4 Logistic regression

Contents

Abstract

We now switch from regression problems to classification problems. Don't be confused by the name "Logistic Regression"; it is named that way for historical reasons and is actually an approach to classification problems, not regression problems. We begin by studying **Binary Classification Problem** where a machine learning algorithm classifies the inputs into one of two possible the outputs, and **Multiclass Classification** where there are many possible classes (this is just an extension of binary classification).

4.1 Binary Classification Problem: Representation

Now our output vector $y \in \{0,1\}$ where 0 typically represents the "negative" class and 1 as the "positive class".
Remark. One method is to use linear regression:

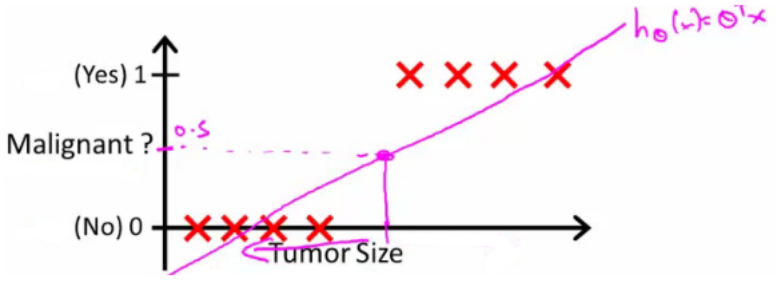


Figure 4.1: Linear regression can be used for classification, e.g. in this example by maping all predictions $h_x(\boldsymbol{\theta}) \geq 0.5$ to 1 and $h_x(\boldsymbol{\theta}) < 0.5$ to 0. This method doesn't work well because *classification is not actually a linear function* - what if we had a single **Yes** with a very small tumour?

4.1.1 Hypothesis Representation

Since $y \in \{0,1\}$, our hypothesis should satisfy $0 \leq h_x(\boldsymbol{\theta}) \leq 1$. Using the **Sigmoid Function** (or **Logistic Function**) on our linear regression hypothesis:

$$\begin{aligned} z &= \boldsymbol{\theta}^T \mathbf{x} \\ \therefore h_\theta(x) &= \frac{1}{1+e^{-z}} \end{aligned}$$

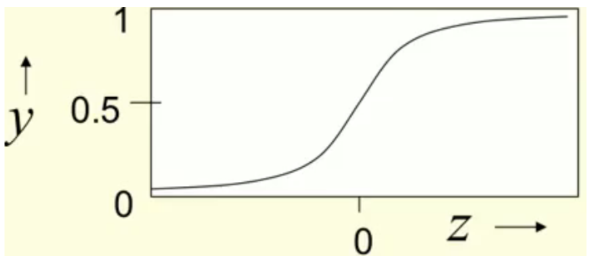


Figure 4.2: The sigmoid function; this has aymptotes at +1 and 0.

4.1.2 Interpretation of the $h_\theta(x)$

We interpret $h_\theta(x)$ as ***"The estimated probabilty that** $y = 1$ **on input** x**"*** (e.g. $h_\theta(x) = 0.7\%$ indicates we can tell the patient they have a 70 % chance of tumour being malignant); that is $h_\theta(x) = \mathbb{P}(y = 1; \mathbf{x}, \boldsymbol{\theta})$.
Since $y \in \{0, 1\}$, we deduce that $\mathbb{P}(y = 0; \mathbf{x}, \boldsymbol{\theta}) + \mathbb{P}(y = 1; \mathbf{x}, \boldsymbol{\theta}) = 1$ and so ***"The estimated probabilty that** $y = 0$ **on input** x**"*** is $1 - h_\theta(x)$.

4.1.3 Decision Boundary

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function via:

$$\begin{aligned} h_\theta(x) \geq 0.5 &\iff z \geq 0 &\mapsto\quad y = 1 \\ h_\theta(x) < 0.5 &\iff z < 0 &\mapsto\quad y = 0 \end{aligned}$$

Definition 4.1. The **decision boundary** is the line that separates the area where $y = 0$ and where $y = 1$, that is created by our hypothesis function.

