Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

[1]Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

[2]Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 24: Logistic Regression

Binary Logistic Regression

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.

Since there are only two outcomes for the response variable Y, its probability mass function for $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$ is given as:

$$P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) = \pi(\tilde{\boldsymbol{x}}) \qquad P(Y = 0 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) = 1 - \pi(\tilde{\boldsymbol{x}})$$

where $\pi(\tilde{\boldsymbol{x}})$ is the unknown true parameter value, denoting the probability of $Y = 1$ given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$.

Binary Logistic Regression

Instead of directly predicting the response value, the goal is to learn the probability, $P(Y = 1|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})$, which is also the expected value of Y given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$.

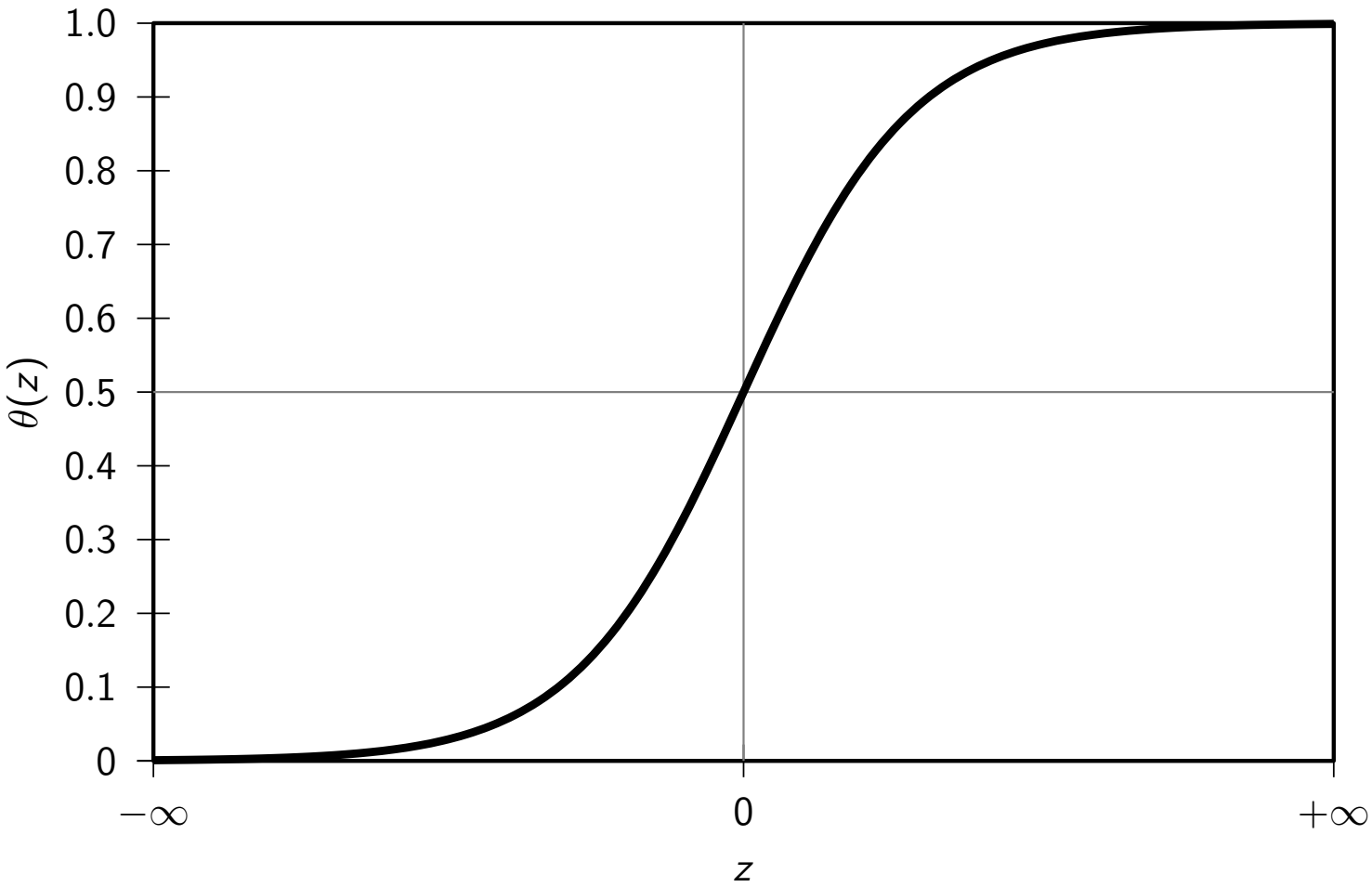
Since $P(Y = 1|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})$ is a probability, it is not appropriate to directly use the linear regression model.

The reason we cannot simply use $P(Y = 1|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) = f(\tilde{\boldsymbol{x}})$ is due to the fact that $f(\tilde{\boldsymbol{x}})$ can be arbitrarily large or arbitrarily small, whereas for logistic regression, we require that the output represents a probability value.

