Contents

1 Logistic Regression ................................. 2
  1.1 Maximum Likelihood Estimation 5
  1.2 Multi-class Logistic Regression 9
  1.3 Further Reading 13
  1.4 Exercises 14
In logistic regression, we are given a set of \( d \) predictor or independent variables \( X_1, X_2, \cdots, X_d \), and a binary or Bernoulli response variable \( Y \) that takes on only two values, namely, 0 and 1. Thus, we are given a training dataset \( D = \{x_i, y_i\}_{i=1,\cdots,n} \), where \( x_i \in \mathbb{R}^d \) and \( y_i \in \{0, 1\} \). As done in ??, we augment the data matrix \( D \) by adding a new dimension \( X_0 \) that is always fixed at the value 1 for each point, so that henceforth, each point \( x_i = (1, x_{1i}, x_{2i}, \cdots, x_{di})^T \in \mathbb{R}^{d+1} \), and the multivariate random vector \( X \), comprising all the independent attributes, is given as \( X = (X_0, X_1, \cdots, X_d)^T \).

Since there are only two outcome for \( Y \), the probability mass function for \( Y \) given \( X \) is:

\[
P(Y = 1 | X) = \pi \quad \quad P(Y = 0 | X) = 1 - \pi
\]

where \( \pi \) is the unknown parameter denoting the probability of \( Y = 1 \) that needs to be estimated. Further, the expected value of \( Y \) is

\[
E[Y] = 1 \cdot P(Y = 1) + 0 \cdot P(Y = 0) = P(Y = 1) = \pi
\]

In logistic regression, instead of directly predicting the response value, the goal is to learn the probability \( P(Y = 1 | X) \), or the expected value of \( Y \) given \( X \).

Since we want to predict the probability \( P(Y = 1 | X) \) it is not appropriate to directly use the linear regression model

\[
f(X) = w^T X = w_0 \cdot X_0 + w_1 \cdot X_1 + w_2 \cdot X_2 + \cdots + w_d \cdot X_d
\]

where \( w = (w_0, w_1, \cdots, w_d)^T \in \mathbb{R}^{d+1} \) with \( w_0 \) the unknown bias term, and \( w_i \) the unknown regression coefficient or weight for attribute \( X_i \). The reason is that \( f(X) \) can be arbitrarily large or arbitrarily small, whereas for logistic regression, we require that the output represents a probability value, and thus we need a model that results in an output that lies in the interval \([0, 1]\). The name “logistic regression” comes from the logistic function (also called the sigmoid function) that meets this requirement. It is defined as follows:

\[
\theta(z) = \frac{1}{1 + \exp[-z]} = \frac{\exp[z]}{1 + \exp[z]} \quad \quad (1.1)
\]
The logistic function “squashes” the output to be between 0 and 1 for any scalar input \( z \) (see Fig. 1.1). The output of \( \theta \) can therefore be interpreted as a probability.

**Example 1.1 (Logistic Function).** Fig. 1.1 shows the plot for the logistic function of values of \( z \) ranging from \(-\infty\) to \(+\infty\). In particular consider what happens when \( z \) is \(-\infty\), \(+\infty\) and 0; we have

\[
\begin{align*}
\theta(-\infty) &= \frac{1}{1 + \exp\{\infty\}} = \frac{1}{\infty} = 0 \\
\theta(+\infty) &= \frac{1}{1 + \exp\{-\infty\}} = \frac{1}{1} = 1 \\
\theta(0) &= \frac{1}{1 + \exp\{0\}} = \frac{1}{2} = 0.5
\end{align*}
\]

As desired, \( \theta(z) \) lies in the range \([0, 1]\), and \( z = 0 \) is the “threshold” value in the sense that for \( z > 0 \) we have \( \theta(z) > 0.5 \), and for \( z < 0 \), we have \( \theta(z) < 0.5 \). Thus, interpreting \( \theta(z) \) as a probability, the larger the \( z \) value, the higher the probability.

