8

Introduction to classification and regression

We will now turn our attention to *supervised learning*. In supervised learning we are given a training set comprised of N observations, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and N targets $y_1, \ldots, y_N$ and we wish to come up with a way to predict y from $\boldsymbol{x}$:

$$y = f(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon, \tag{8.1}$$

where $\boldsymbol{w}$ is a vector of tunable parameters and ε represents a noise term. Learning then consists of selecting the parameters $\boldsymbol{w}$ based on the training data $\boldsymbol{X}, \boldsymbol{y}$. If y is a continuous parameter, for instance the price of a stock, we will say that the model (denoted $\mathcal{M}$) is a regression model. On the other hand if y is discrete, i.e. $y = 1, 2, \ldots, C$ as in the MNIST example we encountered in chapter 1, we will say $\mathcal{M}$ is a classification model. In this chapter, we will discuss the linear and logistic regression models for regression and classification, starting by first explaining what $f(\boldsymbol{x}, \boldsymbol{w})$ looks like and then show how probabilities can be used to treat the noise term ε.

The history of linear regression can be traced back to mathematicians Adrien-Marie Legendre and Carl Friedrich Gauss who independently in 1805 and 1809 applied linear regression models to astronomical observations of the orbit of planets [Legendre, 1805, Gauß, 1809]. Meanwhile logistic regression, which we will consider in a later section, has it's origin with the discovery of the logistic function by Pierre François Verhulst and Adolphe Quételetin in 1838 [Garnier and Quételet, 1838] where it was originally applied to growth curves of populations.

8.1 Linear models

Despite having different goals, linear and logistic regression are closely related by virtue of using a linear transformation of the input features which will be our natural starting point. Recall in a linear model, the output y in eq. (8.1) is modelled as a *linear combination* of the input features:

$$f(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_M x_M, \quad \text{and} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}. \tag{8.2}$$

The reason this is known as a linear model is that it is a *linear function* of the input features $\boldsymbol{x}$. To

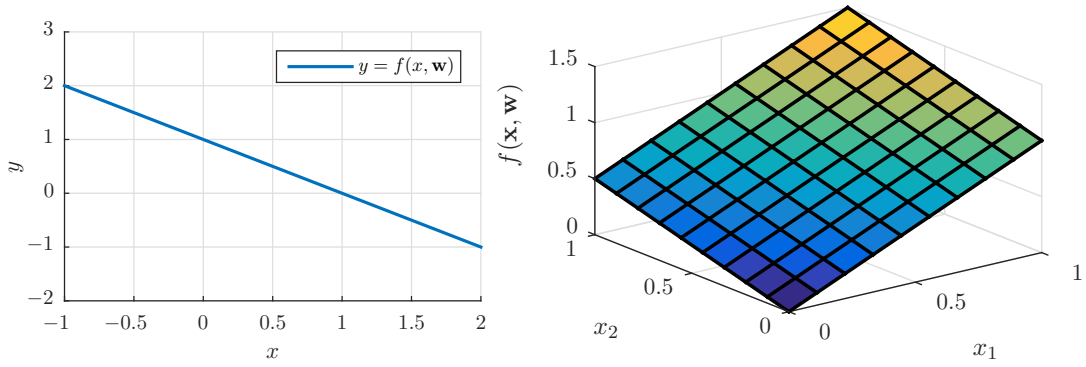


Fig. 8.1. The linear regression models prediction is a linear combination of the features $f(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_M x_M$. This allows for lines (left pane), $y = w_0 + w_1 x$, planes (right pane) $y = w_0 + w_1 x_1 + w_2 x_2$ and in general hyperplanes.

consider a very simple example, consider the linear model with $w_0 = 1, w_1 = -1$ shown in fig. 8.1 as the blue line

$$y = f(x, \boldsymbol{w}) = 1 - x.$$

This naturally also extends to multiple input features. In the right-hand pane of fig. 8.1 is shown the two-dimensional regression example with $w_0 = 0, w_1 = 1, w_2 = \frac{1}{2}$.

$$y = f(\boldsymbol{x}, \boldsymbol{w}) = 0 + x_1 + \frac{1}{2} x_2 = x_1 + \frac{1}{2} x_2.$$

More generally, we can consider a feature transformation of $\boldsymbol{x}$ such that

$$y = f(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\boldsymbol{x})$$

where $\phi_1(\boldsymbol{x}), \ldots, \phi_{M-1}(\boldsymbol{x})$ are $M - 1$ *basis functions*. If we define