The name "logistic regression" comes from the *logistic* function (also called the *sigmoid* function) that "squashes" the output to be between 0 and 1 for any scalar input. z

$$\theta(z) = \frac{1}{1 + \exp\{-z\}} = \frac{\exp\{z\}}{1 + \exp\{z\}} \tag{1}$$

Logistic Function



Logistic Function

Example

Figure shows the plot for the logistic function for z ranging from $-\infty$ to $+\infty$. In particular consider what happens when z is $-\infty$, $+\infty$ and 0; we have

$$\theta(-\infty) = \frac{1}{1+\exp\{\infty\}} = \frac{1}{\infty} = 0$$

$$\theta(+\infty) = \frac{1}{1+\exp\{-\infty\}} = \frac{1}{1} = 1$$

$$\theta(0) = \frac{1}{1+\exp\{0\}} = \frac{1}{2} = 0.5$$

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

$$1 - \theta(z) = 1 - \frac{\exp\{z\}}{1+\exp\{z\}} = \frac{1+\exp\{z\}-\exp\{z\}}{1+\exp\{z\}} = \frac{1}{1+\exp\{z\}} = \theta(-z) \quad (2)$$

Binary Logistic Regression

Using the logistic function, we define the logistic regression model as follows:

$$P(Y=1|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\pi(\tilde{\boldsymbol{x}})=\theta(f(\tilde{\boldsymbol{x}}))=\theta(\tilde{\omega}^T\tilde{\boldsymbol{x}})=\frac{\exp\{\tilde{\omega}^T\tilde{\boldsymbol{x}}\}}{1+\exp\{\tilde{\omega}^T\tilde{\boldsymbol{x}}\}} \tag{3}$$

Thus, the probability that the response is $Y=1$ is the output of the logistic function for the input $\tilde{\omega}^T\tilde{\boldsymbol{x}}$. On the other hand, the probability for $Y=0$ is given as

$$P(Y=0|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=1-P(Y=1|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\theta(-\tilde{\omega}^T\tilde{\boldsymbol{x}})=\frac{1}{1+\exp\{\tilde{\omega}^T\tilde{\boldsymbol{x}}\}}$$

that is, $1-\theta(z)=\theta(-z)$ for $z=\tilde{\omega}^T\tilde{\boldsymbol{x}}$.
Combining these two cases the full logistic regression model is given as

$$P(Y|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\theta(\tilde{\omega}^T\tilde{\boldsymbol{x}})^Y\cdot\theta(-\tilde{\omega}^T\tilde{\boldsymbol{x}})^{1-Y} \tag{4}$$

since Y is a Bernoulli random variable that takes on either the value 1 or 0. We can observe that $P(Y|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\theta(\tilde{\omega}^T\tilde{\boldsymbol{x}})$ when $Y=1$ and $P(Y|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\theta(-\tilde{\omega}^T\tilde{\boldsymbol{x}})$ when $Y=0$, as desired.

Log-Odds Ratio

Define the *odds ratio* for the occurrence of $Y = 1$ as follows:

$$\begin{aligned}\text{odds}(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) &= \frac{P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}{P(Y = 0 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})} = \frac{\theta(\tilde{\omega}^T \tilde{\boldsymbol{x}})}{\theta(-\tilde{\omega}^T \tilde{\boldsymbol{x}})} \\ &= \frac{\exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}}{1 + \exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}} \cdot \left(1 + \exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}\right) \\ &= \boxed{\exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}} \end{aligned} \tag{5}$$

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

$$\begin{aligned}\ln\left(\text{odds}(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})\right) &= \ln\left(\frac{P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}{1 - P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}\right) = \ln\left(\exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}\right) = \tilde{\omega}^T \tilde{\boldsymbol{x}} \\ &= \omega_0 \cdot x_0 + \omega_1 \cdot x_1 + \cdots + \omega_d \cdot x_d \end{aligned} \tag{6}$$

The log-odds ratio function is also called the *logit* function, defined as

$$\text{logit}(z) = \ln\left(\frac{z}{1 - z}\right)$$

It is the inverse of the logistic function.

Log-Odds Ratio

We can see that

$$\ln\big(\text{odds}(Y=1|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})\big) = \text{logit}\big(P(Y=1|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})\big)$$

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

$$\ln(\text{odds}(Y=1|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})) = \omega_i \cdot x_i + C$$

$$\implies \text{odds}(Y=1|\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}}) = \exp\{\omega_i \cdot x_i + C\} = \exp\{\omega_i \cdot x_i\} \cdot \exp\{C\} \propto \exp\{\omega_i \cdot x_i\}$$

where C is a constant comprising the fixed attributes. The regression coefficient ω_i can therefore 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 $Y=1$ increases exponentially per unit change in X_i.