Another interesting property of the logistic function is that

\[
1 - \theta(z) = 1 - \frac{\exp[z]}{1 + \exp[z]} = \frac{1 + \exp[z] - \exp[z]}{1 + \exp[z]} = \frac{1}{1 + \exp[z]} = \theta(-z) \quad (1.2)
\]
Using the logistic function, we can now define the logistic regression model as follows:

\[ P(Y = 1|X) = \frac{\exp(w^T X)}{1 + \exp(w^T X)} \]  

(1.3)

Thus, the probability that the response is \( Y = 1 \) is the output of the logistic function for the input \( w^T X \), which in turn is linear in \( X \). On the other hand, the probability for \( Y = 0 \) is given as

\[ P(Y = 0|X) = 1 - P(Y = 1|X) = \theta(-w^T X) = \frac{1}{1 + \exp(-w^T X)} \]

where we used Eq. (1.2) to note that \( 1 - \theta(z) = \theta(-z) \) for \( z = w^T X \).

Combining these two cases the full logistic regression model is given as

\[ P(Y|X) = \theta(w^T X)^Y \cdot \theta(-w^T X)^{1-Y} \]  

(1.4)

since \( Y \) is a Bernoulli random variable that takes on either the value 1 or 0. We can observe that \( P(Y|X) = \theta(w^T X) \) when \( Y = 1 \) and \( P(Y|X) = \theta(-w^T X) \) when \( Y = 0 \) as desired.

**Log-Odds Ratio** Define the *odds ratio* for the occurrence of \( Y = 1 \) as follows:

\[ \text{odds}(Y = 1) = \frac{P(Y = 1)}{P(Y = 0)} = \frac{P(Y = 1)}{1 - P(Y = 1)} \]

\[ = \frac{\theta(w^T X)}{1 - \theta(w^T X)} = \frac{1}{1 + \exp(-w^T X)} \cdot \frac{1 + \exp(-w^T X)}{\exp(-w^T X)} \]

\[ = \exp(w^T X) \]

Thus, the logarithm of the odds ratio, called the *log-odds ratio*, is given as:

\[ \ln(\text{odds}(Y = 1)) = \ln \left( \frac{P(Y = 1)}{1 - P(Y = 1)} \right) \]

\[ = \ln(\exp(w^T X)) \]

\[ = w^T X = w_0 \cdot X_0 + w_1 \cdot X_1 + \cdots + w_d \cdot X_d \]  

(1.5)

The log odd ratio function \( \ln(\text{odds}(Y = 1)) \) is also called the *logit* function. It is the inverse of the logistic function.

The logistic regression model is therefore based on the assumption that the log-odds ratio for \( Y = 1 \) is given as a linear function (or a weighted sum) of independent attributes. In particular, let us consider the effect of attribute \( X_i \) by fixing the values for all other attributes; we have

\[ \ln(\text{odds}(Y = 1)) = w_i \cdot X_i + C \]

\[ \implies \text{odds}(Y = 1) = \exp(w_i \cdot X_i + C) = \exp(w_i \cdot X_i) \cdot \exp(C) \propto \exp(w_i \cdot X_i) \]

where \( C \) is some constant comprising the fixed attributes. The regression coefficient \( w_i \) can then be interpreted as the change in the log-odds ratio for \( Y = 1 \) for a unit change in \( X_i \), or equivalently the odds ratio for \( Y = 1 \) increase exponentially per unit change in \( X_i \).
1.1 Maximum Likelihood Estimation

In this section, we describe the maximum likelihood approach to estimate the weight vector \( \mathbf{w} \). Likelihood is defined as the probability of the observed data given the parameters \( \mathbf{w} \). We assume that the binary response variables \( y_i \) are conditionally independent given \( \mathbf{x}_i \). Therefore, the likelihood of the observed responses is given as

\[
L(\mathbf{w}) = \prod_{i=1}^{n} P(y_i | \mathbf{x}_i) = \prod_{i=1}^{n} \theta((\mathbf{w}^T \mathbf{x}_i)^{y_i} \cdot (1 - \theta((\mathbf{w}^T \mathbf{x}_i)^{1-y_i})
\]

Instead of trying to maximize the likelihood, we can maximize the logarithm of the likelihood, called log-likelihood, to convert the product into a summation as follows:

\[
\ln(L(\mathbf{w})) = \sum_{i=1}^{n} y_i \cdot \ln \left( \theta((\mathbf{w}^T \mathbf{x}_i)) \right) + (1 - y_i) \cdot \ln \left( \frac{1}{1 - \theta((\mathbf{w}^T \mathbf{x}_i))} \right)
\]

(1.6)

Finally, the negative of the log-likelihood can be considered as an error function, which yields the cross-entropy error function, given as follows:

\[
E(\mathbf{w}) = - \ln(L(\mathbf{w})) = \sum_{i=1}^{n} y_i \cdot \ln \left( \frac{1}{\theta((\mathbf{w}^T \mathbf{x}_i))} \right) + (1 - y_i) \cdot \ln \left( \frac{1}{1 - \theta((\mathbf{w}^T \mathbf{x}_i))} \right)
\]

(1.7)

Thus, the task of maximum likelihood estimation can be also treated as cross-entropy minimization. However, we will directly maximize the log-likelihood function.

