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 23: Linear Regression

Regression

Given $X_1, X_2, \cdots, X_d$ (*predictor*, *explanatory*, or *independent* variables), and given Y (*response* or *dependent* variable), *regression* aims to predict Y based on X.

That is, the goal is to learn a *regression function* f, such that

$$Y = f(X_1, X_2, \cdots, X_d) + \varepsilon = f(\boldsymbol{X}) + \varepsilon$$

where $\boldsymbol{X} = (X_1, X_2, \cdots, X_d)^T$ is the multivariate random variable comprising the predictor attributes, and ε is a random *error term* that is assumed to be independent of $\boldsymbol{X}$.

Y is comprised of two components, one dependent on X, and the other, coming from the error term, independent of the predictor attributes.

The error term encapsulates inherent uncertainty in Y, as well as, possibly the effect of unobserved, hidden or *latent* variables.

Linear Regression

In *linear regression* the function f is assumed to be linear in $\boldsymbol{X}$, that is

$$f(\boldsymbol{X}) = \beta + \omega_1 X_1 + \omega_2 X_2 + \cdots + \omega_d X_d = \beta + \sum_{i=1}^{d} \omega_i X_i = \beta + \boldsymbol{\omega}^T \boldsymbol{X}$$

β is the true (unknown) *bias* term, ω_i is the true (unknown) *regression coefficient* or *weight* for attribute X_i, and $\boldsymbol{\omega} = (\omega_1, \omega_2, \cdots, \omega_d)^T$ is the true d-dimensional weight vector.

f specifies a hyperplane in $\mathbb{R}^{d+1}$, where $\boldsymbol{\omega}$ is the the weight vector that is normal or orthogonal to the hyperplane, and β is the *intercept* or offset term.

f is completely specified by the $d+1$ parameters comprising β and ω_i, for $i = 1, \cdots, d$.

Linear regression

A common approach to predicting the bias and regression coefficients is to use the method of *least squares*.

Given the training data $\boldsymbol{D}$ with points $\boldsymbol{x}_i$ and response values y_i (for $i = 1, \cdots, n$), we seek values b and $\boldsymbol{w}$, so as to minimize the sum of squared residual errors (SSE)

$$SSE = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - b - \boldsymbol{w}^T \boldsymbol{x}_i\right)^2$$

In bivariate regression, $\boldsymbol{D}$ comprises a single predictor attribute, $X = (x_1, x_2, \cdots, x_n)^T$, along with $Y = (y_1, y_2, \cdots, y_n)^T$:

$$\hat{y}_i = f(x_i) = b + w \cdot x_i$$

Bivariate Regression

The residual error is $\epsilon_i = y_i - \hat{y}_i$ and the best line that minimizes the SSE:

$$\min_{b,w} SSE = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - b - w \cdot x_i)^2$$

We differentiate it with respect to b and set the result to 0:

$$\frac{\partial}{\partial b} SSE = -2 \sum_{i=1}^{n} (y_i - b - w \cdot x_i) = 0$$

$$\implies b = \frac{1}{n} \sum_{i=1}^{n} y_i - w \cdot \frac{1}{n} \sum_{i=1}^{n} x_i$$

Therefore, we have

$$b = \mu_Y - w \cdot \mu_X$$

Bivariate Regression

Differentiating with respect to w, we obtain

$$\frac{\partial}{\partial w} SSE = -2 \sum_{i=1}^{n} x_i (y_i - b - w \cdot x_i) = 0$$

$$\implies \sum_{i=1}^{n} x_i \cdot y_i - b \sum_{i=1}^{n} x_i - w \sum_{i=1}^{n} x_i^2 = 0$$

$$\implies \sum_{i=1}^{n} x_i \cdot y_i - \mu_Y \sum_{i=1}^{n} x_i + w \cdot \mu_X \sum_{i=1}^{n} x_i - w \sum_{i=1}^{n} x_i^2 = 0$$

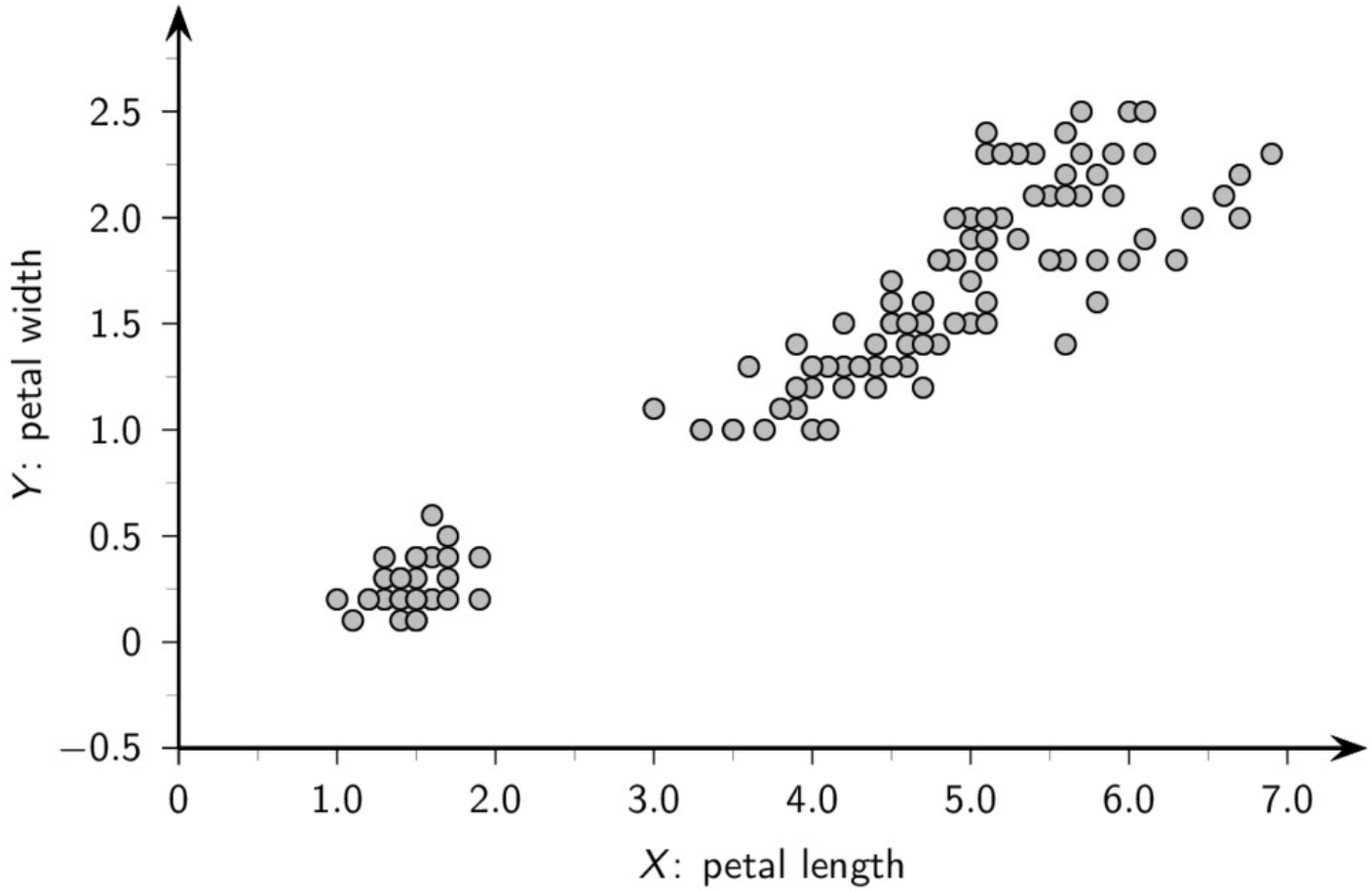
$$\implies w = \frac{\sum_{i=1}^{n} x_i \cdot y_i - n \cdot \mu_X \cdot \mu_Y}{\sum_{i=1}^{n} x_i^2 - n \cdot \mu_X^2}$$

The regression coefficient w can also be written as

$$w = \frac{\sum_{i=1}^{n} (x_i - \mu_X)(y_i - \mu_Y)}{\sum_{i=1}^{n} (x_i - \mu_X)^2} = \frac{\sigma_{XY}}{\sigma_X^2} = \frac{\text{cov}(X, Y)}{\text{var}(X)}$$

Bivariate Regression

Given two attributes `petal length` (X; the predictor variable) and `petal width` (Y; the response variable) in the Iris dataset ($n = 150$).



Bivariate Regression

Example

The mean values for these two variables are

$$\mu_X = \frac{1}{150}\sum_{i=1}^{150} x_i = \frac{563.8}{150} = 3.7587$$

$$\mu_Y = \frac{1}{150}\sum_{i=1}^{150} y_i = \frac{179.8}{150} = 1.1987$$

The variance and covariance is given as

$$\sigma_X^2 = \frac{1}{150}\sum_{i=1}^{150}(x_i - \mu_X)^2 = 3.0924$$

$$\sigma_Y^2 = \frac{1}{150}\sum_{i=1}^{150}(y_i - \mu_Y)^2 = 0.5785$$

$$\sigma_{XY} = \frac{1}{150}\sum_{i=1}^{150}(x_i - \mu_X)\cdot(y_i - \mu_Y) = 1.2877$$