Maximum Likelihood Estimation

We will use the maximum likelihood approach to learn the weight vector $\tilde{\boldsymbol{w}}$. *Likelihood* is defined as the probability of the observed data given the estimated parameters $\tilde{\boldsymbol{w}}$.

$$L(\tilde{\boldsymbol{w}}) = P(Y|\tilde{\boldsymbol{w}}) = \prod_{i=1}^{n} P(y_i|\,\tilde{\boldsymbol{x}}_i) = \prod_{i=1}^{n} \theta(\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}_i)^{y_i} \cdot \theta(-\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}_i)^{1-y_i}$$

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:

$$\ln\left(L(\tilde{\boldsymbol{w}})\right) = \sum_{i=1}^{n} y_i \cdot \ln\left(\theta(\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}_i)\right) + (1-y_i) \cdot \ln\left(\theta(-\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}_i)\right) \tag{7}$$

Maximum Likelihood Estimation

The negative of the log-likelihood can also be considered as an error function, the *cross-entropy error function*, given as follows:

$$E(\tilde{\boldsymbol{w}}) = -\ln(L(\tilde{\boldsymbol{w}})) = \sum_{i=1}^{n} y_i \cdot \ln\left(\frac{1}{\theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i)}\right) + (1 - y_i) \cdot \ln\left(\frac{1}{1 - \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i)}\right) \tag{8}$$

The task of maximizing the log-likelihood is therefore equivalent to minimizing the cross-entropy error.

Maximum Likelihood Estimation

Typically, to obtain the optimal weight vector $\tilde{\boldsymbol{w}}$, we would differentiate the log-likelihood function with respect to $\tilde{\boldsymbol{w}}$, set the result to 0, and then solve for $\tilde{\boldsymbol{w}}$. However, for the log-likelihood formulation there is no closed form solution to compute the weight vector $\tilde{\boldsymbol{w}}$. Instead, we use an iterative *gradient ascent* method to compute the optimal value.

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

$$\nabla(\tilde{\boldsymbol{w}}) = \frac{\partial}{\partial \tilde{\boldsymbol{w}}}\Big\{\ln\big(L(\tilde{\boldsymbol{w}})\big)\Big\} = \frac{\partial}{\partial \tilde{\boldsymbol{w}}}\Big\{\sum_{i=1}^{n} y_i \cdot \ln\big(\theta(z_i)\big) + (1-y_i)\cdot \ln\big(\theta(-z_i)\big)\Big\} \tag{9}$$

where $z_i = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i$.

Maximum Likelihood Estimation

The gradient ascent method starts at some initial estimate for $\tilde{\boldsymbol{w}}$, denoted $\tilde{\boldsymbol{w}}^0$. At each step t, the method moves in the direction of steepest ascent, which is given by the gradient vector. Thus, given the current estimate $\tilde{\boldsymbol{w}}^t$, we can obtain the next estimate as follows:

$$\tilde{\boldsymbol{w}}^{t+1} = \tilde{\boldsymbol{w}}^t + \eta \cdot \nabla(\tilde{\boldsymbol{w}}^t) \tag{10}$$

Here, $\eta > 0$ 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 $\tilde{\boldsymbol{w}}$, the gradient will be zero, i.e., $\nabla(\tilde{\boldsymbol{w}}) = 0$, as desired.

Stochastic Gradient Ascent

The gradient ascent method computes the gradient by considering all the data points, and it is therefore called *batch* gradient ascent. 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 *stochastic gradient ascent* (SGA) for computing the optimal weight vector $\tilde{\boldsymbol{w}}$. Given a randomly chosen point $\tilde{\boldsymbol{x}}_i$, the point-specific gradient is given as

$$\nabla(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}}_i) = \left(y_i - \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i)\right) \cdot \tilde{\boldsymbol{x}}_i \tag{11}$$

Unlike batch gradient ascent that updates $\tilde{\boldsymbol{w}}$ by considering all the points, in stochastic gradient ascent the weight vector is updated after observing each point, and the updated values are used immediately in the next update. Computing the full gradient in the batch approach can be very expensive. In contrast, computing the partial gradient at each point is very fast, and due to the stochastic updates to $\tilde{\boldsymbol{w}}$, typically SGA is much faster than the batch approach for very large datasets.

Binary Logistic Regression

Once the model has been trained, we can predict the response for any new augmented test point $\tilde{\boldsymbol{z}}$ as follows:

$$\hat{y} = \begin{cases} 1 & \text{if } \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{z}}) \geq 0.5 \\ 0 & \text{if } \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{z}}) < 0.5 \end{cases} \tag{12}$$

Logistic Regression: Stochastic Gradient Ascent

LogisticRegression-SGA $(\boldsymbol{D}, \eta, \epsilon)$:

foreach $\boldsymbol{x}_i \in \boldsymbol{D}$ **do** $\tilde{\boldsymbol{x}}_i^T \leftarrow \begin{pmatrix} 1 & \boldsymbol{x}_i^T \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$

$t \leftarrow 0$ // step/iteration counter

$\tilde{\boldsymbol{w}}^0 \leftarrow (0, \ldots, 0)^T \in \mathbb{R}^{d+1}$ // initial weight vector

repeat

  $\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}}^t$ // make a copy of $\tilde{\boldsymbol{w}}^t$

  foreach $\tilde{\boldsymbol{x}}_i \in \tilde{\boldsymbol{D}}$ *in random order* **do**