Typically, to obtain the optimal weight vector \( \mathbf{w} \), i.e., the optimal values of the parameters \( w_0, w_1, \ldots, w_d \), we would differentiate the log-likelihood function and set the result to 0 to solve for \( \mathbf{w} \). However, for the log-likelihood formulation in Eq. (1.6) there is no closed form solution to compute the weight vector \( \mathbf{w} \). Instead, we will use an iterative gradient ascent method to compute the optimal value, since Eq. (1.6) is a concave function, and has a unique global optimal.

The gradient ascent method relies on the gradient of the log-likelihood function, which can be obtained by taking the partial derivative of log-likelihood with respect to \( \mathbf{w} \), as follows:

\[
\nabla(\mathbf{w}) = \frac{\partial}{\partial \mathbf{w}} \ln(L(\mathbf{w})) = \frac{1}{\theta(z_i)} \cdot \sum_{i=1}^{n} y_i \cdot \ln(\theta(z_i)) + (1 - y_i) \cdot \ln(\theta(z_i))
\]

(1.8)

where \( z_i = \mathbf{w}^T \mathbf{x}_i \). We use the chain rule to obtain the derivative of \( \ln(\theta(z_i)) \) with respect to \( \mathbf{w} \). We note that following facts:

\[
\frac{\partial}{\partial \theta(z_i)} \ln(\theta(z_i)) = \frac{1}{\theta(z_i)}
\]

\[
\frac{\partial}{\partial z_i} \theta(z_i) = \theta(z_i) \cdot (1 - \theta(z_i)) = \theta(z_i) \cdot \theta(-z_i)
\]

\[
\frac{\partial}{\partial \mathbf{w}} z_i = \frac{\partial}{\partial \mathbf{w}} \mathbf{w}^T \mathbf{x}_i = \mathbf{x}_i
\]
As per the chain rule, we have
\[
\frac{\partial}{\partial \mathbf{w}} \ln(\theta(z_i)) = \frac{\partial}{\partial \theta(z_i)} \ln(\theta(z_i)) \cdot \frac{\partial}{\partial z_i} \theta(z_i) \cdot \frac{\partial}{\partial \mathbf{w}} z_i
\]
\[
= \frac{1}{\theta(z_i)} \cdot (\theta(z_i) \cdot \theta(-z_i)) \cdot \mathbf{x}_i
\]
\[
= \theta(-z_i) \cdot \mathbf{x}_i
\] (1.9)

Likewise, using the chain rule, we have
\[
\frac{\partial}{\partial \mathbf{w}} \ln(\theta(-z_i)) = -\theta(z_i) \cdot \mathbf{x}_i
\]

Substituting the above into Eq. (1.8) we get
\[
\nabla(\mathbf{w}) = \sum_{i=1}^{n} y_i \cdot \theta(-z_i) \cdot \mathbf{x}_i - (1 - y_i) \cdot \theta(z_i) \cdot \mathbf{x}_i
\]
\[
= \sum_{i=1}^{n} y_i \cdot (\theta(-z_i) + \theta(z_i)) \cdot \mathbf{x}_i - \theta(z_i) \cdot \mathbf{x}_i
\]
\[
= \sum_{i=1}^{n} (y_i - \theta(z_i)) \cdot \mathbf{x}_i
\]
\[
= \sum_{i=1}^{n} (y_i - \theta(\mathbf{w}^T \mathbf{x}_i)) \cdot \mathbf{x}_i
\] (1.10)

The gradient ascent method starts at some initial estimate for \( \mathbf{w} \), denoted \( \mathbf{w}_0 \). At each step, the method moves in the direction of steepest ascent, which is given by the gradient vector. Thus, given the current estimate \( \mathbf{w}_t \), we can obtain the next estimate as follows:
\[
\mathbf{w}_{t+1} = \mathbf{w}_t + \delta \cdot \nabla(\mathbf{w}_t)
\] (1.11)

Here \( \delta > 0 \) is a user-specified parameter called the *learning rate*. It should not be too large, otherwise the estimates will vary wildly from one iteration to the next, and it should not be too small, otherwise it will take a long time to converge. At the optimal value of \( \mathbf{w} \), the gradient will be zero, i.e., \( \nabla(\mathbf{w}) = \mathbf{0} \) as desired.