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ \phi_1(\boldsymbol{x}) \\ \vdots \\ \phi_{M-1}(\boldsymbol{x}) \end{bmatrix}$$

(that is, the first basis function is just a constant) we can write the linear regression model more compactly as simply

$$y = f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w} \quad \text{and} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}. \tag{8.3}$$

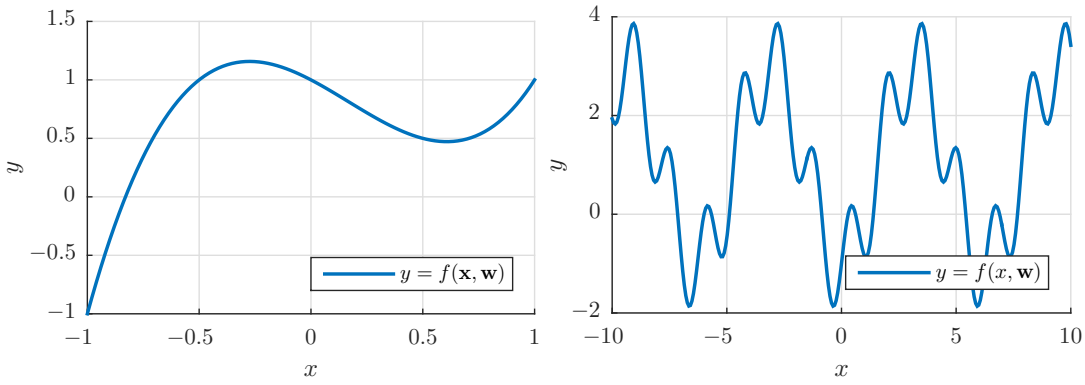


Fig. 8.2. Applying a non-linear transformation to the input x allows much more complicated curves to be fitted by the linear regression model. In the left-hand pane is shown a polynomial $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$ and in the right-hand pane a sinusoidal model $y = w_0 + w_1 \cos(x) + w_2 \sin(4x)$.

The use of non-linear basis functions allow the linear regression model to model non-linear features. In fig. 8.2 is shown two such examples, the first is a 3rd degree polynomial corresponding to

$$\boldsymbol{\phi}(x) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix}^T \quad \text{and} \quad \boldsymbol{w} = \begin{bmatrix} 1 & -1 & -1 & 2 \end{bmatrix}^T. \tag{8.4}$$

The second example corresponds to two trigonometric functions suitable for a periodic signal

$$\boldsymbol{\phi}(x) = \begin{bmatrix} 1 & \cos(x) & \sin(4x) \end{bmatrix}^T \quad \text{and} \quad \boldsymbol{w} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}^T. \tag{8.5}$$

Since the transformation by a basis function does not change the linearity in $\boldsymbol{w}$ the discussion in this chapter will be independent on the choice of basis functions. In practical terms, applying a basis functions to a dataset $\boldsymbol{X}$ to obtain the transformed dataset $\tilde{\boldsymbol{X}}$ is equivalent to applying feature transformations such as the ones we encountered in chapter 2

$$\tilde{\boldsymbol{x}}_i = \phi(\boldsymbol{x}_i), \quad \text{and} \quad \tilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^T \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^T \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^T \end{bmatrix}$$

and we can write the prediction of observation i as $y_i = \tilde{\boldsymbol{x}}_i^T \boldsymbol{w}$.

8.1.1 Training the linear regression model

To learn the parameters of the linear regression model, we will follow the general procedure outlined in section 6.5, in particular eq. (6.47). The first step of which is to come up with an expression for $p(y|\boldsymbol{x}, \boldsymbol{w})$ (see eq. (6.45)).

To this end, note the linear regression model eq. (8.3) is an ideal relationship between the input $\boldsymbol{x}$ and the target y. An actual observation will be noisy which we will capture with a noise parameter ε:

$$y = f(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon = \tilde{\boldsymbol{x}}^\top \boldsymbol{w} + \varepsilon$$

Since we don't know what ε is, we have to model it with our only tool for handling unknown quantities: probabilities. Therefore, assume that for each observation ε follows a normal distribution with mean 0 and variance σ^2. Using this assumption, the the probability density of y is then a normal distribution centered around $\tilde{\boldsymbol{x}}^\top \boldsymbol{w}$:

$$\begin{aligned} p(y|\boldsymbol{x}, \boldsymbol{w}, \sigma) &= \mathcal{N}(y - \tilde{\boldsymbol{x}}^\top \boldsymbol{w} | \mu = 0, \sigma^2) \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y - \tilde{\boldsymbol{x}}^\top \boldsymbol{w})^2}{2\sigma^2}} \end{aligned} \tag{8.6}$$

