value of 0 for a non-student. In fitting this model, `income` was measured in thousands of dollars.

Logistic regression

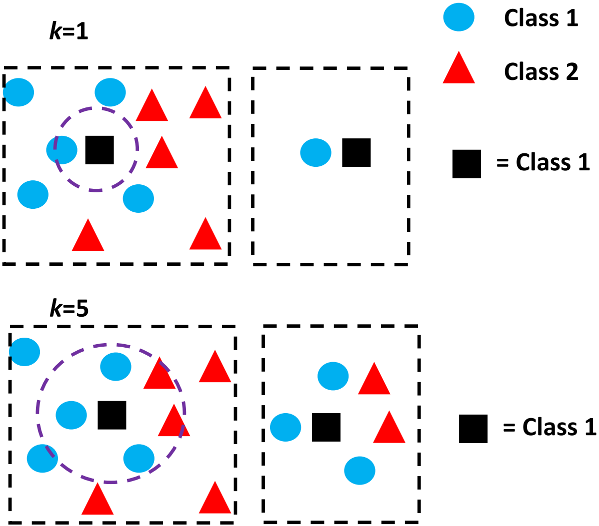
A student with a credit card balance of \$1,500 and an income of \$40 (here meaning \$40000 since the variable is scaled in terms of thousands of dollars) has an estimated probability of default of

$$\begin{aligned}
&\hat{Pr}(Y = 1|X = \mathbf{x}) \\
&= \frac{e^{-10.869+0.00574\times 1{,}500+0.003\times 40-0.6468\times 1}}{1 + e^{-10.869+0.00574\times 1{,}500+0.003\times 40-0.6468\times 1}} \\
&= 0.058.
\end{aligned}$$

This probability is very small, we can classify this student as Not Default. Usually the threshold is set to 0.5.

k nearest neighbours (kNN)

The class of an instance is the same as that of the majority of its k nearest neighbours.



k nearest neighbours (kNN)

- kNN assigns the instance with features $\mathbf{x}_0$ to the class with the largest conditional probability.

$$\Pr(Y = j | X = \mathbf{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

- j: class j, $j = 1, 2, \ldots, C$
- k: number of nearest neighbours
- i: index of instance
- $\mathcal{N}_0$: the nearest neighbours of $\mathbf{x}_0$
- $I(\cdot)$: indicator function

k nearest neighbours (kNN)

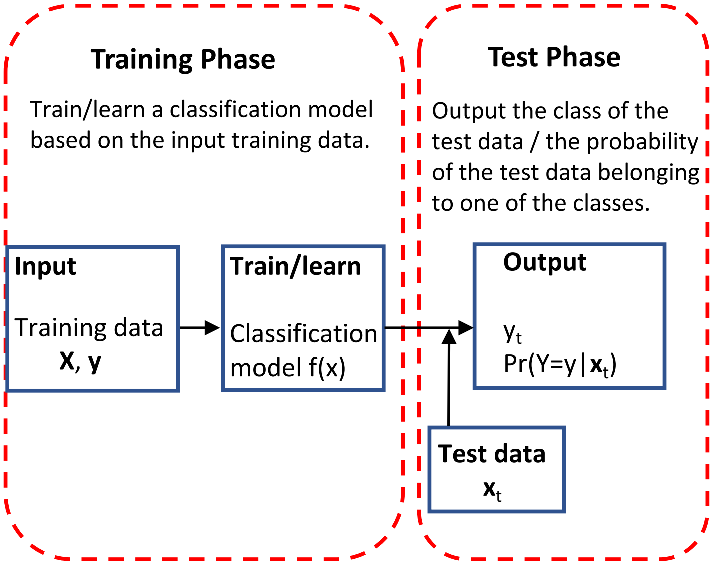
- For $k = 5$: N_0 contains 3 instances from Class 1 and 2 from Class 2.
- $$\text{Pr}(Y = 1 | X = \mathbf{x}_0) = \frac{3}{5}$$
  $$\text{Pr}(Y = 2 | X = \mathbf{x}_0) = \frac{2}{5}$$
  $$\text{Pr}(Y = 1 | X = \mathbf{x}_0) > \text{Pr}(Y = 2 | X = \mathbf{x}_0)$$
- We assign the new instance $\mathbf{x}_0$ to Class 1.

kNN

kNN is a lazy learning algorithm: all computation is deferred until classifying a new/test instance.