    $\nabla(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}}_i) \leftarrow \left(y_i - \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i)\right) \cdot \tilde{\boldsymbol{x}}_i$ // compute gradient at $\tilde{\boldsymbol{x}}_i$

    $\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} + \eta \cdot \nabla(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}}_i)$ // update estimate for $\tilde{\boldsymbol{w}}$

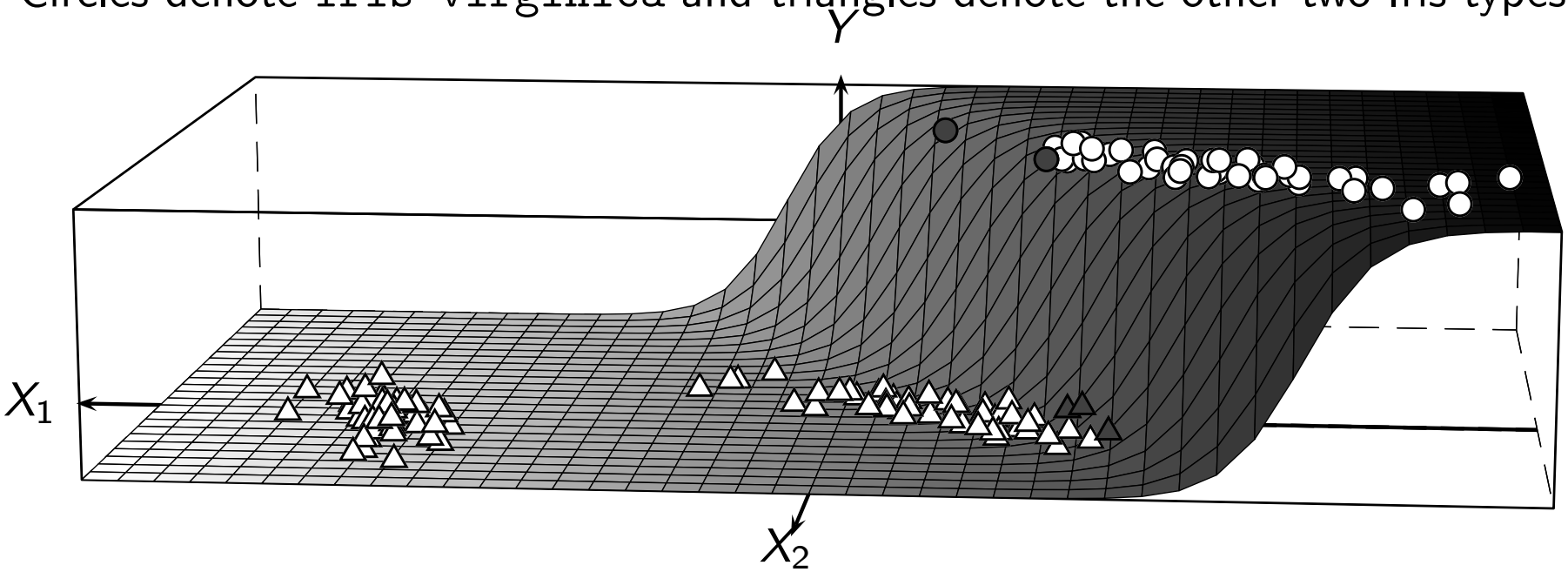
  $\tilde{\boldsymbol{w}}^{t+1} \leftarrow \tilde{\boldsymbol{w}}$ // update $\tilde{\boldsymbol{w}}^{t+1}$

  $t \leftarrow t + 1$

until $\left\| \tilde{\boldsymbol{w}}^t - \tilde{\boldsymbol{w}}^{t-1} \right\| \leq \epsilon$

Logistic Regression

Iris principal components data. Misclassified point are shown in dark gray color. Circles denote `Iris-virginica` and triangles denote the other two Iris types.



Linear Regression

Iris principal components data. Misclassified point are shown in dark gray color. Circles denote `Iris-virginica` and triangles denote the other two Iris types.