Assuming a linear relationship between the response and predictor variables, we obtain the slope and intercept terms as follows

$$w = \frac{\sigma_{XY}}{\sigma_X^2} = \frac{1.2877}{3.0924} = 0.4164$$

$$b = \mu_Y - w \cdot \mu_X = 1.1987 - 0.4164 \cdot 3.7587 = -0.3665$$

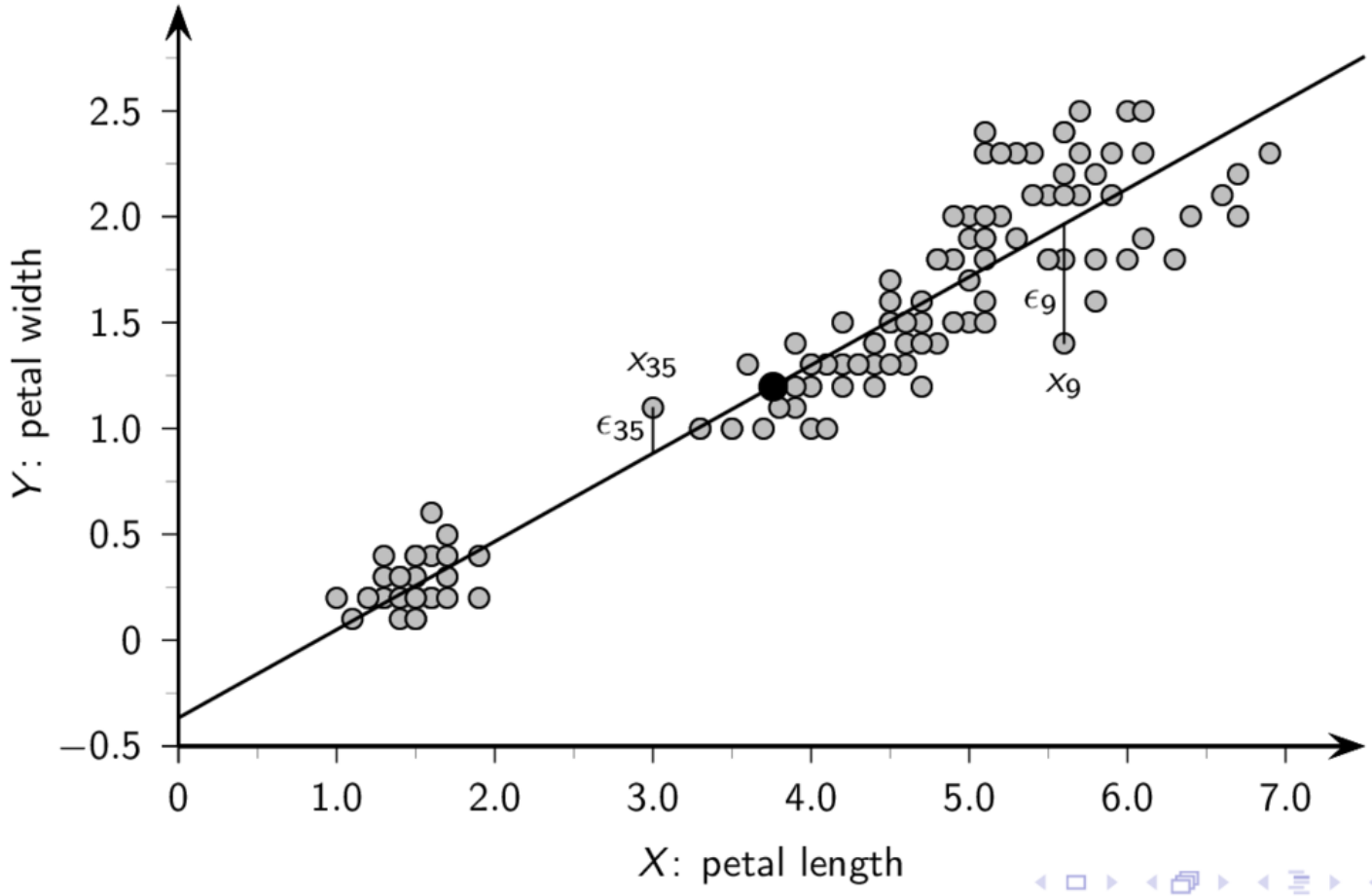
Thus, the fitted regression line is

$$\hat{y} = -0.3665 + 0.4164 \cdot x$$

Finally, we can compute the SSE value as follows:

$$SSE = \sum_{i=1}^{150} \epsilon_i^2 = \sum_{i=1}^{150} (y_i - \hat{y}_i)^2 = 6.343$$

`petal length` (X) versus `petal width` (Y). Solid circle (black) shows the mean point; residual error is shown for two sample points: x_9 and x_{35}.

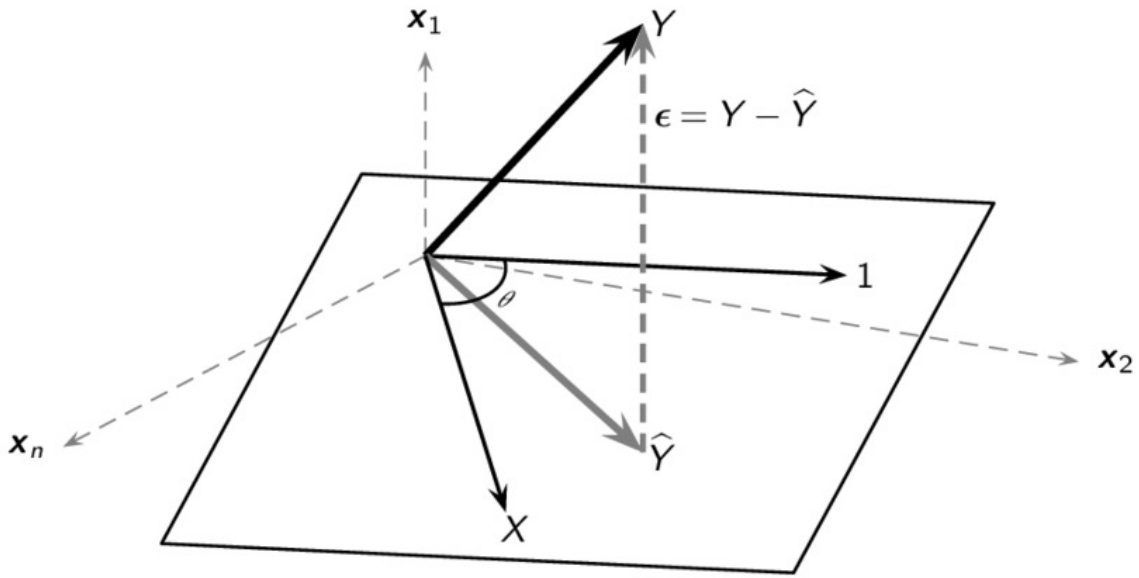


Geometry of Bivariate Regression

We can express the n equations, $y_i = b + w \cdot x_i$ for $i = 1, 2, \cdots, n$, as:

$$\widehat{Y} = b \cdot 1 + w \cdot X$$

where $1 \in \mathbb{R}^n$ is the n-dimensional vector of 1s. $\widehat{Y}$ is a linear combination of 1 and X, i.e., it must lie in the column space spanned by 1 and X, given as $\text{span}(\{1, X\})$. ϵ captures the deviation between Y and $\widehat{Y}$.



Geometry of Bivariate Regression

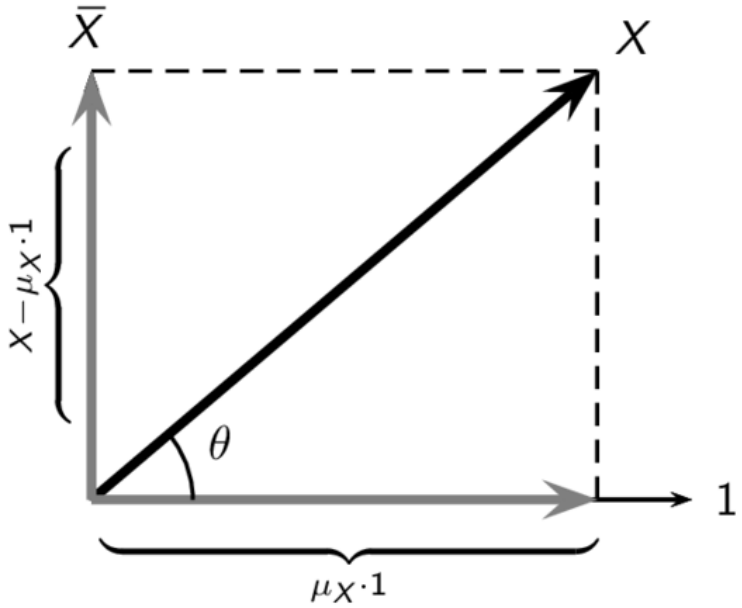
Orthogonal decomposition of X into $\overline{X}$ and $\mu_X \cdot 1$.

Even though 1 and X are linearly independent and form a basis for the column space, they need not be orthogonal.

We can create an orthogonal basis by decomposing X into a component along 1 and a component orthogonal to 1, $\overline{X}$.

$$X = \mu_X \cdot 1 + (X - \mu_X \cdot 1) = \mu_X \cdot 1 + \overline{X}$$

where $\overline{X} = X - \mu_X \cdot 1$ is the centered attribute vector.



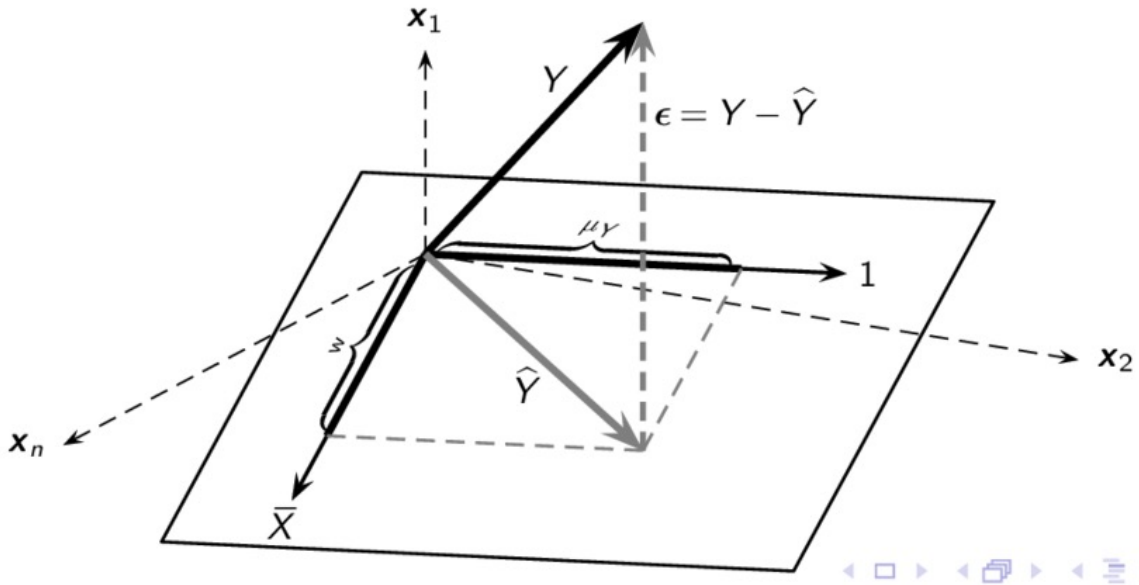
Geometry of Regression

The optimal $\widehat{Y}$ that minimizes the error is the orthogonal projection of Y onto the subspace spanned by 1 and X.

The residual error vector ϵ is thus *orthogonal* to the subspace spanned by 1 and X, and its squared length (or magnitude) equals the SSE value.