Remark. The input to the sigmoid function $z = \boldsymbol{\theta}^T\mathbf{x}$ need not be linear, and could be a function that describes any shape that fits our data.

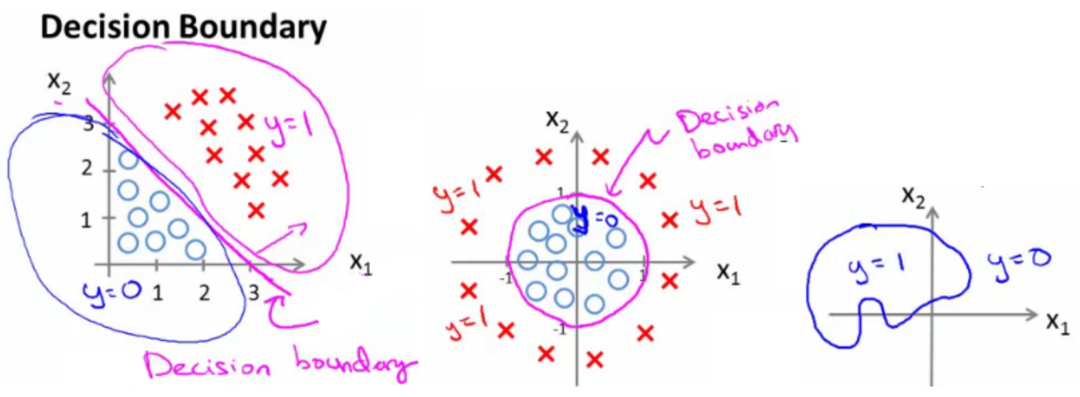


Figure 4.3: Example decision boundaries: on the left, we use the usual (linear) $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$; in the middle, we could have e.g. $h_\theta(x) = g\left(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2 + \theta_4 x_2^2\right)$ so that $\boldsymbol{\theta}^T = [-1, 0, 0, 1, 1]$ and $y = 1$ if $-1 + x_1^2 + x_2^2 \geq 0 \Rightarrow x_1^2 + x_2^2 \geq 1$; so the right, we see that more complex decision boundaries can be found using higher order polynomial terms.

4.2 Binary Classification Problem: Evaluation (Cost function)

We cannot use the same cost function that we use for linear regression because the logistic function will cause the output to be wavy, causing *many local optima* (i.e. it will not be a convex function). Thus the cost function for learning the parameters $\boldsymbol{\theta}$:

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x), y) \quad \text{where Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note. The further away our hypothesis, $h_\theta(x)$, is away from y, the larger the Cost function; conversly, when $h_\theta(x) = y$ our Cost function is 0:

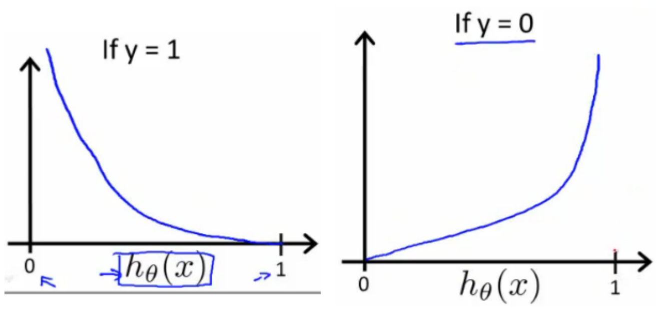


Figure 4.4: The Cost function behaves in a similar manner to the one for linear regression, and ensures $J(\theta)$ is convex for logistic regression.

Remark. This cost function can be derived from statistics using the principle of maximum likelihood estimation.