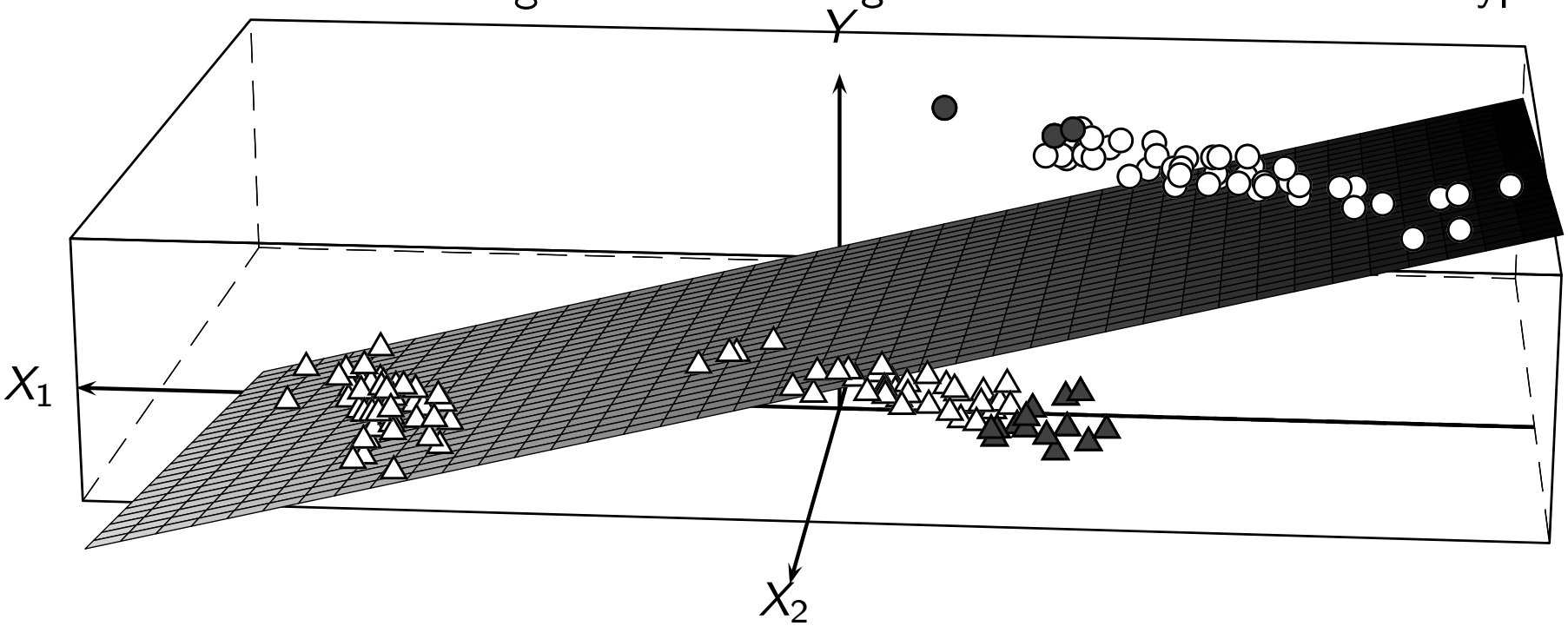


Figure shows the output of logistic regression modeling on the Iris principal components data, where the independent attributes X_1 and X_2 represent the first two principal components, and the binary response variable Y represents the type of Iris flower; $Y = 1$ corresponds to `Iris-virginica`, whereas $Y = 0$ corresponds to the two other Iris types, namely `Iris-setosa` and `Iris-versicolor`.

The fitted logistic model is given as

$$\tilde{\boldsymbol{w}} = (w_0, w_1, w_2)^T = (-6.79, -5.07, -3.29)^T$$

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

Figure plots $P(Y = 1|\tilde{\boldsymbol{x}})$ for various values of $\tilde{\boldsymbol{x}}$. Given $\tilde{\boldsymbol{x}}$, if $P(Y = 1|\tilde{\boldsymbol{x}}) \geq 0.5$, then we predict $\hat{y} = 1$, otherwise we predict $\hat{y} = 0$.

Figure shows that five points (shown in dark gray) are misclassified. For example, for $\tilde{\boldsymbol{x}} = (1, -0.52, -1.19)^T$ we have:

$$P(Y = 1|\tilde{\boldsymbol{x}}) = \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}) = \theta(-0.24) = 0.44$$
$$P(Y = 0|\tilde{\boldsymbol{x}}) = 1 - P(Y = 1|\tilde{\boldsymbol{x}}) = 0.54$$

Thus, the predicted response for $\tilde{\boldsymbol{x}}$ is $\hat{y} = 0$, whereas the true class is $y = 1$. The plane of best fit in linear regression is shown in the figure, with the weight vector:

$$\tilde{\boldsymbol{w}} = (0.333, -0.167, 0.074)^T$$
$$\hat{y} = f(\tilde{\boldsymbol{x}}) = 0.333 - 0.167 \cdot x_1 + 0.074 \cdot x_2$$

Since the response vector Y is binary, we predict the response class as $y = 1$ if $f(\tilde{\boldsymbol{x}}) \geq 0.5$, and $y = 0$ otherwise. The linear regression model results in 17 points being misclassified (dark gray points).
Since there are $n = 150$ points in total, this results in a training set or in-sample accuracy of 88.7% for linear regression, while logistic regression misclassifies only 5 points, an accuracy of 96.7%, which is a much better fit.