**Stochastic Gradient Ascent** The gradient ascent method computes the gradient by considering all the data points. For large datasets, it is typically much faster to compute the gradient by considering only one (randomly chosen) point at a time. The weight vector is updated after each such partial gradient step, giving rise to the *stochastic gradient ascent* method of computing the optimal weight vector \( \mathbf{w} \). Given a randomly chosen point \( \mathbf{x}_i \), we already computed the point-specific gradient in Eq. (1.10), which we re-write as:
\[
\nabla(\mathbf{w}, \mathbf{x}_i) = (y_i - \theta(\mathbf{w}^T \mathbf{x}_i)) \cdot \mathbf{x}_i
\]
Algorithm 1.1: Logistic Regression Algorithm: Stochastic Gradient Ascent

**LOGISTIC-REGRESSION-SGA (D, δ, ε):**

1. foreach \( x_i \in D \) do \( x_i \leftarrow \begin{pmatrix} 1 \\ x_i \end{pmatrix} \) // map to \( \mathbb{R}^{d+1} \)
2. \( t \leftarrow 0 \) // step/iteration counter
3. \( w_0 \leftarrow (0, \ldots, 0)^T \) // initial weight vector
4. repeat
5. foreach \( x_i \in D \) in random order do
6. \( \nabla (w_t, x_i) \leftarrow (y_i - \theta(w_T x_i)) \cdot x_i \) // compute gradient for \( x_i \)
7. \( w_{t+1} \leftarrow w_t + \delta \cdot \nabla (w_t, x_i) \) // update estimate for \( w \)
8. \( t \leftarrow t + 1 \)
9. until \( \|w_t - w_{t-1}\| \leq \epsilon \)

As in the batch approach, in stochastic gradient ascent we update the weight vector as follows:

\[
w_{t+1} = w_t + \delta \cdot \nabla (w_t, x_i) \quad (1.12)
\]

The pseudo-code for the stochastic gradient ascent (SGA) method for logistic regression is shown in Algorithm 1.1.

Once the model has been trained, we can predict the class for any new test point \( z \) as follows:

\[
\hat{y} = \begin{cases} 1 & \text{if } \pi = \theta(w^T z) \geq 0.5 \\ 0 & \text{if } \pi < 0.5 \end{cases}
\]

**Example 1.2 (Logistic Regression).** Fig. 1.2a shows the output of logistic regression modeling on the Iris principal components data, where the independent attributes \( X_1 \) and \( X_2 \) represent the first two principal components, and the binary response variable \( Y \) represents the type of Iris flower; \( Y = 1 \) corresponds to Iris-virginica, whereas \( Y = 0 \) corresponds to the two other Iris types, namely Iris-setosa and Iris-versicolor.

The fitted logistic model is given as

\[
w = (w_0, w_1, w_2)^T = (-6.79, -5.07, -3.29)^T
\]

\[
P(Y = 1|x) = \theta(w^T x) = \frac{1}{1 + \exp(6.79 + 5.07 \cdot x_1 + 3.29 \cdot x_2)}
\]

Fig. 1.2a plots \( P(Y = 1|x) \) for various values of \( x \).

Given \( x \), if \( P(Y = 1|x) \geq 0.5 \), then we predict \( \hat{y} = 1 \), otherwise we predict \( \hat{y} = 0 \). Fig. 1.2a shows that five points (shown in dark gray) are misclassified. For example, for \( x = (1, -0.52, -1.19)^T \) we have:

\[
P(Y = 1|x) = \theta(w^T x) = \theta(-0.24) = 0.44
\]

\[
P(Y = 0|x) = 1 - P(Y = 1|x) = 0.54
\]
Thus, the predicted response for \( \mathbf{x} \) is \( \hat{y} = 0 \), whereas the true class is \( Y = 1 \).