4.3 Binary Classification Problem: Optimisation

4.3.1 Simplified cost function and gradient descent

Since $y \in \{0,1\}$, we can write the Cost function as $\text{Cost}(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$; thus

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{m}\sum_{i=1}^{m}\left[-y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))\right] \\ &= -\frac{1}{m}\left[y\log g\left(X\boldsymbol{\theta}^T\right) + (1-y)\log(1-g(X\theta T))\right] \end{aligned}$$

4.3.2 Minimising the cost function for logistic regression

We can use the same algorithm for gradient descent as for multivariate linear regression; the equation is the same but our definition for the hypothesis has changed.

Algorithm 1. ***Gradient Descent***

```
repeat until convergence {
$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha\nabla J(\boldsymbol{\theta})$
}
```

for α some (positive) ***learning rate*** *(aka. step size) and some initial guess for $\boldsymbol{\theta}$.*

Remark. $\nabla J(\boldsymbol{\theta}) = \left[\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_0}, \frac{\partial J(\boldsymbol{\theta})}{\partial\theta_1}, ..., \frac{\partial J(\boldsymbol{\theta})}{\partial\theta_n}\right]^T$ where $\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j} = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}\left(h_{\boldsymbol{\theta}}\left(x^{(i)}\right) - y^{(i)}\right)$. Equally $\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j} = \frac{1}{m}\mathbf{x}_j^T\left(g(X\boldsymbol{\theta}) - \mathbf{y}\right)$, so $\nabla J(\boldsymbol{\theta}) = \frac{1}{m}X^T\left(g(X\boldsymbol{\theta}) - \mathbf{y}\right)$ and $\boldsymbol{\theta} := \boldsymbol{\theta} - \frac{\alpha}{m}X^T\left(g(X\boldsymbol{\theta}) - \mathbf{y}\right)$.

Remark. Similarly, as before:

- We can monitor this gradient descent to check it's working correctly.
- Vectorized implementation is possible (and as before, more efficient).
- Feature scaling for gradient descent for logistic regression also applies.

4.3.3 Advanced optimisation

Previously we looked at gradient descent for minimizing the cost function. Here look at advanced concepts for minimizing the cost function for logistic regression.

Recall the aim of gradient descent is to minimise some cost function $J(\boldsymbol{\theta})$. Thus we need to write code which can take $\boldsymbol{\theta}$ as input and compute the following:

- $J(\boldsymbol{\theta})$
- Partial derivatives of $J(\boldsymbol{\theta})$ w.r.t. j (for $j = 0, ..., n$).

Conjugate gradient, **BFGS** (Broyden-Fletcher-Goldfarb-Shanno or “Big Friendly Giants Steps”) and **L-BFGS** (Limited memory - BFGS) are more sophisticated, faster ways to optimize theta instead of using gradient descent.

Advantages:

- No need to manually pick alpha (learning rate).
  - Have a clever inner loop (line search algorithm) which tries a bunch of alpha values and picks a good one.
- Often faster than gradient descent.
  - Do more than just pick a good learning rate.
- Can be used successfully without understanding their complexity

Disadvantages:

- Could make debugging more difficult.
- Different libraries may use different implementations - may hit performance.

4.4 Multiclass classification problems

We now look at getting logistic regression for multiclass classification (where $y \in \{0, 1, ..., n\}$) using **one vs. all** (or **one vs. rest**).

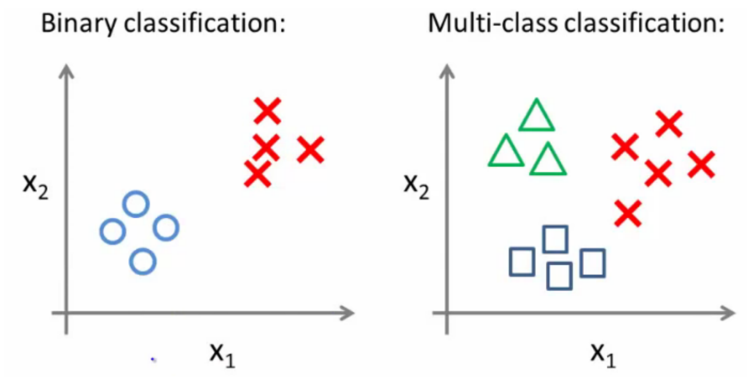


Figure 4.5: Binary classification vs Multi-class classification.