Summarizing:

$$\mu_Y = \text{proj}_1(Y) \qquad w = \text{proj}_{\overline{X}}(Y) \qquad b = \mu_Y - w \cdot \mu_X$$



Geometry of Regression

Example

Let us consider the regression of `petal length` (X) on `petal width` (Y) for the Iris dataset, with $n = 150$. First, we center X by subtracting the mean $\mu_X = 3.759$. Next, we compute the scalar projections of Y onto 1 and $\bar{X}$, to obtain

$$\mu_Y = \text{proj}_1(Y) = \left(\frac{Y^T 1}{1^T 1}\right) = \frac{179.8}{150} = 1.1987$$

$$w = \text{proj}_{\bar{X}}(Y) = \left(\frac{Y^T \bar{X}}{\bar{X}^T \bar{X}}\right) = \frac{193.16}{463.86} = 0.4164$$

Thus, the bias term b is given as

$$b = \mu_Y - w \cdot \mu_X = 1.1987 - 0.4164 \cdot 3.7587 = -0.3665$$

We can compute the SSE value as the squared length of the residual error vector

$$SSE = \|\epsilon\|^2 = \left\|Y - \hat{Y}\right\|^2 = (Y - \hat{Y})^T (Y - \hat{Y}) = 6.343$$

Multiple Regression

Multiple regression: multiple predictor attributes $X_1, X_2, \cdots, X_d$ and a single response attribute Y.

The training data sample $\boldsymbol{D} \in \mathbb{R}^{n \times d}$ comprises n points $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^T$ in a d-dimensional space, along with the corresponding observed response value y_i.

Instead of dealing with the bias b separately from the weights w_i, we can introduce a new "constant" valued attribute X_0 whose value is always fixed at 1.

The predicted response value for an augmented $(d+1)$ dimensional point $\tilde{\boldsymbol{x}}_i$ can be written as

$$\hat{y}_i = w_0 x_{i0} + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i$$

$$\hat{y}_i = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i = \sum_j \tilde{w}_j \tilde{x}_{ij} = \sum_j \tilde{D}_{ij} \tilde{w}_j$$

$$\hat{Y} = \tilde{\boldsymbol{D}} \tilde{\boldsymbol{w}}$$

Multiple Regression

The multiple regression task is to find the *best fitting hyperplane* defined by $\tilde{\boldsymbol{w}}$ that minimizes the SSE:

$$\begin{aligned}\min_{\tilde{\boldsymbol{w}}} SSE &= \sum_{i=1}^{n} \epsilon_i^2 = \|\boldsymbol{\epsilon}\|^2 = \left\|Y - \widehat{Y}\right\|^2 \\ &= (Y - \widehat{Y})^T (Y - \widehat{Y}) = Y^T Y - 2Y^T \widehat{Y} + \widehat{Y}^T \widehat{Y} \\ &= Y^T Y - 2Y^T (\tilde{\boldsymbol{D}}\tilde{\boldsymbol{w}}) + (\tilde{\boldsymbol{D}}\tilde{\boldsymbol{w}})^T (\tilde{\boldsymbol{D}}\tilde{\boldsymbol{w}}) \\ &= Y^T Y - 2\tilde{\boldsymbol{w}}^T (\tilde{\boldsymbol{D}}^T Y) + \tilde{\boldsymbol{w}}^T (\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}})\tilde{\boldsymbol{w}}\end{aligned}$$

Therefore, the optimal weight vector is given as

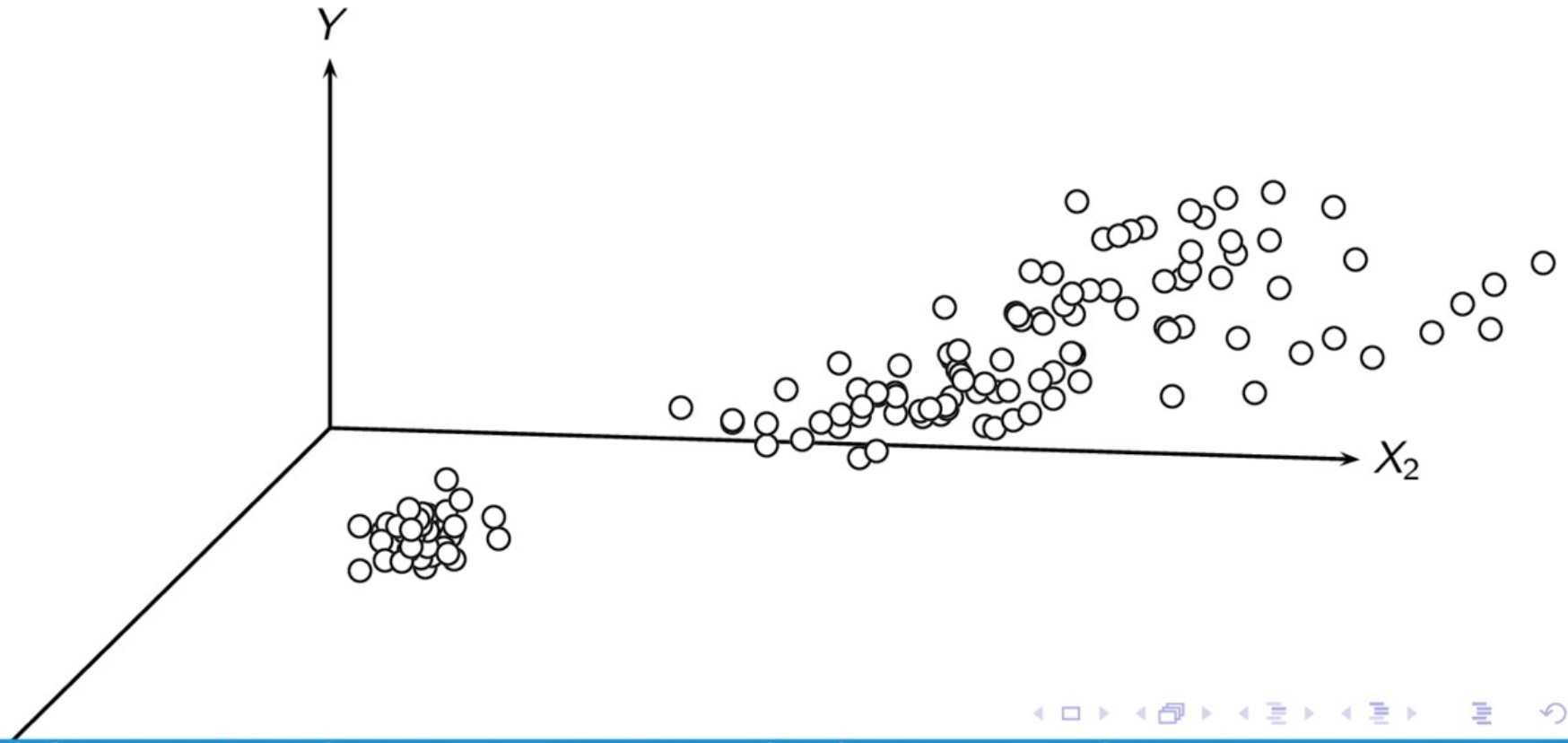
$$\tilde{\boldsymbol{w}} = (\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}})^{-1} \tilde{\boldsymbol{D}}^T Y$$

$$\widehat{Y} = \tilde{\boldsymbol{D}}\tilde{\boldsymbol{w}}$$

Multiple Regression

Example

Given `sepal length` (X_1) and `petal length` (X_2) on the response attribute `petal width` (Y) for the Iris dataset with $n = 150$ points, we want to learn the multiple regression.



Multiple Regression

Example

We and $X_0 = 1_{150}$ and $\tilde{\boldsymbol{D}} \in \mathbb{R}^{150 \times 3}$. We then compute $\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}}$ and its inverse

$$\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} = \begin{pmatrix} 150.0 & 876.50 & 563.80 \\ 876.5 & 5223.85 & 3484.25 \\ 563.8 & 3484.25 & 2583.00 \end{pmatrix} \quad (\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}})^{-1} = \begin{pmatrix} 0.793 & -0.176 & 0.064 \\ -0.176 & 0.041 & -0.017 \\ 0.064 & -0.017 & 0.009 \end{pmatrix}$$

We also compute $\tilde{\boldsymbol{D}}^T Y$, given as

$$\tilde{\boldsymbol{D}}^T Y = \begin{pmatrix} 179.80 \\ 1127.65 \\ 868.97 \end{pmatrix}$$

The augmented weight vector $\tilde{\boldsymbol{w}}$ is then given as

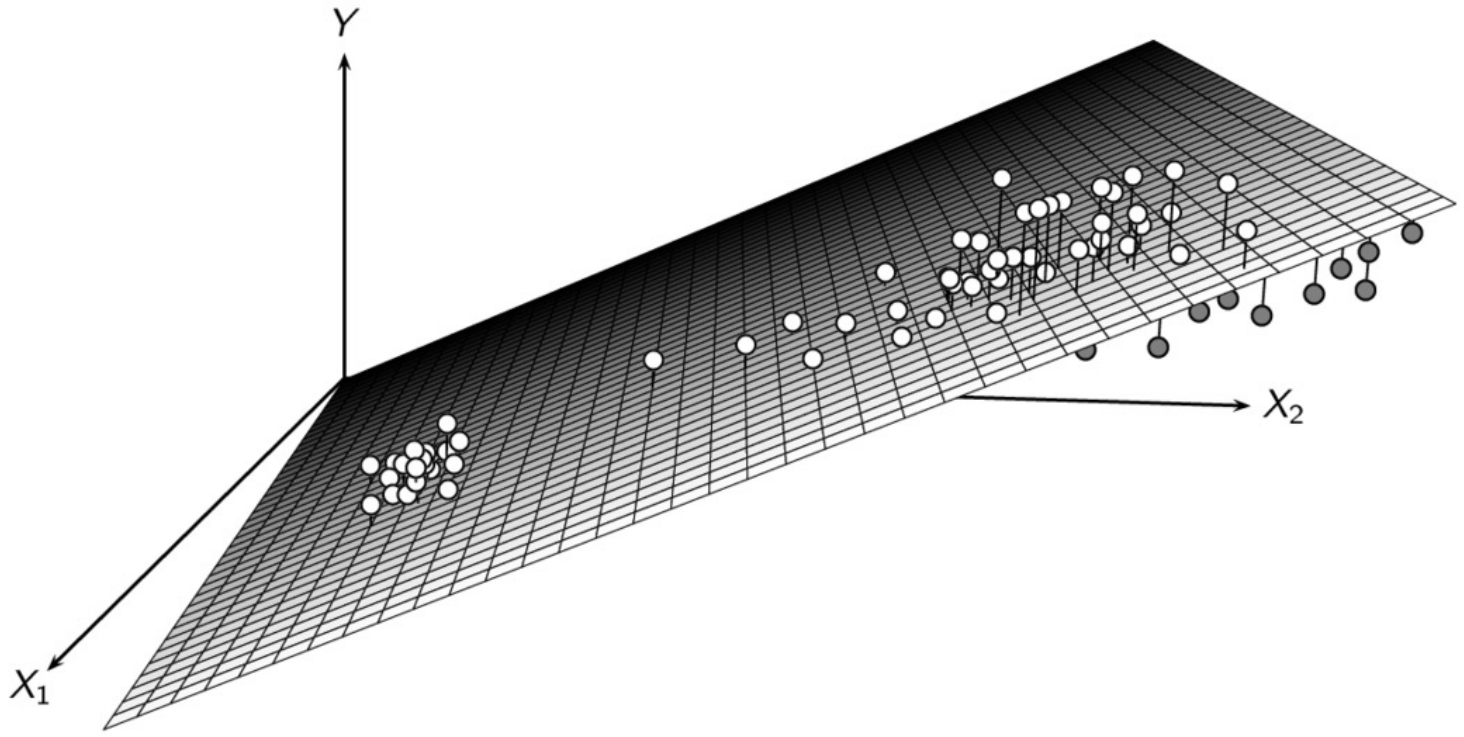
$$\tilde{\boldsymbol{w}} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = (\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}})^{-1} \cdot (\tilde{\boldsymbol{D}}^T Y) = \begin{pmatrix} -0.014 \\ -0.082 \\ 0.45 \end{pmatrix}$$