Our objective (see eq. (6.40)) is to find $\boldsymbol{w}$ as the value which is most plausible given the data, i.e. as maximizing $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$. Using Bayes' theorem this can be written as

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})}. \tag{8.7}$$

As we saw earlier (see eq. (6.46)), this is equivalent to selecting $\boldsymbol{w}^*$ as the value which *maximizes*

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \left[\log p(\boldsymbol{w}) + \sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) \right] \tag{8.8}$$

$$= \arg\max_{\boldsymbol{w}} \left[\log p(\boldsymbol{w}) - \frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} \left(y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w} \right)^2 \right]. \tag{8.9}$$

We will now assume the prior of $\boldsymbol{w}$ is flat, $p(\boldsymbol{w}) = 1$, which is also known as the *uniform* or *improper* prior[1], and furthermore we will choose to formulate the problem as a minimization problem by dropping constant terms and re-scaling the above expression. The problem of finding $\boldsymbol{w}^*$ is therefore equivalent to *minimizing* the *sum-of-squares* error function (compare to eq. (6.47)):

$$\begin{aligned} \boldsymbol{w}^* &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}) \\ \text{where } E(\boldsymbol{w}) &= \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \tilde{\boldsymbol{x}}_i^\top \boldsymbol{w} \right)^2. \end{aligned} \tag{8.10}$$

As we know from analysis, this can be accomplished by setting the derivative of E equal to zero and solving for $\boldsymbol{w}$. If we differentiate eq. (8.10) with respect to a weight w_j we obtain:

$$\frac{\partial}{\partial w_j} E(\boldsymbol{w}) = \frac{2}{N} \sum_{i=1}^{N} (y_i - \tilde{\boldsymbol{x}}_i^T \boldsymbol{w}) \tilde{X}_{ij} = \frac{2}{N} \sum_{i=1}^{N} y_i \tilde{X}_{ij} - \frac{2}{N} \left[\sum_{i=1}^{N} X_{ij} \tilde{\boldsymbol{x}}_i^T \right] \boldsymbol{w} \tag{8.11}$$

The gradient is therefore

[1] Notice the choice $p(\boldsymbol{w}) = 1$ strictly speaking does not make sense since the density is no longer normalized: $\int p(\boldsymbol{w}) d\boldsymbol{w} = \infty$. The prior can however be understood as saying that no particular value of $\boldsymbol{w}$ is preferred over another or more formally it can be understood as using the prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}\delta)$ throughout the derivation and then taking the limit $\delta \to \infty$.

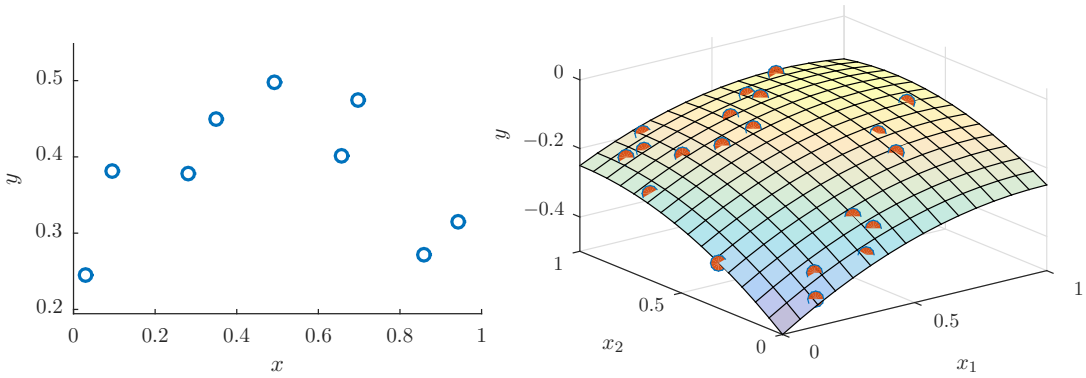


Fig. 8.3. Examples of two datasets for which we will apply linear regression. In the left-hand pane is a 1-d dataset comprised of x, in the right-hand side a 2d dataset comprised of red dots lying on a curved plane.

$$\nabla E(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial E(\boldsymbol{w})}{\partial w_1} \\ \vdots \\ \frac{\partial E(\boldsymbol{w})}{\partial w_M} \end{bmatrix} = \frac{2}{N}\tilde{\boldsymbol{X}}^T\boldsymbol{y} - \frac{2}{N}(\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})\boldsymbol{w}. \tag{8.12}$$

Setting the gradient equal to zero and solving we obtain

$$\boldsymbol{w}^* = (\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^T\boldsymbol{y} = (\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})\backslash\tilde{\boldsymbol{X}}^T\boldsymbol{y}. \tag{8.13}$$

Thus, training the linear regression model can be accomplished using one line of code in many computing environments. Since the linear regression model is so basic and important we will illustrate it in two scenarios in the following.