One vs. rest classification make binary classification work for multiclass classification:

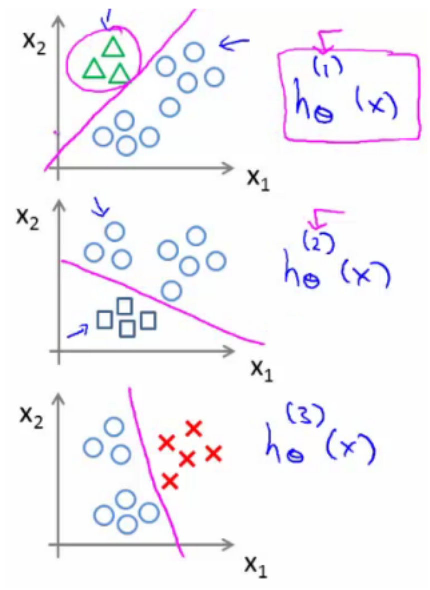
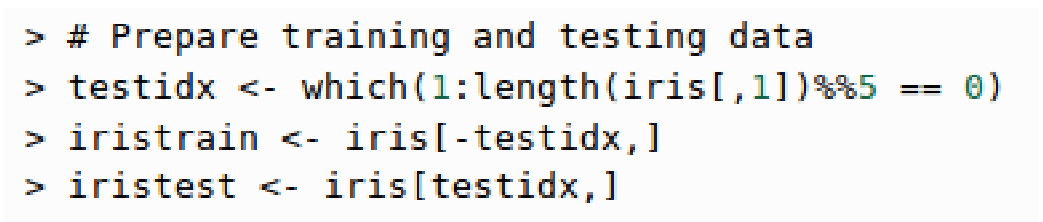
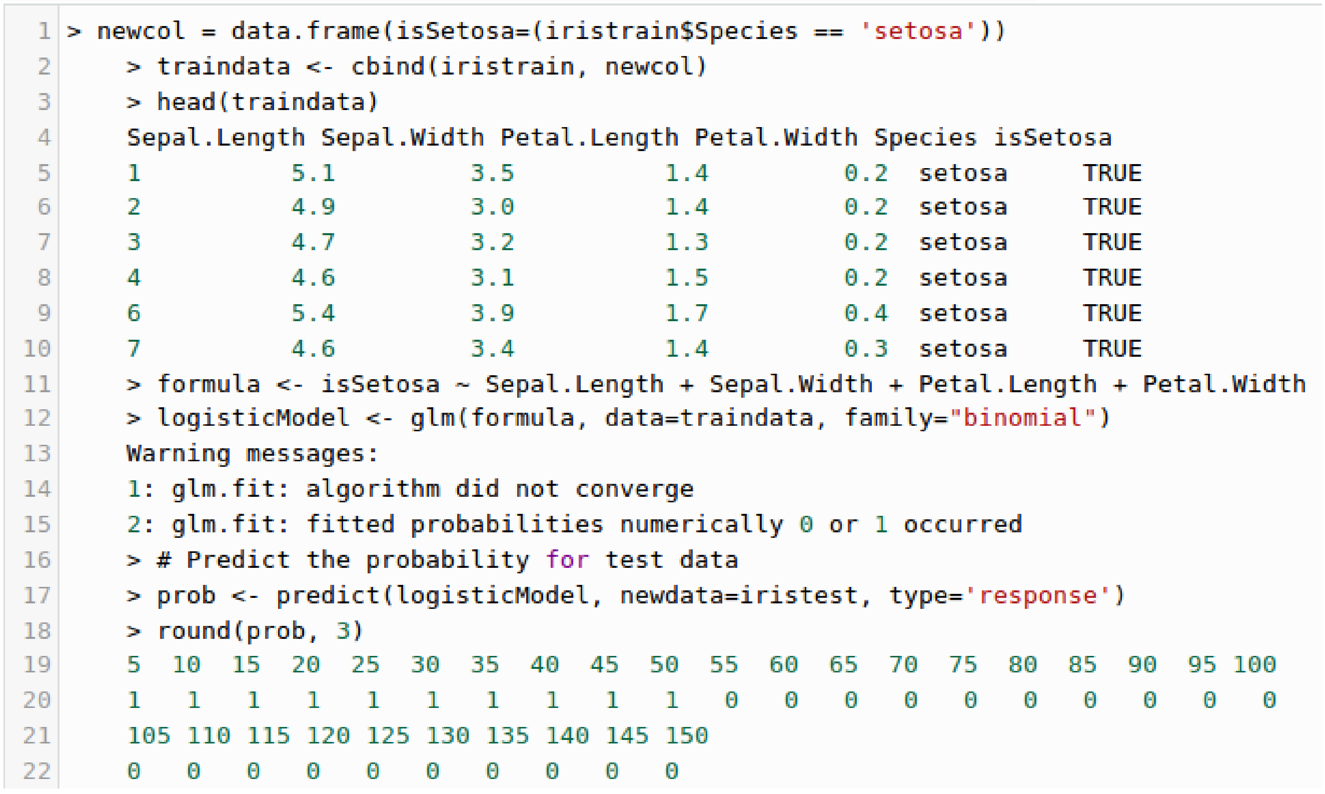