Therefore $b = w_0 = -0.014$, and $\widehat{Y} = -0.014 - 0.082 \cdot X_1 + 0.45 \cdot X_2$

Multiple Regression

Example

Figure shows the fitted hyperplane and the residual error for each point. Positive residuals (i.e., $\epsilon_i > 0$ or $\hat{y}_i > y_i$) are white, while negative residuals (i.e., $\epsilon_i < 0$ or $\hat{y}_i < y$) are gray. The SSE value for the model is 6.18.



Multiple Regression: Stochastic Gradient Descent

Instead of using the QR-factorization approach to exactly solve the multiple regression problem, we can also employ the simpler stochastic gradient algorithm. The gradient of the SSE objective is given as

$$\nabla_{\tilde{\boldsymbol{w}}} = \frac{\partial}{\partial \tilde{\boldsymbol{w}}} SSE = -\tilde{\boldsymbol{D}}^T Y + (\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}})\tilde{\boldsymbol{w}}$$

From an initial weight vector $\tilde{\boldsymbol{w}}^0$, we update $\tilde{\boldsymbol{w}}$ as:

$$\tilde{\boldsymbol{w}}^{t+1} = \tilde{\boldsymbol{w}}^t - \eta \cdot \nabla_{\tilde{\boldsymbol{w}}} = \tilde{\boldsymbol{w}}^t + \eta \cdot \tilde{\boldsymbol{D}}^T (Y - \tilde{\boldsymbol{D}} \cdot \tilde{\boldsymbol{w}}^t)$$

where $\tilde{\boldsymbol{w}}^t$ is the estimate of the weight vector at step t. We update the weight vector by considering only one (random) point at each iteration.

$$\begin{aligned} \tilde{\boldsymbol{w}}^{t+1} &= \tilde{\boldsymbol{w}}^t - \eta \cdot \nabla_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}_k) \\ &= \tilde{\boldsymbol{w}}^t + \eta \cdot (y_k - \tilde{\boldsymbol{x}}_k \cdot \tilde{\boldsymbol{w}}^t) \cdot \tilde{\boldsymbol{x}}_k \end{aligned}$$

Multiple Regression: SGD Algorithm

Multiple Regression: SGD $(\boldsymbol{D}, Y, \eta, \epsilon)$**:**

$\tilde{\boldsymbol{D}} \leftarrow \begin{pmatrix} 1 & \boldsymbol{D} \end{pmatrix}$ // augment data

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

$\tilde{\boldsymbol{w}}^t \leftarrow$ random vector in $\mathbb{R}^{d+1}$ // initial weight vector

repeat

  foreach $k = 1, 2, \cdots, n$ *(in random order)* **do**

    $\nabla_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}_k) \leftarrow -(y_k - \tilde{\boldsymbol{x}}_k^T \tilde{\boldsymbol{w}}^t) \cdot \tilde{\boldsymbol{x}}_k$ // compute gradient at $\tilde{\boldsymbol{x}}_k$

    $\tilde{\boldsymbol{w}}^{t+1} \leftarrow \tilde{\boldsymbol{w}}^t - \eta \cdot \nabla_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}_k)$ // update estimate for w_k

  $t \leftarrow t + 1$

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

Multiple Regression: SGD

Example

Multiple regression of `sepal length` (X_1) and `petal length` (X_2) on the response attribute `petal width` (Y) for the Iris dataset with $n = 150$ points.

Using the exact approach the multiple regression model was given as

$$\widehat{Y} = -0.014 \cdot X_0 - 0.082 \cdot X_1 + 0.45 \cdot X_2$$

Using SGD we obtain the following model with $\eta = 0.001$ and $\epsilon = 0.0001$:

$$\widehat{Y} = -0.031 \cdot X_0 - 0.078 \cdot X_1 + 0.45 \cdot X_2$$

The results from the SGD approach are essentially the same as the exact method, with a slight difference in the bias term.

The SSE value for the exact method is 6.179, whereas for SGD it is 6.181.

Ridge Regression

For linear regression, $\widehat{Y}$ lies in the span of the column vectors comprising the augmented data matrix $\tilde{\boldsymbol{D}}$.

Often the data is noisy and uncertain, and, therefore, instead of fitting the model to the data exactly, it may be better to fit a more robust model.

Regularization constrains the solution vector $\tilde{\boldsymbol{w}}$ to have a small norm.

Besides minimizing $\left\| Y - \widehat{Y} \right\|^2$, we add a regularization term ($\|\tilde{\boldsymbol{w}}\|^2$):

$$\min_{\tilde{\boldsymbol{w}}} \; J(\tilde{\boldsymbol{w}}) = \left\| Y - \widehat{Y} \right\|^2 + \alpha \cdot \|\tilde{\boldsymbol{w}}\|^2 = \left\| Y - \tilde{\boldsymbol{D}}\tilde{\boldsymbol{w}} \right\|^2 + \alpha \cdot \|\tilde{\boldsymbol{w}}\|^2$$

$\alpha \geq 0$ controls the tradeoff between minimizing the squared norm of the weight vector and the squared error.

Ridge Regression

We differentiate w.r.t. $\tilde{\boldsymbol{w}}$ and set the results to 0 to obtain

$$\tilde{\boldsymbol{w}} = (\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} + \alpha \cdot \boldsymbol{I})^{-1} \tilde{\boldsymbol{D}}^T Y$$

The matrix $(\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} + \alpha \cdot \boldsymbol{I})$ is always invertible (or non-singular) for $\alpha > 0$ even if $\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}}$ is not invertible (or singular).

If λ_i is an eigenvalue of $\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}}$, then $\lambda_i + \alpha$ is an eigenvalue of $(\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} + \alpha \cdot \boldsymbol{I})$. Since $\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}}$ is positive semi-definite it has non-negative eigenvalues. Even if an $\lambda_i = 0$, the corresponding eigenvalue of $(\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} + \alpha \cdot \boldsymbol{I})$ is $\lambda_i + \alpha = \alpha > 0$.

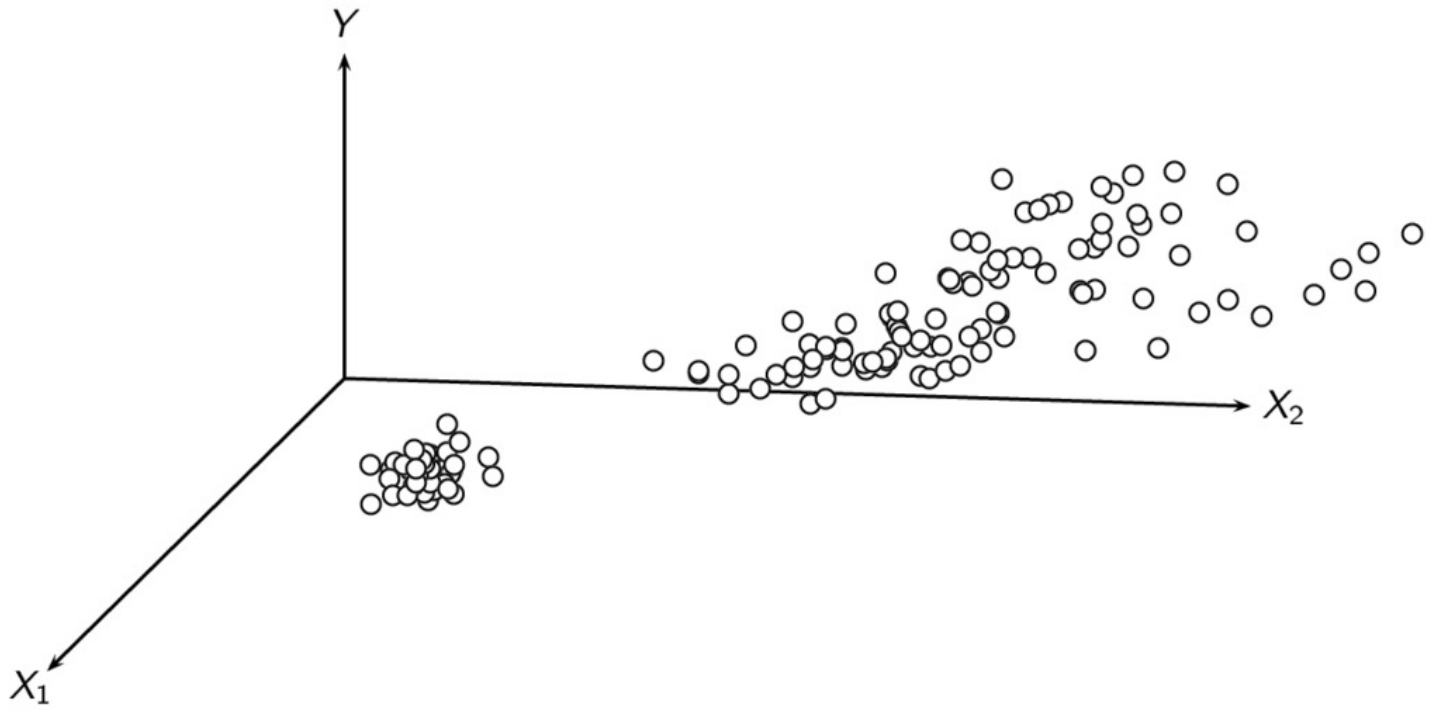
Regularized regression is called *ridge regression* because it adds a "ridge" along the main diagonal of the $\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}}$ matrix, i.e., the solution depends on $(\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} + \alpha \cdot \boldsymbol{I})$.