Example 1: Linear regression applied to a 1d dataset

Consider the 1d dataset shown in the left-pane of fig. 8.3. Suppose we wish to fit two models to the dataset, one corresponding to plain linear regression and the other to feature transforming the data to correspond to a second-degree polynomial.

For the first order polynomial linear regression case, this is accomplished by applying the (identity) feature transformation:

$$\tilde{\boldsymbol{X}}_{(1)} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{8.14}$$

then we compute $\boldsymbol{w}_{(1)} = (\tilde{\boldsymbol{X}}_{(1)}^T\tilde{\boldsymbol{X}}_{(1)})\backslash\tilde{\boldsymbol{X}}_{(1)}^T\boldsymbol{y}$ and and the red curve for an arbitrary point x can be predicted as $y = f(x, \boldsymbol{w}_{(1)}) = \begin{bmatrix}1 \; x\end{bmatrix}\boldsymbol{w}_{(1)}$. In the right-hand pane of fig. 8.4 we illustrate the second-degree polynomial linear regression corresponding to the feature transformation:

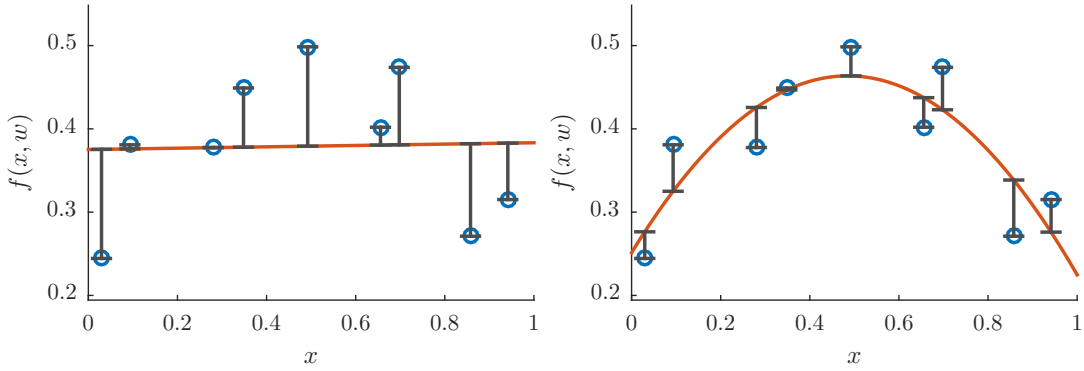


Fig. 8.4. Examples of applying the linear regression model to the dataset shown in the left-hand pane of fig. 8.3. The two panes respectively show a basic linear regression model $\boldsymbol{y} = \tilde{\boldsymbol{X}}_{(1)}\boldsymbol{w}_{(1)}$ without feature transformations, and linear regression model with feature transformation by adding the feature x^2 to produce a second-polynomial curve, $\boldsymbol{y} = \tilde{\boldsymbol{X}}_{(2)}\boldsymbol{w}_{(2)}$. See text for details.

$$\tilde{\boldsymbol{X}}_{(2)} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \\ 1 & x_N & x_N^2 \end{bmatrix} \tag{8.15}$$

then we compute $\boldsymbol{w}_{(2)} = (\tilde{\boldsymbol{X}}_{(2)}^T\tilde{\boldsymbol{X}}_{(2)})\backslash\tilde{\boldsymbol{X}}_{(2)}^T\boldsymbol{y}$ and the red curve for an arbitrary point x can be predicted as $y = f(x, \boldsymbol{w}_{(2)}) = \begin{bmatrix}1 & x & x^2\end{bmatrix}\boldsymbol{w}_{(2)}$.

Example 2: Linear regression applied to a 2d dataset

In the second example, we will consider the 2d dataset shown in the right-hand pane of fig. 8.3. We will again consider two models, one corresponding to plain linear regression and the other to feature transforming the data to correspond to a second order Taylor expansion.