Multiclass 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., $Y \in \{c_1, c_2, \cdots, c_K\}$. We model Y as a K-dimensional multivariate Bernoulli random variable. 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

$$\boldsymbol{e}_i = (\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{K-i})^T$$

whose ith element $e_{ii} = 1$, and all other elements $e_{ij} = 0$, so that $\sum_{j=1}^{K} e_{ij} = 1$. The probability mass function for $\boldsymbol{Y}$ given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$ is

$$P(\boldsymbol{Y} = \boldsymbol{e}_i | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) = \pi_i(\tilde{\boldsymbol{x}}), \text{ for } i = 1, 2, \ldots, K$$

where $\pi_i(\tilde{\boldsymbol{x}})$ is the (unknown) probability of observing class c_i given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$. Thus, there are K unknown parameters, which must satisfy the following constraint:

$$\sum_{i=1}^{K} \pi_i(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{K} P(\boldsymbol{Y} = \boldsymbol{e}_i | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) = 1$$

Multiclass Logistic Regression

Given that only one element of $\boldsymbol{Y}$ is 1, the probability mass function of $\boldsymbol{Y}$ can be written compactly as

$$P(\boldsymbol{Y}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}) = \prod_{j=1}^{K} (\pi_j(\tilde{\boldsymbol{x}}))^{Y_j} \tag{13}$$

Note that if $\boldsymbol{Y} = \boldsymbol{e}_i$, only $Y_i = 1$ and the rest of the elements $Y_j = 0$ for $j \neq i$. In multiclass logistic regression, we select one of the values, say c_K, as a reference or base class, and consider the log-odds ratio of the other classes with respect to c_K; we assume that each of these log-odd ratios are linear in $\tilde{\boldsymbol{X}}$, but with a different augmented weight vector $\tilde{\boldsymbol{\omega}}_i$, for class c_i. That is, the log-odds ratio of class c_i with respect to class c_K is assumed to satisfy

$$\begin{aligned} \ln(\text{odds}(\boldsymbol{Y} = \boldsymbol{e}_i|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})) &= \ln\left(\frac{P(\boldsymbol{Y} = \boldsymbol{e}_i|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}{P(\boldsymbol{Y} = \boldsymbol{e}_K|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}\right) = \ln\left(\frac{\pi_i(\tilde{\boldsymbol{x}})}{\pi_K(\tilde{\boldsymbol{x}})}\right) = \tilde{\boldsymbol{\omega}}_i^T \tilde{\boldsymbol{x}} \\ &= \omega_{i0} \cdot x_0 + \omega_{i1} \cdot x_1 + \cdots + \omega_{id} \cdot x_d \end{aligned}$$

where $\omega_{i0} = \beta_i$ is the true bias value for class c_i.

Multiclass Logistic Regression

Setting $\tilde{\omega}_K = 0$, we have $\exp\{\tilde{\omega}_K^T \tilde{x}\} = 1$, and thus we can write the full model for multiclass logistic regression as follows:

$$\pi_i(\tilde{x}) = \frac{\exp\{\tilde{\omega}_i^T \tilde{x}\}}{\sum_{j=1}^{K} \exp\{\tilde{\omega}_j^T \tilde{x}\}}, \qquad \text{for all } i = 1, 2, \cdots, K \tag{14}$$

This function is also called the *softmax* function. When $K = 2$, this formulation yields exactly the same model as in binary logistic regression.

Maximum Likelihood Estimation

To find the K sets of regression weight vectors $\tilde{\boldsymbol{w}}_i$, for $i = 1, 2, \cdots, K$, we use the gradient ascent approach to maximize the log-likelihood function. The likelihood of the data is given as

$$L(\tilde{\boldsymbol{W}}) = P(\boldsymbol{Y}|\tilde{\boldsymbol{W}}) = \prod_{i=1}^{n} P(\boldsymbol{y}_i|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}_i) = \prod_{i=1}^{n}\prod_{j=1}^{K} (\pi_j(\tilde{\boldsymbol{x}}_i))^{y_{ij}}$$

where $\tilde{\boldsymbol{W}} = \{\tilde{\boldsymbol{w}}_1, \tilde{\boldsymbol{w}}_2, \cdots, \tilde{\boldsymbol{w}}_K\}$ is the set of K weight vectors. The log-likelihood is then given as:

$$\ln\Big(L(\tilde{\boldsymbol{W}})\Big) = \sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij} \cdot \ln(\pi_j(\tilde{\boldsymbol{x}}_i)) = \sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij} \cdot \ln\left(\frac{\exp\{\tilde{\boldsymbol{w}}_j^T \tilde{\boldsymbol{x}}_i\}}{\sum_{a=1}^{K}\exp\{\tilde{\boldsymbol{w}}_a^T \tilde{\boldsymbol{x}}_i\}}\right) \tag{15}$$

Note that the negative of the log-likelihood function can be regarded as an error function, commonly known as *cross-entropy error*. For stochastic gradient ascent, we update the weight vectors by considering only one point at a time.

