COMS30035, Machine learning: Linear Regresssion, Linear Discriminant and Logistic Regression

James Cussens

School of Computer Science
University of Bristol

11th September 2024

Acknowledgement

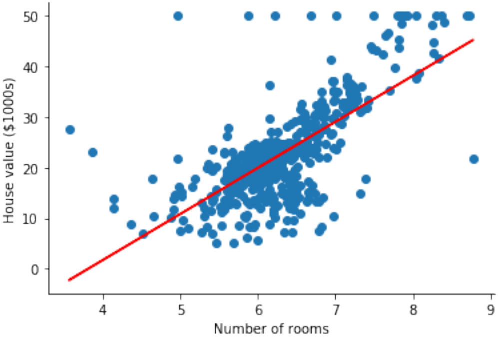
- These slides are adapted from ones originally created by Rui Ponte Costa and Dima Damen and later edited by Edwin Simpson.

Agenda

- Linear regression
- Nonlinear regression
- Probabilistic models
- Maximum likelihood estimation
- Discriminant functions
- Logistic regression

Revisiting regression

- Goal: Finding a relationship between two variables (e.g. regress *house value* against *number of rooms*)
- Model: Linear relationship between *house value* and *number of rooms*?



Revisiting regression – deterministic model

Data: a set of data points $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ where x_i is the number of rooms of house i and y_i the house value.

Task: build a model that can predict the house value from the number of rooms

Model Type: parametric; assumes a polynomial relationship between house value and number of rooms

Model Complexity: assume the relationship is linear
house value $= a_0 + a_1 \times$ rooms

$$y_i = a_0 + a_1 x_i \tag{1}$$

Model Parameters: model has two parameters a_0 and a_1 which should be estimated.

- a_0 is the y-intercept
- a_1 is the slope of the line

Revisiting regression – deterministic model

Data: a set of data points $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ where x_i is the number of rooms of house i and y_i the house value.

Task: build a model that can predict the house value from the number of rooms

Model Type: parametric; assumes a polynomial relationship between house value and number of rooms

Model Complexity: assume the relationship is linear
house value $= a_0 + a_1 \times$ rooms

$$y_i = a_0 + a_1 x_i \tag{1}$$

Model Parameters: model has two parameters a_0 and a_1 which should be estimated.

- a_0 is the y-intercept
- a_1 is the slope of the line

Revisiting regression – deterministic model

Data: a set of data points $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ where x_i is the number of rooms of house i and y_i the house value.

Task: build a model that can predict the house value from the number of rooms

Model Type: parametric; assumes a polynomial relationship between house value and number of rooms

Model Complexity: assume the relationship is linear
house value $= a_0 + a_1 \times$ rooms

$$y_i = a_0 + a_1 x_i \tag{1}$$

Model Parameters: model has two parameters a_0 and a_1 which should be estimated.

- a_0 is the y-intercept
- a_1 is the slope of the line

Revisiting regression – deterministic model

Data: a set of data points $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ where x_i is the number of rooms of house i and y_i the house value.

Task: build a model that can predict the house value from the number of rooms

Model Type: parametric; assumes a polynomial relationship between house value and number of rooms

Model Complexity: assume the relationship is linear
house value $= a_0 + a_1 \times$ rooms

$$y_i = a_0 + a_1 x_i \tag{1}$$

Model Parameters: model has two parameters a_0 and a_1 which should be estimated.

