

Equation 4-14. Logistic function

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

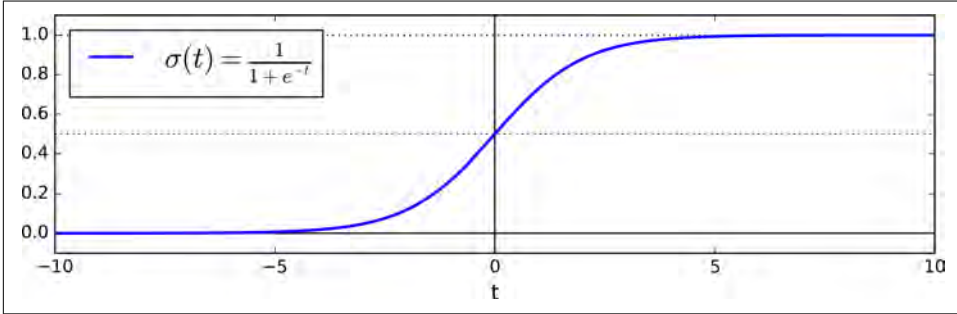


Figure 4-21. Logistic function

Once the Logistic Regression model has estimated the probability $\hat{p} = h_{\theta}(\mathbf{x})$ that an instance \mathbf{x} belongs to the positive class, it can make its prediction \hat{y} easily (see [Equation 4-15](#)).

Equation 4-15. Logistic Regression model prediction

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5, \\ 1 & \text{if } \hat{p} \geq 0.5. \end{cases}$$

Notice that $\sigma(t) < 0.5$ when $t < 0$, and $\sigma(t) \geq 0.5$ when $t \geq 0$, so a Logistic Regression model predicts 1 if $\theta^T \cdot \mathbf{x}$ is positive, and 0 if it is negative.

Training and Cost Function

Good, now you know how a Logistic Regression model estimates probabilities and makes predictions. But how is it trained? The objective of training is to set the parameter vector θ so that the model estimates high probabilities for positive instances ($y = 1$) and low probabilities for negative instances ($y = 0$). This idea is captured by the cost function shown in [Equation 4-16](#) for a single training instance \mathbf{x} .

Equation 4-16. Cost function of a single training instance

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1, \\ -\log(1 - \hat{p}) & \text{if } y = 0. \end{cases}$$

This cost function makes sense because $-\log(t)$ grows very large when t approaches 0, so the cost will be large if the model estimates a probability close to 0 for a positive

instance, and it will also be very large if the model estimates a probability close to 1 for a negative instance. On the other hand, $-\log(t)$ is close to 0 when t is close to 1, so the cost will be close to 0 if the estimated probability is close to 0 for a negative instance or close to 1 for a positive instance, which is precisely what we want.

The cost function over the whole training set is simply the average cost over all training instances. It can be written in a single expression (as you can verify easily), called the *log loss*, shown in [Equation 4-17](#).

Equation 4-17. Logistic Regression cost function (log loss)

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)}) \right]$$

The bad news is that there is no known closed-form equation to compute the value of θ that minimizes this cost function (there is no equivalent of the Normal Equation). But the good news is that this cost function is convex, so Gradient Descent (or any other optimization algorithm) is guaranteed to find the global minimum (if the learning rate is not too large and you wait long enough). The partial derivatives of the cost function with regards to the j^{th} model parameter θ_j is given by [Equation 4-18](#).

Equation 4-18. Logistic cost function partial derivatives

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m \left(\sigma(\theta^T \cdot \mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

This equation looks very much like [Equation 4-5](#): for each instance it computes the prediction error and multiplies it by the j^{th} feature value, and then it computes the average over all training instances. Once you have the gradient vector containing all the partial derivatives you can use it in the Batch Gradient Descent algorithm. That's it: you now know how to train a Logistic Regression model. For Stochastic GD you would of course just take one instance at a time, and for Mini-batch GD you would use a mini-batch at a time.

Decision Boundaries

Let's use the iris dataset to illustrate Logistic Regression. This is a famous dataset that contains the sepal and petal length and width of 150 iris flowers of three different species: Iris-Setosa, Iris-Versicolor, and Iris-Virginica (see [Figure 4-22](#)).