Maximum Likelihood Estimation

The gradient of the log-likelihood function with respect to $\tilde{\boldsymbol{w}}_j$ at a given point $\tilde{\boldsymbol{x}}_i$ is given as

$$\nabla(\tilde{\boldsymbol{w}}_j, \tilde{\boldsymbol{x}}_i) = \big(y_{ij} - \pi_j(\tilde{\boldsymbol{x}}_i)\big) \cdot \tilde{\boldsymbol{x}}_i \tag{16}$$

which results in the following update rule for the jth weight vector:

$$\tilde{\boldsymbol{w}}_j^{t+1} = \tilde{\boldsymbol{w}}_j^{t} + \eta \cdot \nabla(\tilde{\boldsymbol{w}}_j^{t}, \tilde{\boldsymbol{x}}_i) \tag{17}$$

where $\tilde{\boldsymbol{w}}_j^{t}$ denotes the estimate of $\tilde{\boldsymbol{w}}_j$ at step t, and η is the learning rate. Once the model has been trained, we can predict the class for any new augmented test point $\tilde{\boldsymbol{z}}$ as follows:

$$\hat{y} = \arg\max_{c_i}\big\{\pi_i(\tilde{\boldsymbol{z}})\big\} = \arg\max_{c_i}\left\{\frac{\exp\{\tilde{\boldsymbol{w}}_i^T \tilde{\boldsymbol{z}}\}}{\sum_{j=1}^{K}\exp\{\tilde{\boldsymbol{w}}_j^T \tilde{\boldsymbol{z}}\}}\right\} \tag{18}$$

That is, we evalute the softmax function, and then predict the class of $\tilde{\boldsymbol{z}}$ as the one with the highest probability.

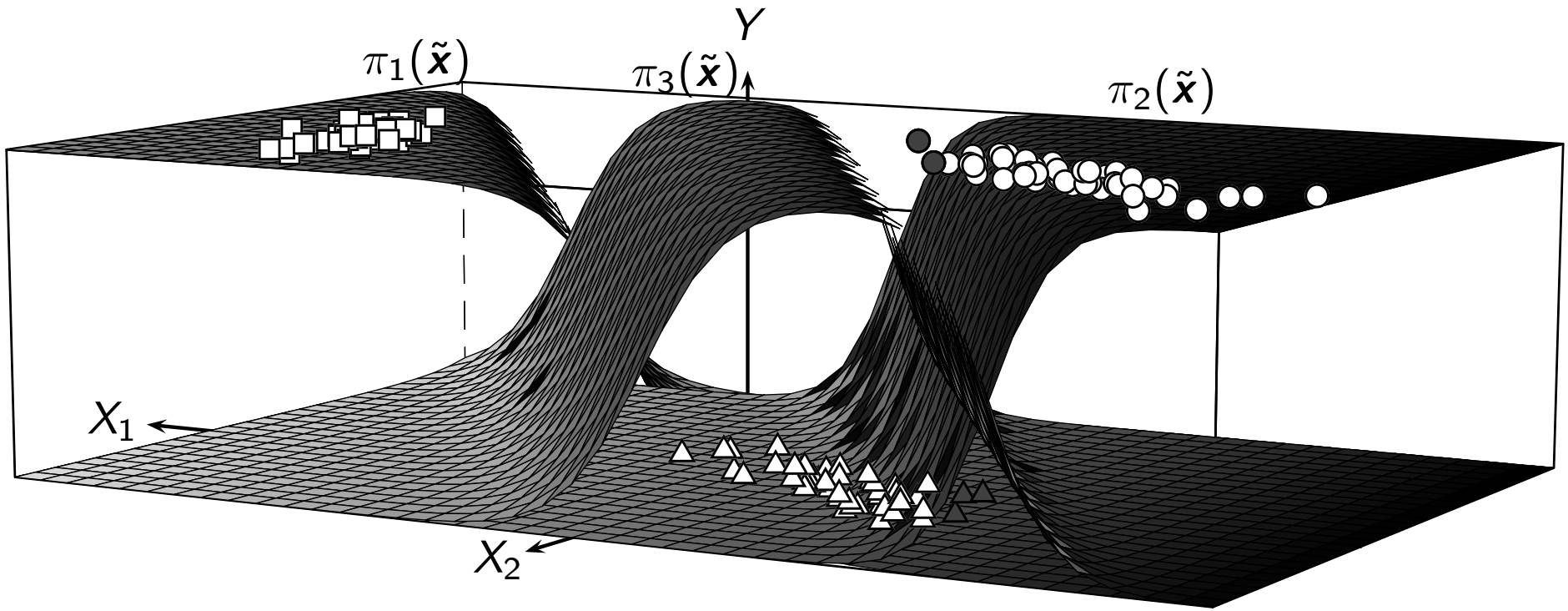
Multiclass Logistic Regression Algorithm