- a_0 is the y-intercept
- a_1 is the slope of the line

Revisiting linear regression – fitting

- Although we are currently assuming the model is linear, the data will not typically be consistent with such an assumption. So ...
- Find $\theta = (a_0, a_1)$ which minimises

$$R(a_0, a_1) = \sum_{i=1}^{N} (y_i - (a_0 + a_1 x_i))^2 \tag{2}$$

- This is the sum of squared residuals.
- Using matrix notation we have:

$$\frac{\partial R}{\partial \theta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\theta)$$

- Here $X_{i,j}$ is the value of the jth feature in the ith datapoint (where we add a fake feature which is always 1 to handle the intercept), and y_i is the value of the *response* in the ith datapoint.
- Setting $\frac{\partial R}{\partial \theta}$ to 0 and re-arranging we get:

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Linear regression for nonlinear models

- For a polynomial of degree $p + 1$ we use (note: $p > 1$ gives nonlinear regression)

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_p \; x_i^p \tag{3}$$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \frac{1}{20}\begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \frac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \frac{1}{20}\begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \tfrac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \frac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$$y = 1.8 + 2.9x$$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \tfrac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \frac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \tfrac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \frac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$y = 1.8 + 2.9x$

Least Squares Solution

Example

Find the best least squares fit by a linear function to the data using $p = 1$

x	-1	0	1	2
y	0	1	3	9

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \tfrac{1}{20} \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 3 \\ 9 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 2.9 \end{bmatrix}$$

$$y = 1.8 + 2.9x$$

Regression with probabilistic models

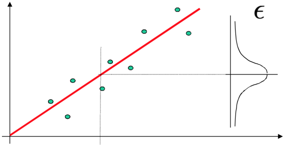
Probabilistic models are a core part of ML, as they allow us to also capture the uncertainty the model has about the data, which is critical for real world applications. For simplicity, lets drop a_0 from the previous model and add a random variable ϵ that captures the uncertainty

$$\text{house price} = a_1 \times \text{number of rooms} + \epsilon$$

We can assume, for example, that ϵ is given by $\mathcal{N}(\mu = 0, \sigma^2)$ which gives the likelihood

$$p(\mathbf{y}|\mathbf{X}, \theta) = \prod_{i=1}^{N} p(\text{price}_i|\text{rooms}_i, \theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(\text{price}_i - a_1 \text{rooms}_i)^2}{\sigma^2}}$$

This model has two parameters: the slope a_1 and variance σ [1]



[1] Note that here $\mu = a_0$ which, for simplicity, we assume to be zero.

Regression with probabilistic models

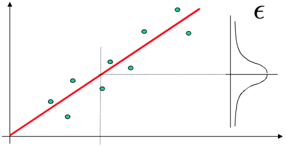
Probabilistic models are a core part of ML, as they allow us to also capture the uncertainty the model has about the data, which is critical for real world applications. For simplicity, lets drop a_0 from the previous model and add a random variable ϵ that captures the uncertainty

$$\text{house price} = a_1 \times \text{number of rooms} + \epsilon$$

We can assume, for example, that ϵ is given by $\mathcal{N}(\mu = 0, \sigma^2)$ which gives the likelihood

$$p(\mathbf{y}|\mathbf{X}, \theta) = \prod_{i=1}^{N} p(\text{price}_i|\text{rooms}_i, \theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(\text{price}_i - a_1 \text{rooms}_i)^2}{\sigma^2}}$$

This model has two parameters: the slope a_1 and variance σ [1]



[1] Note that here $\mu = a_0$ which, for simplicity, we assume to be zero.

Regression with probabilistic models

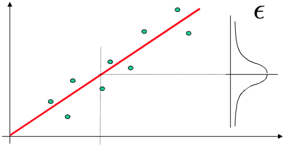
Probabilistic models are a core part of ML, as they allow us to also capture the uncertainty the model has about the data, which is critical for real world applications. For simplicity, lets drop a_0 from the previous model and add a random variable ϵ that captures the uncertainty

$$\text{house price} = a_1 \times \text{number of rooms} + \epsilon$$

We can assume, for example, that ϵ is given by $\mathcal{N}(\mu = 0, \sigma^2)$ which gives the likelihood

$$p(\mathbf{y}|\mathbf{X}, \theta) = \prod_{i=1}^{N} p(\text{price}_i|\text{rooms}_i, \theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(\text{price}_i - a_1 \text{rooms}_i)^2}{\sigma^2}}$$

This model has two parameters: the slope a_1 and variance σ [1]



[1] Note that here $\mu = a_0$ which, for simplicity, we assume to be zero.