Fig. 1.2 also contrasts logistic versus linear regression. The plane of best fit in linear regression is shown in Fig. 1.2b, with the weight vector:

\[
\mathbf{w} = (0.333, -0.167, 0.074)^T
\]

\[
\hat{y} = f(\mathbf{x}) = 0.333 - 0.167 \cdot x_1 + 0.074 \cdot x_2
\]

Since the response vector \( Y \) is binary, we predict the response class as \( Y = 1 \) if \( f(\mathbf{x}) \geq 0.5 \), and \( Y = 0 \) otherwise. The linear regression model results in 17 points being misclassified (dark gray points), as shown in Fig. 1.2b.

Since there are \( n = 150 \) points in total, this results in a training set or in-sample accuracy of 88.7% for linear regression. On the other hand logistic regression misclassifies only 5 points, for an in-sample accuracy of 96.7%, which is a much better fit, as is also apparent in Fig. 1.2.
1.2 Multi-class Logistic Regression

1.2 MULTI-CLASS LOGISTIC REGRESSION

We now generalize logistic regression to the case when the response variable $Y$ can take on $K$ distinct nominal categorical values called classes, i.e., $Y \in \{c_1, c_2, \ldots, c_K\}$. We model $Y$ as a $K$-dimensional multivariate Bernoulli random variable (see ??). Since $Y$ can assume only one of the $K$ values, we use the one-hot encoding approach to map each categorical value $c_i$ to the $K$-dimensional binary vector $e_i = (0, \ldots, 0, 1, \ldots, 0)^T$ all of whose $i$th element $e_{ii} = 1$, and all other elements $e_{ij} = 0$, so that $\sum_{j=1}^K e_{ij} = 1$. Henceforth, we assume that the categorical response variable $Y$ is a multivariate Bernoulli variable $Y \in \{e_1, e_2, \ldots, e_K\}$, and by $Y_j$ we refer to the $j$th component of $Y$.

The probability mass function for $Y$ is given as

$$P(Y = e_i) = \pi_i$$

where $\pi_i$ is the (unknown) probability of observing class $c_i$. Thus, there are $K$ unknown parameters, which must satisfy the following constraint:

$$\sum_{i=1}^K \pi_i = \sum_{i=1}^K P(Y = e_i) = 1$$

Given that only one element of $e_i$ is 1, the probability function can be written compactly as

$$P(Y = e_i) = \prod_{j=1}^K (\pi_j)^{Y_j}$$

Note that since $Y = e_i$, only $Y_i = 1$ and the rest of its elements $Y_j = 0$ for $j \neq i$.

In multi-class logistic regression, we select one of the values, say $c_K$, as a reference or base class, and consider the log-odds ratio of all other values with respect to $c_K$; we assume that each of these log odd ratios are linear in $X$, but with a different weight vectors $w_i$, for class $c_i$. That is, the log-odds ratio of class $c_i$ with respect to class $c_K$ is assumed to satisfy

$$\ln(\text{odds}(Y = e_i)) = \ln \left( \frac{P(Y = e_i)}{P(Y = e_K)} \right) = \ln \left( \frac{\pi_i}{\pi_K} \right) = w_i^T X = w_{i0} \cdot X_0 + w_{i1} \cdot X_1 + \cdots + w_{id} \cdot X_d$$

(1.13)

We can rewrite the above set of equations as follows:

$$\frac{\pi_i}{\pi_K} = \exp(w_i^T X)$$

$$\pi_i = \exp(w_i^T X) \cdot \pi_K$$

(1.14)
Logistic Regression

Given that \( \sum_{i=1}^{K} \pi_i = 1 \), we have
\[
\sum_{i=1}^{K} \pi_i = 1
\]
\[
\Rightarrow \left( \sum_{i \neq K} \exp(w_i^T X) \cdot \pi_i \right) + \pi_K = 1
\]
\[
\Rightarrow \pi_K = \frac{1}{1 + \sum_{i \neq K} \exp(w_i^T X)}
\]

Substituting the above into Eq. (1.14), we have
\[
\pi_i = \exp(w_i^T X) \cdot \pi_K = \frac{\exp(w_i^T X)}{1 + \sum_{i \neq K} \exp(w_i^T X)}
\]

Finally, setting \( w_K = 0 \), we have \( \exp(w_K^T X) = 1 \), and thus we can write the full model for nominal logistic regression as follows:
\[
\pi_i = \frac{\exp(w_i^T X)}{\sum_{j=1}^{K} \exp(w_j^T X)}, \quad \text{for all } i = 1, 2, \ldots, K \tag{1.15}
\]

This function is also called the \textit{softmax} function. When \( K = 2 \), this formulation yields exactly the same model as in binary logistic regression.