If we choose a small positive α we are always guaranteed a solution.

Ridge Regression

Example

Given `sepal length` (X_1) and `petal length` (X_2) on the response attribute `petal width` (Y) for the Iris dataset with $n = 150$ points, we want to learn the ridge regression.



Ridge Regression

Example

The uncentered scatter matrix is given as

$$\tilde{\boldsymbol{D}}^T \tilde{\boldsymbol{D}} = \begin{pmatrix} 150.0 & 563.8 \\ 563.8 & 2583.0 \end{pmatrix}$$

We obtain different lines of best fit for different values of the regularization constant α:

$$\alpha = 0 : \widehat{Y} = -0.367 + 0.416 \cdot X, \quad \|\tilde{\boldsymbol{w}}\|^2 = \left\|(-0.367, 0.416)^T\right\|^2 = 0.308, \quad SSE = 6.34$$

$$\alpha = 10 : \widehat{Y} = -0.244 + 0.388 \cdot X, \quad \|\tilde{\boldsymbol{w}}\|^2 = \left\|(-0.244, 0.388)^T\right\|^2 = 0.210, \quad SSE = 6.75$$

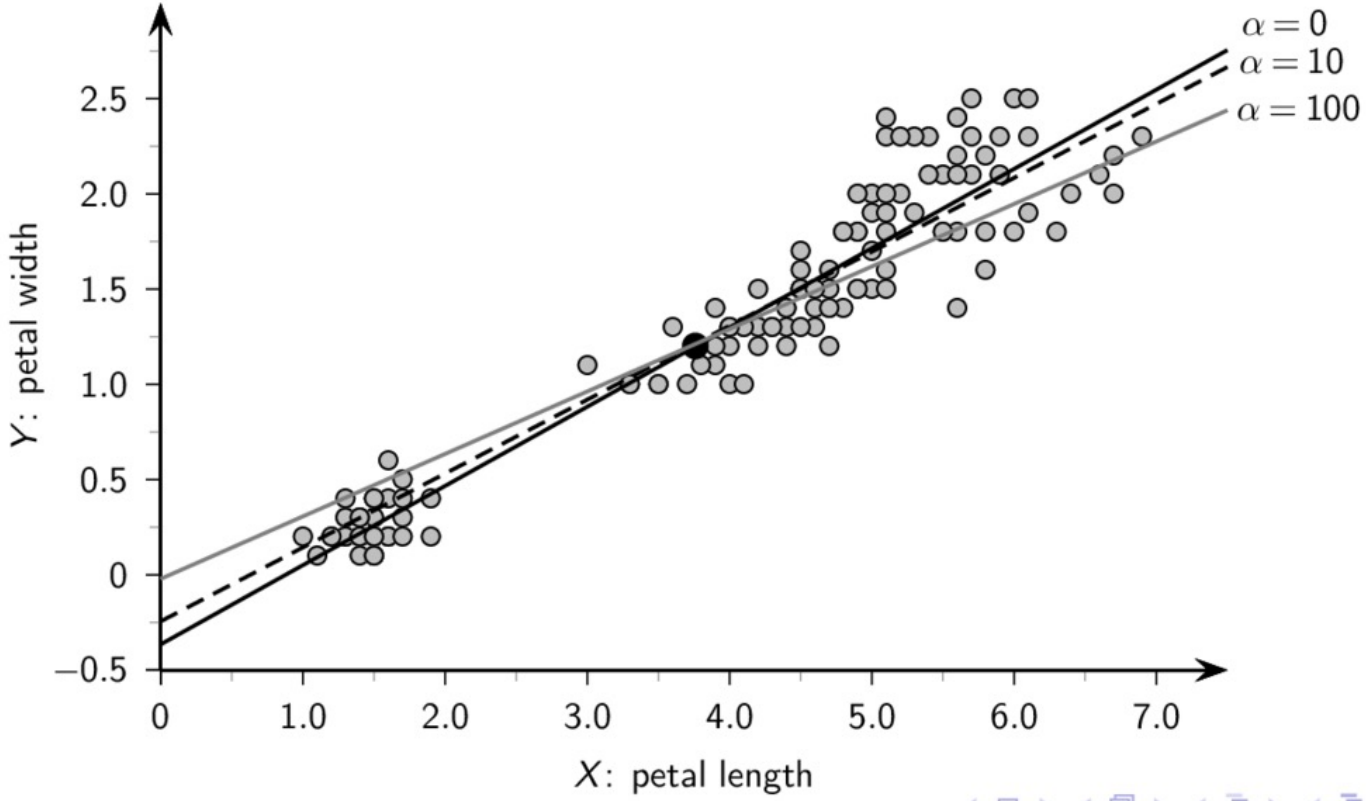
$$\alpha = 100 : \widehat{Y} = -0.021 + 0.328 \cdot X, \quad \|\tilde{\boldsymbol{w}}\|^2 = \left\|(-0.021, 0.328)^T\right\|^2 = 0.108, \quad SSE = 9.97$$

Ridge Regression

Example

As α increases there is more emphasis on minimizing the squared norm of $\tilde{\boldsymbol{w}}$.

Since $\|\tilde{\boldsymbol{w}}\|^2$ is more constrained as α increases, the fit of the model decreases, as seen from the increase in SSE values.



Ridge Regression: Unpenalized Bias Term

Often in L_2 regularized regression we do not want to penalize the bias term w_0, since it simply provides the intercept information.

Consider the new regularized objective where $\boldsymbol{w} = (w_1, w_2, \cdots, w_d)^T$ without w_0:

$$\begin{aligned}\min_{\boldsymbol{w}} \; J(\boldsymbol{w}) &= \|Y - w_0 \cdot 1 - \boldsymbol{D}\boldsymbol{w}\|^2 + \alpha \cdot \|\boldsymbol{w}\|^2 \\ &= \left\|Y - w_0 \cdot 1 - \sum_{i=1}^{d} w_i \cdot X_i\right\|^2 + \alpha \cdot \left(\sum_{i=1}^{d} w_i^2\right)\end{aligned}$$

Therefore, we have

$$\min_{\boldsymbol{w}} \; J(\boldsymbol{w}) = \left\|\overline{Y} - \overline{\boldsymbol{D}}\boldsymbol{w}\right\|^2 + \alpha \cdot \|\boldsymbol{w}\|^2$$

where $\overline{Y} = Y - \mu_Y \cdot 1$ is the centered Y, and $\overline{\boldsymbol{D}} = \boldsymbol{D} - 1\boldsymbol{\mu}^T$ is the centered $\boldsymbol{D}$.

We can exclude w_0 from the L_2 regularization objective by centering the response vector and the unaugmented data matrix.

Ridge Regression: Unpenalized Bias

Example

When we do not penalize w_0, we obtain the following lines of best fit for different values of the regularization constant α:

$$\alpha = 0 : \widehat{Y} = -0.365 + 0.416 \cdot X \qquad w_0^2 + w_1^2 = 0.307 \qquad SSE = 6.34$$

$$\alpha = 10 : \widehat{Y} = -0.333 + 0.408 \cdot X \qquad w_0^2 + w_1^2 = 0.277 \qquad SSE = 6.38$$

$$\alpha = 100 : \widehat{Y} = -0.089 + 0.343 \cdot X \qquad w_0^2 + w_1^2 = 0.125 \qquad SSE = 8.87$$

We observe that for $\alpha = 10$, when we penalize w_0, we obtain the following model:

$$\alpha = 10 : \widehat{Y} = -0.244 + 0.388 \cdot X \qquad w_0^2 + w_1^2 = 0.210 \qquad SSE = 6.75$$

As expected, we obtain a higher bias term when we do not penalize w_0.

Ridge Regression: Stochastic Gradient Descent

Instead of inverting the matrix $(\tilde{\boldsymbol{D}}^T\tilde{\boldsymbol{D}} + \alpha \cdot \boldsymbol{I})$ as called for in the exact ridge regression solution, we can employ the stochastic gradient descent algorithm.

The gradient of $\tilde{\boldsymbol{w}}$ multiplied by 1/2 for convenience is:

$$\nabla_{\tilde{\boldsymbol{w}}} = \frac{\partial}{\partial \tilde{\boldsymbol{w}}} J(\tilde{\boldsymbol{w}}) = -\tilde{\boldsymbol{D}}^T Y + (\tilde{\boldsymbol{D}}^T\tilde{\boldsymbol{D}})\tilde{\boldsymbol{w}} + \alpha \cdot \tilde{\boldsymbol{w}}$$

Using (batch) gradient descent, we can iteratively compute $\tilde{\boldsymbol{w}}$ as follows

$$\tilde{\boldsymbol{w}}^{t+1} = \tilde{\boldsymbol{w}}^t - \eta \cdot \nabla_{\tilde{\boldsymbol{w}}} = (1 - \eta \cdot \alpha)\tilde{\boldsymbol{w}}^t + \eta \cdot \tilde{\boldsymbol{D}}^T(Y - \tilde{\boldsymbol{D}} \cdot \tilde{\boldsymbol{w}}^t)$$

In SGD, we update the weight vector by considering only one (random) point at each time:

$$\tilde{\boldsymbol{w}}^{t+1} = \tilde{\boldsymbol{w}}^t - \eta \cdot \nabla_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}_k) = \left(1 - \frac{\eta \cdot \alpha}{n}\right)\tilde{\boldsymbol{w}}^t + \eta \cdot (y_k - \tilde{\boldsymbol{x}}_k \cdot \tilde{\boldsymbol{w}}^t) \cdot \tilde{\boldsymbol{x}}_k$$

Ridge Regression: SGD Algorithm

Ridge Regression: SGD $(\boldsymbol{D}, Y, \eta, \epsilon)$**:**

$\tilde{\boldsymbol{D}} \leftarrow \begin{pmatrix} 1 & \boldsymbol{D} \end{pmatrix}$ // augment data

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

$\tilde{\boldsymbol{w}}^t \leftarrow$ random vector in $\mathbb{R}^{d+1}$ // initial weight vector

repeat

  foreach $k = 1, 2, \cdots, n$ *(in random order)* **do**

    $\nabla_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}_k) \leftarrow -(y_k - \tilde{\boldsymbol{x}}_k^T \tilde{\boldsymbol{w}}^t) \cdot \tilde{\boldsymbol{x}}_k + \frac{\alpha}{n} \cdot \tilde{\boldsymbol{w}}$ // gradient at $\tilde{\boldsymbol{x}}_k$

    $\tilde{\boldsymbol{w}}^{t+1} \leftarrow \tilde{\boldsymbol{w}}^t - \eta \cdot \nabla_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}_k)$ // update estimate for w_k

  $t \leftarrow t + 1$

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

Ridge Regression: SGD

Example

We apply ridge regression on the Iris dataset ($n = 150$), using `petal length` (X) as the independent attribute, and `petal width` (Y) as the response variable.