In the left-hand pane of fig. 8.5 we illustrate the second-degree polynomial linear regression corresponding to the (trivial) feature transformation:

$$\tilde{\boldsymbol{X}}_{(1)} = \begin{bmatrix} 1 & X_{11} & X_{12} \\ 1 & X_{21} & X_{22} \\ \vdots & \vdots & \vdots \\ 1 & X_{N1} & X_{N2} \end{bmatrix}, \tag{8.16}$$

then we compute $\boldsymbol{w}_{(1)} = (\tilde{\boldsymbol{X}}_{(1)}^T\tilde{\boldsymbol{X}}_{(1)})\backslash\tilde{\boldsymbol{X}}_{(1)}^T\boldsymbol{y}$ (notice $\boldsymbol{w}$ is three-dimensional) and for an arbitrary point $\boldsymbol{x}$ we can predict $y = f(\boldsymbol{x}, \boldsymbol{w}_{(1)}) = \begin{bmatrix}1 & x_1 & x_2\end{bmatrix}\boldsymbol{w}_{(1)}$. This is used to generate the plane shown in the left-hand pane of fig. 8.5.

In the second example, we will attempt to fit a second-order expansion to the same dataset. This is accomplished by the feature transformation:

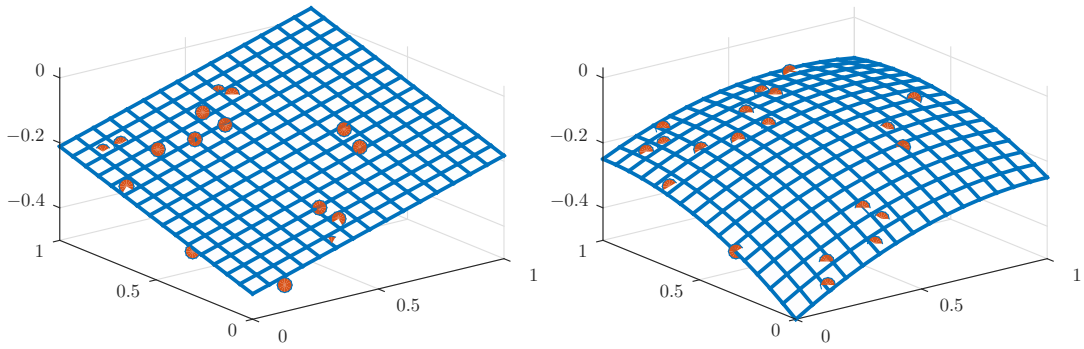


Fig. 8.5. Examples of applying the linear regression model to the dataset shown in the righ-hand pane of fig. 8.3. The left-hand pane shows the basic linear regression model $\boldsymbol{y} = \tilde{\boldsymbol{X}}_{(1)}\boldsymbol{w}$, and in the right-hand pane we make a feature transformation to include second order terms corresponding to $\boldsymbol{y} = \tilde{\boldsymbol{X}}_{(2)}\boldsymbol{w}$. See text for details.

$$\tilde{\boldsymbol{X}}_{(2)} = \begin{bmatrix} 1 & X_{11} & X_{12} & X_{11}^2 & X_{12}^2 & X_{11}X_{12} \\ 1 & X_{21} & X_{22} & X_{21}^2 & X_{22}^2 & X_{21}X_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{N1} & X_{N2} & X_{N1}^2 & X_{N2}^2 & X_{N1}X_{N2} \end{bmatrix} \tag{8.17}$$

Again, learning $\boldsymbol{w}$ (which is now six-dimensional!) can be accomplished as $\boldsymbol{w}_{(2)} = (\tilde{\boldsymbol{X}}_{(2)}^T\tilde{\boldsymbol{X}}_{(2)})\backslash\tilde{\boldsymbol{X}}_{(2)}^T\boldsymbol{y}$ and predictions, as shown in the right-hand pane of fig. 8.5, for a new point $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ can be made as

$$y = f(\boldsymbol{x}, \boldsymbol{w}_{(2)}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1x_2 \end{bmatrix} \boldsymbol{w}_{(2)}.$$

In the later case the found value of $\boldsymbol{w}_{(2)}$ is

$$\boldsymbol{w}_{(2)} = \begin{bmatrix} -0.5 & 0.5 & 0.5 & -0.25 & -0.25 & -0.125 \end{bmatrix}^T,$$

which is exactly (at this precision at least) equal to the value of $\boldsymbol{w}$ used to generate the data.