It is also interesting to note that the choice of the reference class is not important, since we can derive the log-odds ratio for any two classes \( c_i \) and \( c_j \) as follows:
\[
\ln \left( \frac{\pi_i}{\pi_j} \right) = \ln \left( \frac{\pi_i}{\pi_K} \cdot \frac{\pi_K}{\pi_j} \right)
\]
\[
= \ln \left( \frac{\pi_i}{\pi_K} \right) + \ln \left( \frac{\pi_K}{\pi_j} \right)
\]
\[
= \ln \left( \frac{\pi_i}{\pi_K} \right) - \ln \left( \frac{\pi_j}{\pi_K} \right)
\]
\[
= w_i^T X - w_j^T X
\]
\[
= (w_i - w_j)^T X
\]

That is, the log-odds ratio between any two classes can be computed from the difference of the corresponding weight vectors.

**Maximum Likelihood Estimation** To find the \( K \) sets of regression parameter vectors \( w_i \in \mathbb{R}^{d+1} \) we will one again use gradient ascent approach to maximize the log-likelihood function. For a given point \( x_i \), let \( y_i = e_a \), and let \( y_{ij} \) denote the \( j \)th element of \( y_i \) treated as a multivariate Bernoulli random vector. In this case, \( y_{ia} = 1 \) and \( y_{ij} = 0 \) for all \( j \neq a \).

The likelihood of the data is given as
\[
L(W) = \prod_{i=1}^{n} P(y_i = e_a) = \prod_{i=1}^{n} \prod_{j=1}^{K} (\pi_j)^{y_{ij}}
\]
1.2 Multi-class Logistic Regression

where \( \mathbf{W} = \{ \mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K \} \). The log-likelihood is then given as:

\[
\ln (L(\mathbf{W})) = \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \cdot \ln(\pi_j) = \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \cdot \ln \left( \frac{\exp[\mathbf{w}_j^T \mathbf{x}_i]}{\sum_{a=1}^{K} \exp[\mathbf{w}_a^T \mathbf{x}_i]} \right) \tag{1.16}
\]

We note the following facts:

\[
\frac{\partial}{\partial \pi_j} \ln(\pi_j) = \frac{1}{\pi_j}
\]

\[
\frac{\partial}{\partial \mathbf{w}_a} \pi_a = \pi_a \cdot (1 - \pi_a) \cdot \mathbf{x}_i
\]

\[
\frac{\partial}{\partial \mathbf{w}_a} \pi_j = -\pi_a \cdot \pi_j \cdot \mathbf{x}_i \quad \text{for } j \neq a
\]

Let us consider the gradient of the log-likelihood with respect to the weight vector \( \mathbf{w}_a \):

\[
\nabla(\mathbf{w}_a) = \frac{\partial}{\partial \mathbf{w}_a} \ln(L(\mathbf{W}))
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \cdot \frac{\partial}{\partial \pi_j} \ln(\pi_j) \cdot \frac{\partial}{\partial \mathbf{w}_a} \pi_j
\]

\[
= \sum_{i=1}^{n} \left( \sum_{j \neq a} y_{ij} \cdot \left( \frac{1}{\pi_j} \cdot (-\pi_a \cdot \pi_j) \cdot \mathbf{x}_i \right) + y_{ia} \cdot \frac{1}{\pi_a} \cdot \pi_a \cdot (1 - \pi_a) \cdot \mathbf{x}_i \right)
\]

\[
= \sum_{i=1}^{n} \left( y_{ia} - y_{ia} \cdot \pi_a - \sum_{j \neq a} y_{ij} \cdot \pi_a \right) \cdot \mathbf{x}_i
\]

\[
= \sum_{i=1}^{n} \left( y_{ia} - \sum_{j=1}^{K} y_{ij} \cdot \pi_a \right) \cdot \mathbf{x}_i
\]

\[
= \sum_{i=1}^{n} (y_{ia} - \pi_a) \cdot \mathbf{x}_i
\]

The last step follows from the fact that \( \sum_{j=1}^{K} y_{ij} = 1 \), since only one element of \( y_i \) can be 1.