LogisticRegression-MultiClass $(\boldsymbol{D}, \eta, \epsilon)$**:**

foreach $(\boldsymbol{x}_i^T, y_i) \in \boldsymbol{D}$ **do**
  $\tilde{\boldsymbol{x}}_i^T \leftarrow \begin{pmatrix} 1 & \boldsymbol{x}_i^T \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$
  $\boldsymbol{y}_i \leftarrow \boldsymbol{e}_j$ if $y_i = c_j$ // map y_i to K-dim Bernoulli vector
$t \leftarrow 0$ // step/iteration counter
foreach $j = 1, 2, \cdots, K$ **do** $\tilde{\boldsymbol{w}}_j^t \leftarrow (0, \ldots, 0)^T \in \mathbb{R}^{d+1}$
repeat
  foreach $j = 1, 2, \cdots, K-1$ **do** $\tilde{\boldsymbol{w}}_j \leftarrow \tilde{\boldsymbol{w}}_j^t$ // make a copy of $\tilde{\boldsymbol{w}}_j^t$
  foreach $\tilde{\boldsymbol{x}}_i \in \tilde{\boldsymbol{D}}$ *in random order* **do**
    foreach $j = 1, 2, \cdots, K-1$ **do**
      $\pi_j(\tilde{\boldsymbol{x}}_i) \leftarrow \exp\left\{\tilde{\boldsymbol{w}}_j^T \tilde{\boldsymbol{x}}_i\right\} / \sum_{a=1}^{K} \exp\left\{\tilde{\boldsymbol{w}}_a^T \tilde{\boldsymbol{x}}_i\right\}$
      $\nabla(\tilde{\boldsymbol{w}}_j, \tilde{\boldsymbol{x}}_i) \leftarrow \left(y_{ij} - \pi_j(\tilde{\boldsymbol{x}}_i)\right) \cdot \tilde{\boldsymbol{x}}_i$ // gradient at $\tilde{\boldsymbol{w}}_j$
      $\tilde{\boldsymbol{w}}_j \leftarrow \tilde{\boldsymbol{w}}_j + \eta \cdot \nabla(\tilde{\boldsymbol{w}}_j, \tilde{\boldsymbol{x}}_i)$ // update estimate for $\tilde{\boldsymbol{w}}_j$
  foreach $j = 1, 2, \cdots, K-1$ **do** $\tilde{\boldsymbol{w}}_j^{t+1} \leftarrow \tilde{\boldsymbol{w}}_j$ // update $\tilde{\boldsymbol{w}}_j^{t+1}$
  $t \leftarrow t + 1$
until $\sum_{j=1}^{K-1} \left\| \tilde{\boldsymbol{w}}_j^t - \tilde{\boldsymbol{w}}_j^{t-1} \right\| \leq \epsilon$

Multiclass logistic regression

Iris principal components data. Misclassified point are shown in dark gray color. All the points actually lie in the (X_1, X_2) plane, but c_1 and c_2 are shown displaced along Y with respect to the base class c_3 purely for illustration purposes.



Multiclass logistic regression

Example

Consider the Iris dataset, with $n = 150$ points in a 2D space spanned by the first two principal components, as shown in the figure. 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 $\boldsymbol{e}_1 = (1,0,0)^T$, $Y = c_2$ to $\boldsymbol{e}_2 = (0,1,0)^T$ and $Y = c_3$ to $\boldsymbol{e}_3 = (0,0,1)^T$.

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

$$\tilde{\boldsymbol{w}}_1 = (-3.52, 3.62, 2.61)^T$$
$$\tilde{\boldsymbol{w}}_2 = (-6.95, -5.18, -3.40)^T$$
$$\tilde{\boldsymbol{w}}_3 = (0,0,0)^T$$

Multiclass logistic regression

Example

Figure plots the decision surfaces corresponding to the softmax functions:

$$\pi_1(\tilde{\boldsymbol{x}}) = \frac{\exp\{\tilde{\boldsymbol{w}}_1^T \tilde{\boldsymbol{x}}\}}{1 + \exp\{\tilde{\boldsymbol{w}}_1^T \tilde{\boldsymbol{x}}\} + \exp\{\tilde{\boldsymbol{w}}_2^T \tilde{\boldsymbol{x}}\}}$$

$$\pi_2(\tilde{\boldsymbol{x}}) = \frac{\exp\{\tilde{\boldsymbol{w}}_2^T \tilde{\boldsymbol{x}}\}}{1 + \exp\{\tilde{\boldsymbol{w}}_1^T \tilde{\boldsymbol{x}}\} + \exp\{\tilde{\boldsymbol{w}}_2^T \tilde{\boldsymbol{x}}\}}$$

$$\pi_3(\tilde{\boldsymbol{x}}) = \frac{1}{1 + \exp\{\tilde{\boldsymbol{w}}_1^T \tilde{\boldsymbol{x}}\} + \exp\{\tilde{\boldsymbol{w}}_2^T \tilde{\boldsymbol{x}}\}}$$

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 along Y to emphasize the contrast with c_3, which is the reference class.
Overall, the training set accuracy for the multiclass logistic classifier is 96.7%, since it misclassifies only five points (shown in dark gray). For example, for the point $\tilde{\boldsymbol{x}} = (1, -0.52, -1.19)^T$, we have:

$$\pi_1(\tilde{\boldsymbol{x}}) = 0 \qquad \pi_2(\tilde{\boldsymbol{x}}) = 0.448 \qquad \pi_3(\tilde{\boldsymbol{x}}) = 0.552$$

Thus, the predicted class is $\hat{y} = \arg\max_{c_i}\{\pi_i(\tilde{\boldsymbol{x}})\} = c_3$, whereas the true class is $y = c_2$.

Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

[1]Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

[2]Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 24: Logistic Regression