8.2 Logistic Regression

The goal of classification and regression may seem very different, however, it turns out linear regression can easily be extended to classification by the use of probabilities. Consider a binary classification task where $y = 0$ corresponds to the negative class and $y = 1$ to the positive class. We will now proceed exactly as we did in the case of linear regression by first finding an expression for

$$p(y|\boldsymbol{x}, \boldsymbol{w}) \tag{8.18}$$

and apply the maximum likelihood framework from section 6.5.

Since y is binary, our immediate idea should be to model it's density eq. (8.18) as a Bernoulli variable. However, as the output of a linear model is a general continuous number, and the parameter

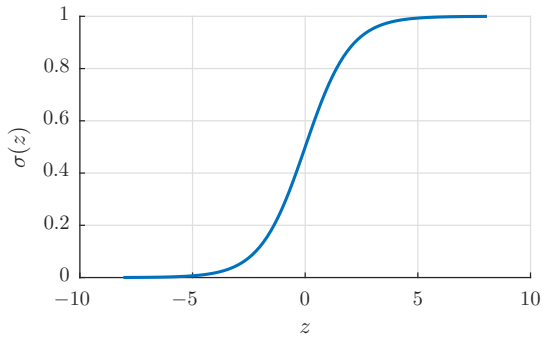


Fig. 8.6. The logistic sigmoid $\sigma(z) = (1 + e^{-z})^{-1}$. Notice as $z \to -\infty$ then $\sigma(z) \to 0$ and when $z \to \infty$ then $\sigma(z) \to 1$.

θ of the Bernoulli distribution belongs to the unit interval $[0, 1]$, we re-parameterize the Bernoulli distribution using the sigmoid function. Recall that according to eq. (5.30) from section 5.4.3 the Bernoulli density could be written as:

$$p(b|z) = \text{Bernouilli}(b|\theta = \sigma(z)) = \sigma(z)^b(1 - \sigma(z))^{1-b}, \quad \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{8.19}$$

Where the function $\sigma(z)$ is the *logistic sigmoid* and is shown in fig. 8.6. The way we will proceed is to define z as being equal to the output of a standard linear model $\tilde{\boldsymbol{x}}^\top \boldsymbol{w}$. Specifically, define:

$$\hat{y}_i = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})$$

The probability density of a given observation y_i is then:

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \text{Bernouilli}(y_i|\hat{y}_i) = \hat{y}_i^{y_i}(1 - \hat{y}_i)^{1-y_i}. \tag{8.20}$$

If we then proceed exactly as in the case of the linear regression model, by using eq. (8.20) and eq. (6.47), we see we should select the parameters $\boldsymbol{w}^*$ as the solution of the optimization problem:

$$\begin{aligned} \boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}) \\ \text{where:} \quad E(\boldsymbol{w}) &= -\frac{1}{N} \log \left[\prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) \right] \\ &= -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right], \quad \hat{y}_i = \sigma\left[\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w} \right], \end{aligned} \tag{8.21}$$

However, at this point our discussion has to departs from linear regression: there is no closed-form analytical solution for the minimum of the error function. How we find $\boldsymbol{w}^*$ in practice will be discussed later in the chapter for neural networks where we will consider a general method for minimizing error functions, however, until then simply rely on the commands build into your computing environment for solving logistic regression problems.

As mentioned, all the tricks of feature-transforming $\boldsymbol{X}$ also "work" for logistic regression and we will therefore only consider two simple examples where $\boldsymbol{X}$ has not had feature-transformations

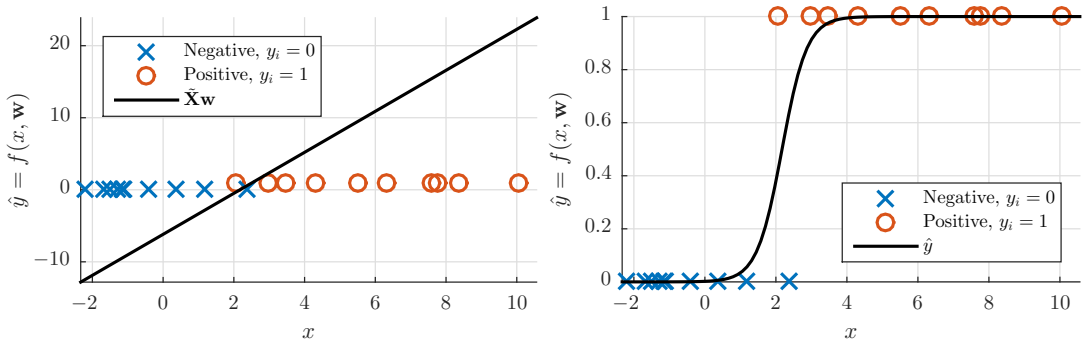


Fig. 8.7. A one-dimensional logistic regression example. The left-hand pane shows the intermediate (linear) output $\boldsymbol{X}\boldsymbol{w}$ and the right-hand pane the true logistic-regression decision boundary $\hat{y} = \sigma(\boldsymbol{X}\boldsymbol{w})$.