Using SGD (with $\eta = 0.001$ and $\epsilon = 0.0001$) we obtain different lines of best fit for different values of the regularization constant α:

$$\alpha = 0 : \widehat{Y} = -0.366 + 0.413 \cdot X \qquad SSE_{SGD} = 6.37 \qquad SSE_{Ridge} = 6.34$$

$$\alpha = 10 : \widehat{Y} = -0.244 + 0.387 \cdot X \qquad SSE_{SGD} = 6.76 \qquad SSE_{Ridge} = 6.38$$

$$\alpha = 100 : \widehat{Y} = -0.022 + 0.327 \cdot X \qquad SSE_{SGD} = 10.04 \qquad SSE_{Ridge} = 8.87$$

L_1 Regression

Example

We can contrast the coefficients for L_2 (ridge) and L_1 (Lasso) regression by comparing models with the same level of squared error.

For $\alpha = 5$, the L_1 model has $SSE = 8.82$.

We adjust the ridge value in L_2 regression, with $\alpha = 35$ resulting in a similar SSE value. The two models are given as follows:

$$L_1: \ \widehat{Y} = -0.553 + 0.0 \cdot X_1 + 0.0 \cdot X_2 + 0.359 \cdot X_3 + 0.170 \cdot X_4 \qquad \|\boldsymbol{w}\|_1 = 0.156$$

$$L_2: \ \widehat{Y} = -0.394 + 0.019 \cdot X_1 - 0.051 \cdot X_2 + 0.316 \cdot X_3 + 0.212 \cdot X_4 \qquad \|\boldsymbol{w}\|_1 = 0.598$$

L_2: the coefficients for X_1 and X_2 are small, and therefore less important, but they are not zero.

L_1: the coefficients for X_1 and X_2 are exactly zero, leaving only X_3 and X_4;

Lasso can thus act as an automatic feature selection approach.

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

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 Y, its PMF for $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$ is:

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

where $\pi(\tilde{\boldsymbol{x}})$ denotes the probability of $Y = 1$ given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$.

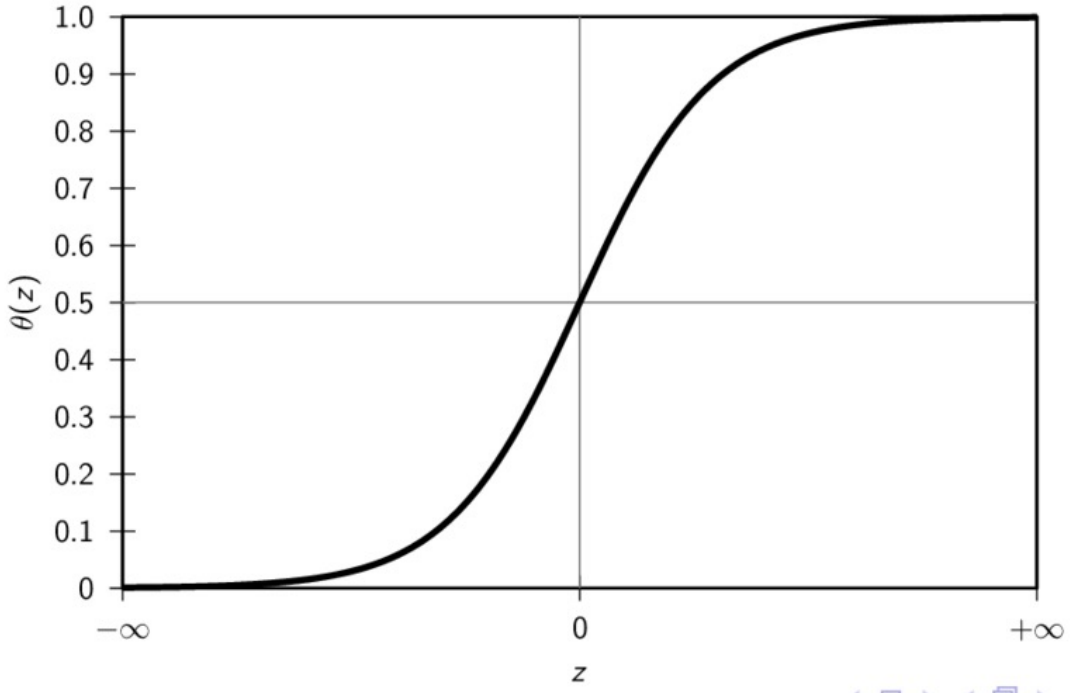
Instead of directly predicting the response value, the goal is to learn the probability, $P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})$, which is also the expected value of Y given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$.

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

Logistic Function

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

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



Logistic Function

Example

Consider what happens when z is $-\infty$, $+\infty$ and 0:

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

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

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

$z = 0$ acts as a threshold, since, for $z > 0$, $\theta(z) > 0.5$, and, for $z < 0$, $\theta(z) < 0.5$.

Interpreting $\theta(z)$ as a probability, the larger the z value, the higher the probability.

Another interesting property of the logistic function is that

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

Binary Logistic Regression

We define the logistic regression model as follows:

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

The probability that $Y=1$ is the output of the logistic function for the input $\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{x}}$. The probability for $Y=0$ is given as

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

that is, $1-\theta(z)=\theta(-z)$ for $z=\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{x}}$.

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

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

Since Y is a Bernoulli binary random variable, $P(Y|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\theta(\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{x}})$ when $Y=1$ and $P(Y|\,\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{x}})=\theta(-\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{x}})$ when $Y=0$.

Log-Odds Ratio

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

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

$$= \frac{\exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}}{1 + \exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}} \cdot \left(1 + \exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}\right)$$

$$= \boxed{\exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}}$$

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

$$\ln\big(\text{odds}(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})\big) = \ln\left(\frac{P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}{1 - P(Y = 1 | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}\right) = \ln\left(\exp\{\tilde{\omega}^T \tilde{\boldsymbol{x}}\}\right) = \tilde{\omega}^T \tilde{\boldsymbol{x}}$$

$$= \omega_0 \cdot x_0 + \omega_1 \cdot x_1 + \cdots + \omega_d \cdot x_d$$

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.

Consider the effect of attribute X_i by fixing the values for all other attributes:

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

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

where C is a constant comprising the fixed attributes.

ω_i can 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 $\tilde{\boldsymbol{w}}$.

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

Instead of maximizing the likelihood, we can maximize the *log-likelihood*, to convert the product into a summation:

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

Maximum Likelihood Estimation

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

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

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

Maximum Likelihood Estimation

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 presented,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:

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

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

After solving the derivative:

$$\nabla(\tilde{\boldsymbol{w}}) = \sum_{i=1}^{n} \big(y_i - \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i)\big) \cdot \tilde{\boldsymbol{x}}_i$$

Maximum Likelihood Estimation

The gradient ascent method starts at $\tilde{\boldsymbol{w}}^0$.

At each step t, the method moves in the direction of steepest ascent, which is given by the gradient vector.

Given the current $\tilde{\boldsymbol{w}}^t$, the next estimate is:

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

$\eta > 0$ is 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, which is called *stochastic gradient ascent* (SGA).

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

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

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

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**

    $\nabla(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}}_i) \leftarrow \left(y_i - \theta(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_i)\right) \cdot \tilde{\boldsymbol{x}}_i$ // 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 $\left\| \tilde{\boldsymbol{w}}^t - \tilde{\boldsymbol{w}}^{t-1} \right\| \leq \epsilon$

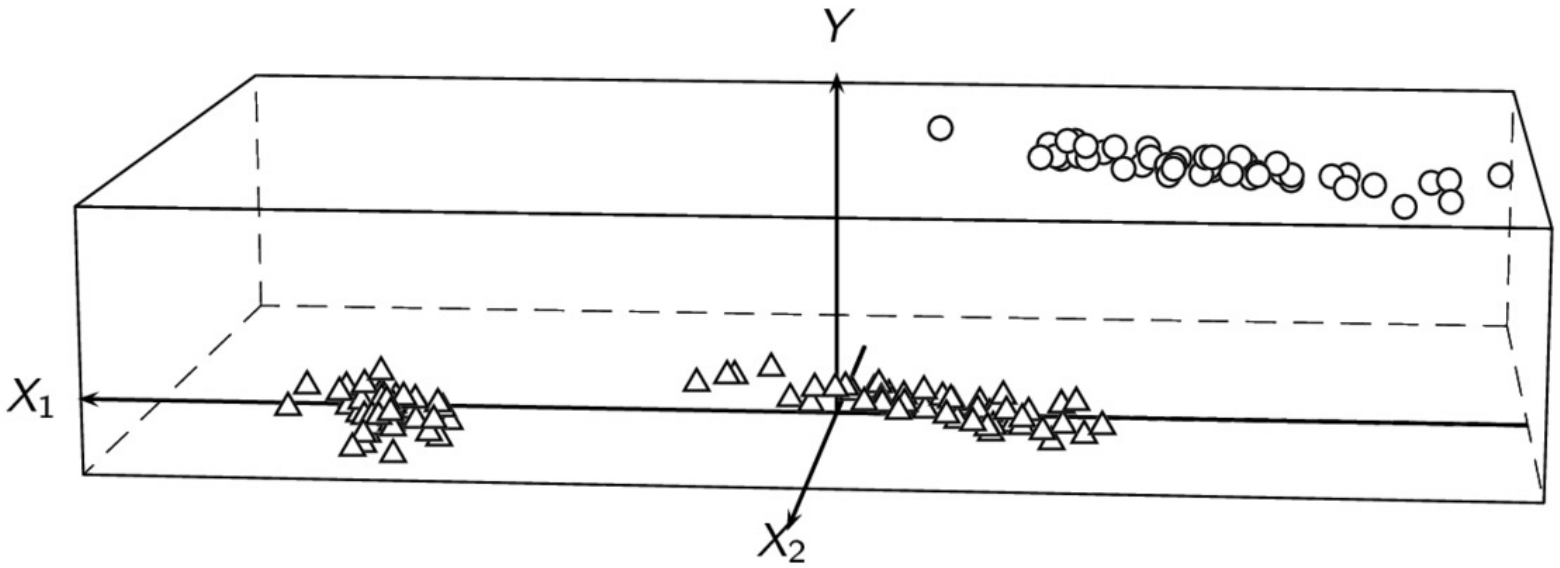
Logistic Regression

Example

Figure shows the Iris principal components data.