- Given a specific k, there is no computation in the training process.
- It's very simple.

kNN


Training Phase
Train/learn a classification model based on the input training data.
Test Phase
Output the class of the test data / the probability of the test data belonging to one of the classes.
Input
Training data
X, y
Train/learn
Classification model f(x)
Output
y_t
$Pr(Y=y|\mathbf{x}_t)$
Test data
$\mathbf{x}_t$

kNN

More about kNN:

- How to determine the nearest neighbours?
- Do we need a preprocessing step to transform the data?

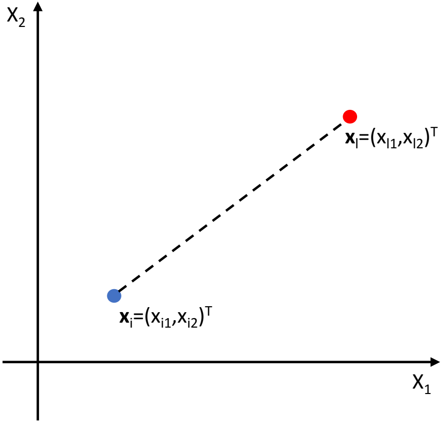
kNN: How to determine the nearest neighbours?

How to determine the nearest neighbours in N_0?

- Distance between $\mathbf{x}_i$ and $\mathbf{x}_l$
- The most commonly used distance: Euclidean distance
- Other distance measurements, e.g. Mahalanobis distance
- We only focus on the Euclidean distance in this module

kNN: Euclidean distance

In a two-dimensional feature space, the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_l)$ between two instances, $\mathbf{x}_i = (x_{i1}, x_{i2})^T$ and $\mathbf{x}_l = (x_{l1}, x_{l2})^T$, is $d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2}$.



kNN: Euclidean distance

- A simple example: if $\mathbf{x}_i = (1,2)^T$ and $\mathbf{x}_l = (10,3)^T$, then $d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(1-10)^2 + (2-3)^2}$.
- For instances living in a p-dimensional space,

$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \ldots + (x_{ip} - x_{lp})^2}.$$

- Vector representation:

$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(\mathbf{x}_i - \mathbf{x}_l)^T (\mathbf{x}_i - \mathbf{x}_l)}.$$

kNN: preprocessing data

- Consider the situation: a 2-dimensional space described by X_1 and X_2, where X_1 can take values in [0,1] while X_2 can take values in [0,10000].
- X_1 has very small contribution to the Euclidean distance.
- The Euclidean distance can be dominated by the values of X_2.
- Solution: scale the features.

kNN: preprocessing data

- Scale the features: a preprocessing of columns in $\mathbf{X} \in \mathbb{R}^{N \times p}$.
- Standardise: make the features have mean 0 and standard deviation 1 (subtract the mean and divide the standard deviation).

kNN for regression

The response of an instance is the average of the responses of its nearest neighbours.

$$\hat{Y}(\mathbf{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} y_i$$

A very simple example of kNN for classification

Training set:

	x_1	x_2	y
1	0.5	1.2	1
2	0.8	0.9	1
3	1.3	1.5	1
4	0.1	2.1	2
5	1.8	0.7	2
6	0.9	1.7	2

Test data: $\mathbf{x}_t = (1, 1)^T$.

Task: classify $\mathbf{x}_t$ to class 1 or class 2?

A very simple example of kNN for classification

3NN:

- Calculate the Euclidean distances between $\mathbf{x}_t$ and training instances:

$$(0.539, 0.224, 0.583, 1.421, 0.854, 0.707)$$

- Sort the distances in assending order:

$$(0.224, 0.539, 0.583, 0.707, 0.854, 1.421)$$

  The corresponding training instance indexes are

$$(2, 1, 3, 6, 5, 4)$$

- Select the first three instances $(2, 1, 3)$ as nearest neighbours, whose labels are $(1, 1, 1)$.
- We label $\mathbf{x}_t$ as class 1.

How about 5 *NN?*

kNN: an example for classification