applied to it asides being pre-fixed with 1s as is required for any regression problem. In fig. 8.7 is shown a basic logistic regression example for a simple 1-dimensional dataset. The procedure is exactly similar to linear regression. We first define:

$$\tilde{\boldsymbol{X}}_{(1)} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{8.22}$$

Then, the left-hand pane shown the intermediate linear regression value $z = \tilde{\boldsymbol{X}}_{(1)}\boldsymbol{w}$ as the black line, and the right-hand pane the predicted probabilities $\hat{y} = \sigma(\tilde{\boldsymbol{X}}_{(1)}\boldsymbol{w})$ are plotted as a black line. As expected for such a simple problem the logistic regression learns how to separate the two classes. For completeness, fig. 8.8 illustrates a 2d example, where the two classes are fitted with a logistic regression model and the decision surface $\hat{y}$ is shown. In the example, the class-membership probabilities are computed as:

$$\hat{y}_i = \sigma(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w}) = \sigma\left(\begin{bmatrix} 1 & X_{i1} & X_{21} \end{bmatrix} \boldsymbol{w}\right).$$

Notice the decision boundary is linear and quite steep. Logistic regression will have linear decision boundaries unless we apply (non-linear) feature transformations to our dataset.

8.2.1 The confusion matrix

While the error function $E(\boldsymbol{w})$ could be used to evaluate a logistic regression model it is important to keep in mind that the error function measures the *probability* of the data, however, at the end of the day, we are probably more interested in how often the logistic regression model classifies correctly and how often it classifies wrongly. To this end, we must turn the output of the logistic regression (which is a probability) into a class label. This can be accomplished by simply thresholding at 0.5 (we will return to how the threshold should be selected in chapter 10) and predict that observation i belongs to the positive class if $\hat{y}_i > \frac{1}{2}$ and the negative class if $\hat{y}_i \leq \frac{1}{2}$.

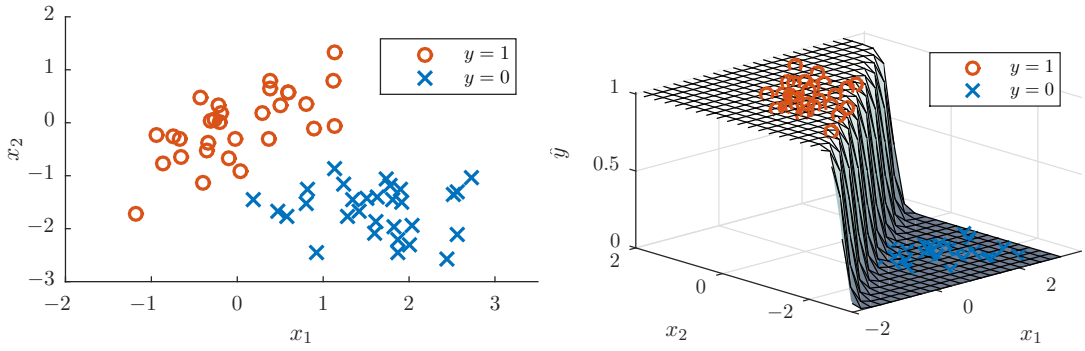


Fig. 8.8. 2d logistic regression example. The dataset shown in the left-hand pane is fitted with a logistic regression model and the class-membership prediction $\hat{y}$ is shown in the right-hand pane.

There are now four different combinations of what class an observation actually belongs to and what it is predicted to belong to by the model. They are called:

True Positives, TP: Number of observations which are in fact positive $y_i = 1$ which the classifier correctly labels as positive $\hat{y}_i > \frac{1}{2}$

False Positives, FP: Number of observations which are in fact negative $y_i = 0$ which the classifier incorrectly labels as positive $\hat{y}_i > \frac{1}{2}$

False Negatives, FN: Number of observations which are in fact positive $y_i = 1$ which the classifier incorrectly labels as negative $\hat{y}_i < \frac{1}{2}$

True Negatives, TN: Number of observations which are in fact negative $y_i = 0$ which the classifier correctly labels as negative $\hat{y}_i < \frac{1}{2}$