Figure 4.6: One vs. all algorithm (3-class problem):
1. Split the training set into 3 separate binary classification problems (i.e. create a new fake training set).
2. Train logistic regression classifier on each sub problem:
a. *Triangle vs. Rest:* Train logistic regression classifier, $h_\theta^{(1)}(x) = \mathbb{P}\left(y = 1 \mid \mathbf{x}^{(1)}; \theta\right)$.
b. *Square vs. Rest:* Train logistic regression classifier $h_\theta^{(2)}(x) = \mathbb{P}\left(y = 1 \mid \mathbf{x}^{(2)}; \theta\right)$.
c. *Crosses vs. Rest:* Train logistic regression classifier $h_\theta^{(3)}(x) = \mathbb{P}\left(y = 1 \mid \mathbf{x}^{(3)}; \theta\right)$.
3. The prediction is then $\max_i \left(h_\theta^{(i)}(x)\right)$.

4.5 R Example



```
> # Prepare training and testing data
> testidx <- which(1:length(iris[,1])%%5 == 0)
> iristrain <- iris[-testidx,]
> iristest <- iris[testidx,]
```



```
> newcol = data.frame(isSetosa=(iristrain$Species == 'setosa'))
    > traindata <- cbind(iristrain, newcol)
    > head(traindata)
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species isSetosa
    1          5.1         3.5          1.4         0.2  setosa     TRUE
    2          4.9         3.0          1.4         0.2  setosa     TRUE
    3          4.7         3.2          1.3         0.2  setosa     TRUE
    4          4.6         3.1          1.5         0.2  setosa     TRUE
    6          5.4         3.9          1.7         0.4  setosa     TRUE
    7          4.6         3.4          1.4         0.3  setosa     TRUE
    > formula <- isSetosa ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
    > logisticModel <- glm(formula, data=traindata, family="binomial")
    Warning messages:
    1: glm.fit: algorithm did not converge
    2: glm.fit: fitted probabilities numerically 0 or 1 occurred
    > # Predict the probability for test data
    > prob <- predict(logisticModel, newdata=iristest, type='response')
    > round(prob, 3)
    5   10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95 100
    1   1   1   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0
    105 110 115 120 125 130 135 140 145 150
    0   0   0   0   0   0   0   0   0   0
```

Figure 4.7: Logistic regression in **R**.