X_1 and X_2 are the independent attributes and represent the first two principal components.

Y is the binary response variable and 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`.



Linear Regression

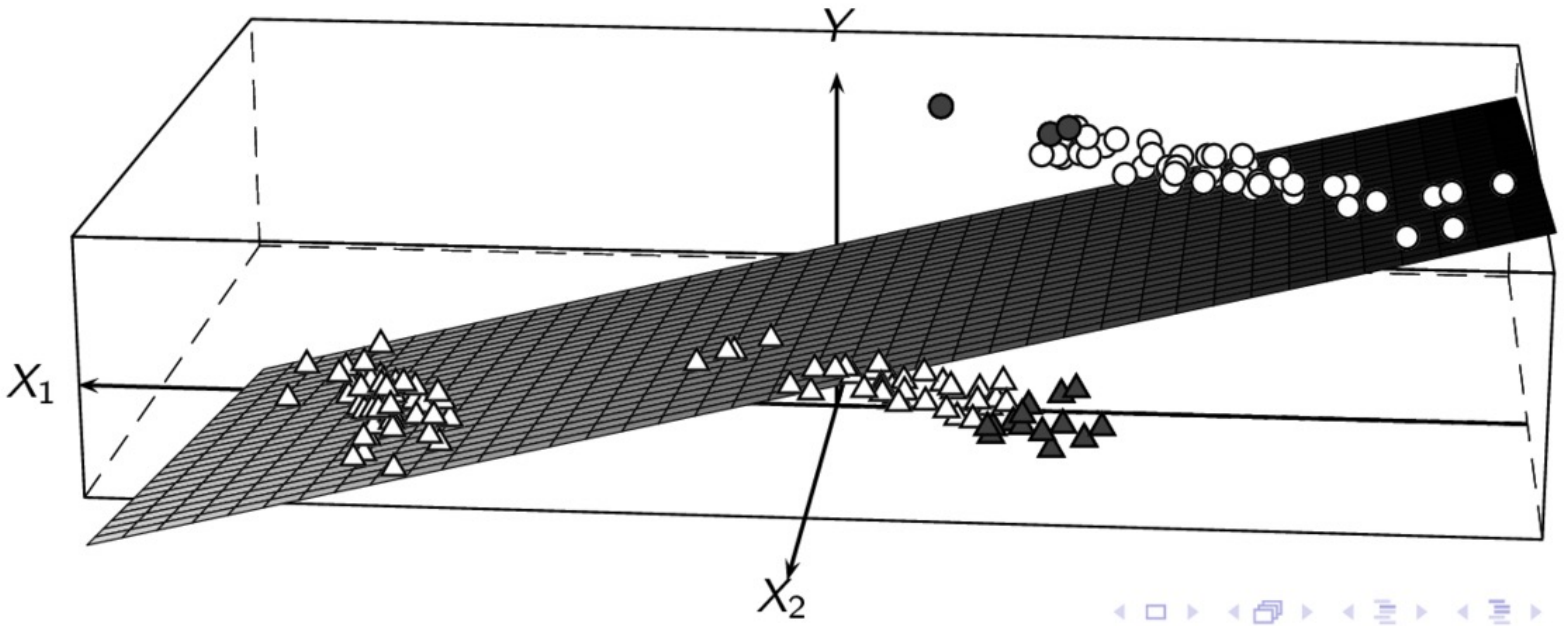
Example

The plane of best fit in linear regression has 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 Y is binary, we predict $y = 1$ if $f(\tilde{\boldsymbol{x}}) \geq 0.5$, and $y = 0$ otherwise.

Linear regression misclassifies 17 points, achieving 88.7% accuracy.



Logistic Regression

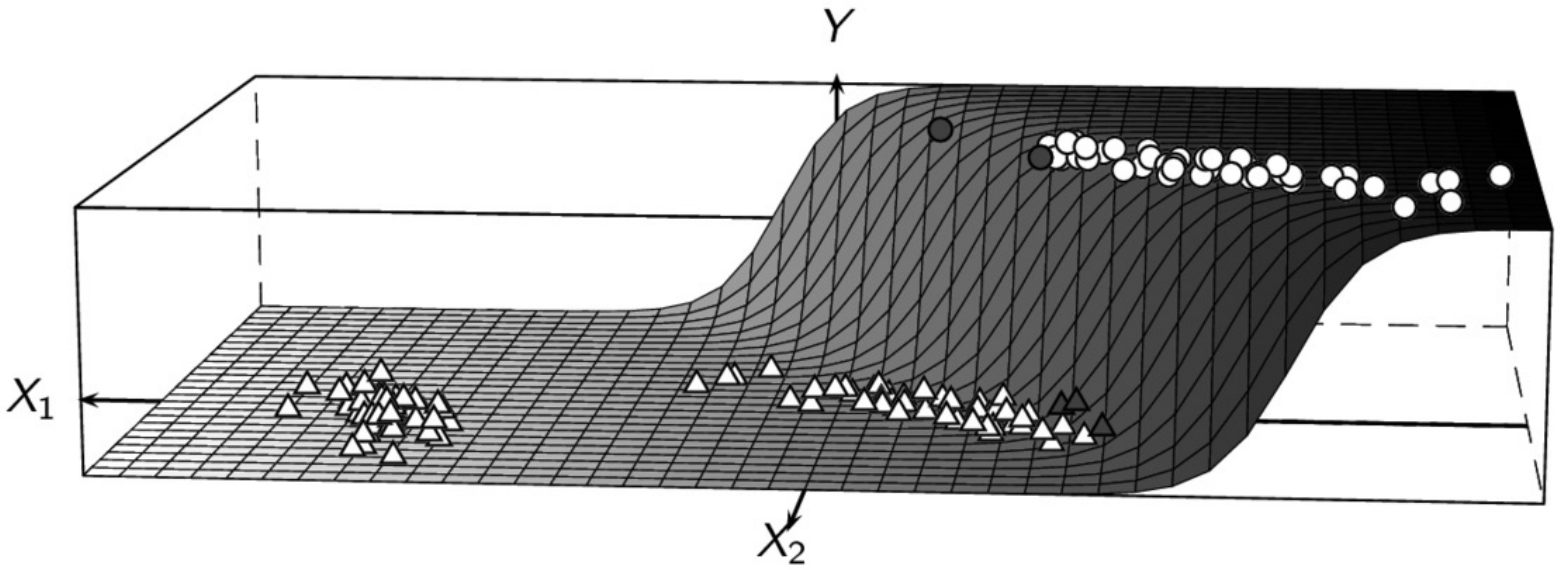
The fitted logistic model is given as

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

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

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

Logistic regression misclassifies only 5 points, achieving 96.7% accuracy.



Multiclass Logistic Regression

We now generalize logistic regression to the case when 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

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

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 PMF of $\boldsymbol{Y}$ is:

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

We select c_K as a reference or base class, and consider the log-odds ratio of the other classes w.r.t. c_K.

We assume these log-odd ratios are linear in $\tilde{\boldsymbol{X}}$, but $\tilde{\omega}_i$ is specific to for class c_i:

$$\ln(\text{odds}(\boldsymbol{Y} = \boldsymbol{e}_i|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})) = \ln\left(\frac{P(\boldsymbol{Y} = \boldsymbol{e}_i|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}{P(\boldsymbol{Y} = \boldsymbol{e}_K|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}})}\right) = \ln\left(\frac{\pi_i(\tilde{\boldsymbol{x}})}{\pi_K(\tilde{\boldsymbol{x}})}\right) = \tilde{\omega}_i^T \tilde{\boldsymbol{x}}$$

$$= \omega_{i0} \cdot x_0 + \omega_{i1} \cdot x_1 + \cdots + \omega_{id} \cdot x_d$$

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:

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

This function is also called the *softmax* function.

When $K = 2$, this formulation yields exactly the same model as in binary logistic regression.

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.

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:

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

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 w.r.t. $\tilde{\boldsymbol{w}}_j$ at a given point $\tilde{\boldsymbol{x}}_i$ is:

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

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

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

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 $\hat{y}$ for any new $\tilde{\boldsymbol{z}}$ as:

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

We evalute the softmax function and the predicted $\hat{y}$ has 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** $\tilde{\boldsymbol{w}}_j \leftarrow \tilde{\boldsymbol{w}}_j^t$ // copy $\tilde{\boldsymbol{w}}_j^t$
  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

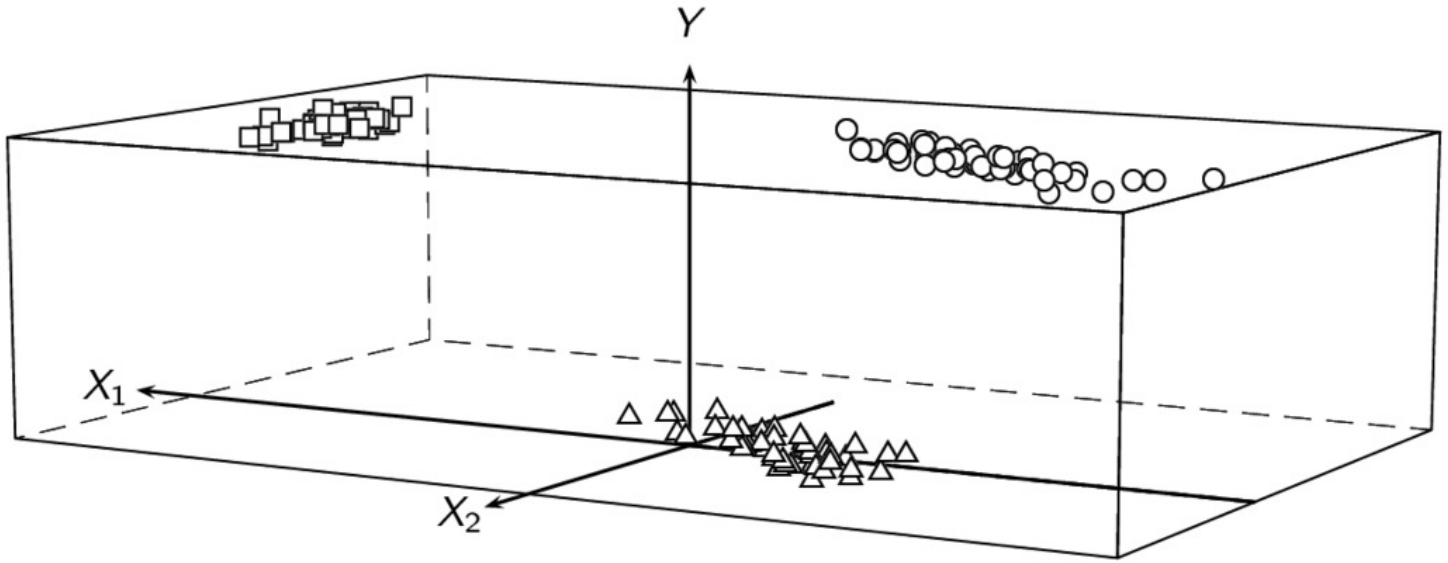
Example

Consider the 2D Iris PCA dataset, $n = 150$.