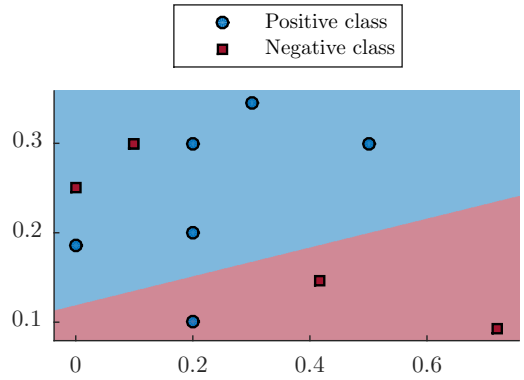
These are illustrated in fig. 8.9. In the left-hand pane is shown a binary classification problem of $N = 10$ observations where the colors indicate the predictions made by a logistic regression model (blue corresponds to the positive class and red to the negative). In the right-hand pane the true positives, false negatives, etc. are collected in what is known as the *confusion matrix*, where the inserted ticks on colored background indicate which observations counts towards which numbers. As mentioned we will return to the meaning of these numbers in more detail in chapter 10, however, for now we will simply focus on *how often the classifier is right* which is known as the *accuracy* (or equivalently *how often the classifier is wrong* which is known as the *error rate*):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N}, \quad \text{Error rate} = \frac{\text{FP} + \text{FN}}{N} = 1 - \text{Accuracy}.$$

For instance the accuracy of the logistic regression model in fig. 8.9 is $\frac{5+2}{N} = \frac{7}{10}$.

8.3 The general linear model⋆

The overall form of the cost-function in the linear and logistic regression models can be generalized into what is known as the *general linear model*. Since this is the form of these models which is mostly commonly encountered in a computing environment, we will briefly discuss it here. Basically,



$N = 10$	Predicted **Positive**	Predicted **Negative**
Actually **Positive** $N^+ = 6$	TP = 5 (True Positive)	FN = 1 (False Negative)
Actually **Negative** $N^- = 4$	FP = 2 (False Positive)	TN = 2 (True Negative)

Fig. 8.9. (Left:) A small $N = 10$ observation binary classification problem. The colors indicate the prediction made by a logistic regression classifier obtained by thresholding $\hat{y}_i$ at $\frac{1}{2}$. (Right:) The confusion matrix of the classifier in the left-hand pane. The inserts (ticks on background) indicate which observations counts towards which numbers in the confusion matrix.

it decompose the model into two parts: a *link* function g and a *cost function* d. It then assumes the output of the model is:

$$y = f(\boldsymbol{x}, \boldsymbol{w}) = g(\tilde{\boldsymbol{x}}^\top \boldsymbol{w})$$

and that the cost function can be written as:

$$E(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} d(y_i, g(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})) \tag{8.23}$$

Parameters are found in the usual way by minimizing $E(\boldsymbol{w})$. For a summary, see Box 8.3.1.

Method 8.3.1: The general linear model

Given a dataset consisting of N pairs $(\boldsymbol{x}_i, y_i)$, we first define $\tilde{\boldsymbol{x}}_i$ as a feature-transformation of $\boldsymbol{x}_i$, in the simplest form obtained by pre-fixing $\boldsymbol{x}_i$ with a constant intercept term:

$$\tilde{\boldsymbol{x}}_i = \begin{bmatrix} 1 & \boldsymbol{x}_i^\top \end{bmatrix}^\top.$$

The predicted output $\hat{y}$, cost function E and learned parameters $\boldsymbol{w}^*$ in a general linear model (GLM) are then defined as:

$$\begin{aligned} \hat{y} &= g(\tilde{\boldsymbol{x}}^\top \boldsymbol{w}) \\ E(\boldsymbol{w}) &= \frac{1}{N} \sum_{i=1}^{N} d(y_i, g(\tilde{\boldsymbol{x}}_i^\top \boldsymbol{w})) \\ \boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}). \end{aligned}$$

The *linear regression model* can be recovered by defining

$$g(\tilde{\boldsymbol{x}}^\top \boldsymbol{w}) = \tilde{\boldsymbol{x}}^\top \boldsymbol{w} \quad \text{and} \quad d(y, \hat{y}) = (y - \hat{y})^2$$

and the *logistic regression model* as:

$$g(\tilde{\boldsymbol{x}}^\top \boldsymbol{w}) = \sigma(\tilde{\boldsymbol{x}}^\top \boldsymbol{w}) \quad \text{and} \quad d(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}).$$