For stochastic gradient ascent, we update the weight vectors by considering only one point at a time. The gradient of the log-likelihood function with respect to \( \mathbf{w}_j \) at a given point \( \mathbf{x}_i \) is given as

\[
\nabla(\mathbf{w}_j, \mathbf{x}_i) = (y_{ij} - \pi_j) \cdot \mathbf{x}_i \tag{1.17}
\]

which results in the following update rule for the \( j \)th weight vector:

\[
\mathbf{w}_{j,t+1} = \mathbf{w}_{j,t} + \delta \cdot \nabla(\mathbf{w}_{j,t}, \mathbf{x}_i)
\]

where \( \mathbf{w}_{j,t} \) denotes the estimate of \( \mathbf{w}_j \) at step \( t \). The pseudo-code for the stochastic gradient ascent method for multi-class logistic is shown in Algorithm 1.2.
Once the model has been trained, we can predict the class for any new test point \( z \) as follows:

\[
\hat{y} = \arg \max_{c_i} \{ \pi_i \} = \arg \max_{c_i} \left\{ \frac{\exp[w_i^T z]}{\sum_{j=1}^K \exp[w_j^T z]} \right\}
\]

(1.18)

That is, we compute the softmax for each class, and then predict the class of \( z \) and the one with the highest probability.
Example 1.3. Consider the Iris dataset, with \( n = 150 \) points in a 2D space spanned by the first two principal components, as shown in Fig. 1.3. Here the response variable takes on three values: \( Y = c_1 \) corresponds to Iris-setosa (shown as squares), \( Y = c_2 \) corresponds to Iris-versicolor (as circles) and \( Y = c_3 \) corresponds to Iris-virginica (as triangles). Thus, we map \( Y = c_1 \) to \( e_1 = (1, 0, 0)^T \), \( Y = c_2 \) to \( e_2 = (0, 1, 0)^T \) and \( Y = c_3 \) to \( e_3 = (0, 0, 1)^T \).

The multi-class logistic model uses \( Y = c_3 \) (Iris-virginica; triangles) as the reference or base class. The fitted model is given as:

\[
\begin{align*}
\mathbf{w}_1 &= (-3.52, 3.62, 2.61)^T \\
\mathbf{w}_2 &= (-6.95, -5.18, -3.40)^T \\
\mathbf{w}_3 &= (0, 0, 0)^T
\end{align*}
\]

Fig. 1.3 plots the decision surfaces corresponding to the softmax functions:

\[
\begin{align*}
\pi_1 &= \frac{\exp(\mathbf{w}_1^T \mathbf{x})}{1 + \exp(\mathbf{w}_1^T \mathbf{x}) + \exp(\mathbf{w}_2^T \mathbf{x})} \\
\pi_2 &= \frac{\exp(\mathbf{w}_2^T \mathbf{x})}{1 + \exp(\mathbf{w}_1^T \mathbf{x}) + \exp(\mathbf{w}_2^T \mathbf{x})} \\
\pi_3 &= \frac{1}{1 + \exp(\mathbf{w}_1^T \mathbf{x}) + \exp(\mathbf{w}_2^T \mathbf{x})}
\end{align*}
\]

The surfaces indicate regions where one class dominates over the others. It is important to note that the points for \( c_1 \) and \( c_2 \) are shown displaced to emphasize the contrast with \( c_3 \), which is the reference class.

Overall, the training set accuracy for the multi-class logistic classifier is 96.7%, since it misclassifies only five points (shown in dark gray). For example, for the point \( \mathbf{x} = (-0.52, -1.19, 1)^T \), we have:

\[
\begin{align*}
\pi_1 &= 0.0001 \\
\pi_2 &= 0.447 \\
\pi_3 &= 0.553
\end{align*}
\]

Thus, the predicted class is \( \hat{y} = \arg\max_{c_j} \{ \pi_j \} = c_3 \), whereas the true class is \( y = c_2 \).
1.4 EXERCISES

Q1. Show that $\frac{\partial}{\partial z} \theta(z) = \theta(z) \cdot \theta(-z)$, where $\theta(\cdot)$ is the logistic function.

Q2. Show that the logit function is the inverse of the logistic function.

Q3. Given the softmax function:

$$\pi_j = \frac{\exp\{w_j^T X\}}{\sum_{i=1}^K \exp\{w_i^T X\}}$$

Show that

$$\frac{\partial}{\partial w_a} \pi_j = \begin{cases} 
\pi_a \cdot (1 - \pi_a) \cdot x_i & \text{if } j = a \\
-\pi_a \cdot \pi_j \cdot x_i & \text{if } j \neq a
\end{cases}$$