Y takes on three values: $Y = c_1$: `Iris-setosa` ($\square$), $Y = c_2$: `Iris-versicolor` ($\bigcirc$) and $Y = c_3$: `Iris-virginica` ($\triangle$).

$Y = c_1$ to $\boldsymbol{e}_1 = (1,0,0)^T$, $Y = c_2$ to $\boldsymbol{e}_2 = (0,1,0)^T$ and $Y = c_3$ to $\boldsymbol{e}_3 = (0,0,1)^T$.

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

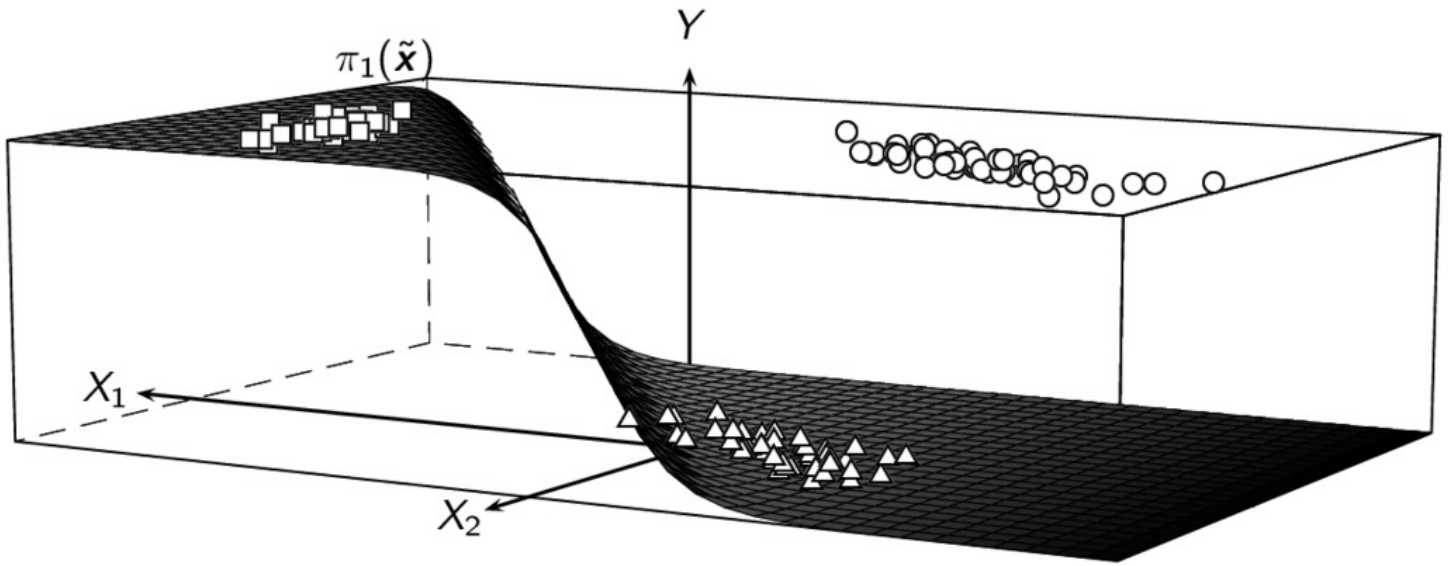
Example

We use $Y = c_3$ as the reference or base class. The fitted model is:

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

The decision surface corresponding to c_1 is:

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



Multiclass Logistic Regression

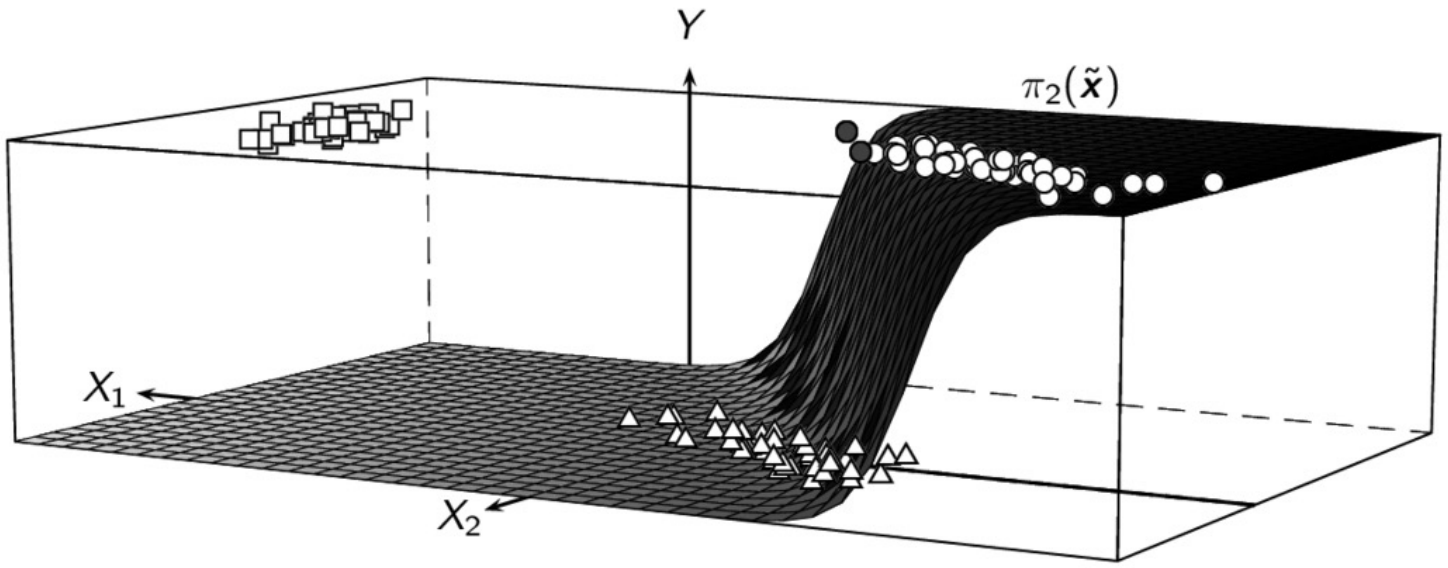
Example

We use $Y = c_3$ as the reference or base class. The fitted model is:

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

The decision surface corresponding to c_2 is:

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



Multiclass Logistic Regression

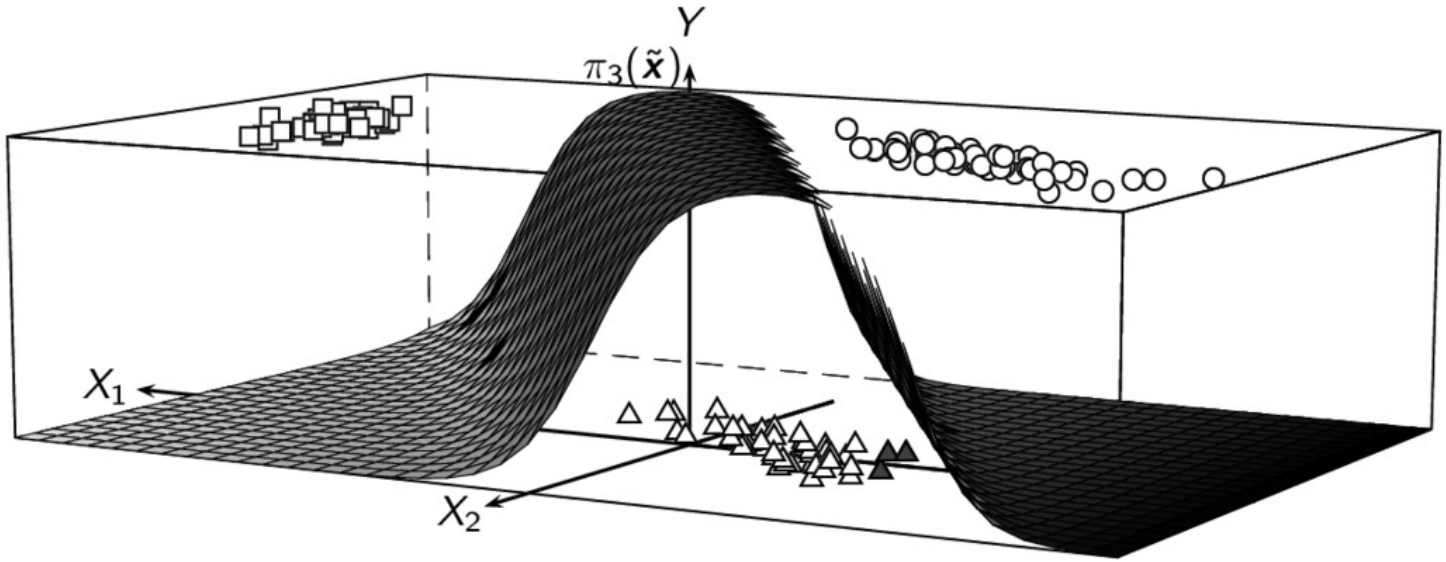
Example

We use $Y = c_3$ as the reference or base class. The fitted model is:

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

The decision surface corresponding to c_3 is:

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



Multiclass Logistic Regression

Example

The training set accuracy 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:

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

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