Maximum Likelihood Estimation

- Similar to building deterministic models, probabilistic model parameters need to be tuned/trained
- **Maximum-likelihood estimation (MLE)** is a method of estimating the parameters of a probabilistic model.
- Assume θ is a vector of all parameters of the probabilistic model. (e.g. $\theta = \{a_1, \sigma\}$).
- **MLE** is an extremum estimator[2] obtained by maximising an objective function of θ

[2] "Extremum estimators are a wide class of estimators for parametric models that are calculated through maximization (or minimization) of a certain objective function, which depends on the data." wikipedia.org

Maximum Likelihood Estimation

- Similar to building deterministic models, probabilistic model parameters need to be tuned/trained
- **Maximum-likelihood estimation (MLE)** is a method of estimating the parameters of a probabilistic model.

- Assume $\boldsymbol{\theta}$ is a vector of all parameters of the probabilistic model. (e.g. $\boldsymbol{\theta} = \{a_1, \sigma\}$).
- **MLE** is an extremum estimator[2] obtained by maximising an objective function of $\boldsymbol{\theta}$

[2] "Extremum estimators are a wide class of estimators for parametric models that are calculated through maximization (or minimization) of a certain objective function, which depends on the data." wikipedia.org

Maximum Likelihood Estimation

Definition

Assume $f(\theta)$ is an objective function to be optimised (e.g. maximised), the *arg max* corresponds to the value of θ that attains the maximum value of the objective function f

$$\hat{\theta} = arg\ max_{\theta}\, f(\theta)$$

- Tuning the parameter is then equal to finding the maximum argument *arg max*

Maximum Likelihood Estimation

Definition

Assume $f(\theta)$ is an objective function to be optimised (e.g. maximised), the *arg max* corresponds to the value of θ that attains the maximum value of the objective function f

$$\hat{\theta} = \mathit{arg\ max}_{\theta}\, f(\theta)$$

- Tuning the parameter is then equal to finding the maximum argument *arg max*

Maximum Likelihood Estimation

Definition

Assume $f(\theta)$ is an objective function to be optimised (e.g. maximised), the *arg max* corresponds to the value of θ that attains the maximum value of the objective function f

$$\hat{\theta} = \mathit{arg\ max}_{\theta}\, f(\theta)$$

- Tuning the parameter is then equal to finding the maximum argument *arg max*

Maximum Likelihood Estimation - General

- Maximum Likelihood Estimation (MLE) is a common method for solving such problems

$$
\begin{aligned}
\theta_{MLE} &= arg\ max_{\theta}\ p(D|\theta) \\
&= arg\ max_{\theta}\ \ln p(D|\theta) \\
&= arg\ min_{\theta}\ -\ln p(D|\theta)
\end{aligned}
$$

MLE Recipe

1. Determine θ, D and expression for likelihood $p(D|\theta)$
2. Take the natural logarithm of the likelihood
3. Take the derivative of $\ln p(D|\theta)$ w.r.t. θ. If θ is a multi-dimensional vector, take partial derivatives
4. Set derivative(s) to 0 and solve for θ

Maximum Likelihood Estimation - General

- Maximum Likelihood Estimation (MLE) is a common method for solving such problems

$$
\begin{aligned}
\theta_{MLE} &= arg\ max_{\theta}\ p(D|\theta) \\
&= arg\ max_{\theta}\ \ln p(D|\theta) \\
&= arg\ min_{\theta}\ -\ln p(D|\theta)
\end{aligned}
$$

MLE Recipe

1. Determine θ, D and expression for likelihood $p(D|\theta)$
2. Take the natural logarithm of the likelihood
3. Take the derivative of $\ln p(D|\theta)$ w.r.t. θ. If θ is a multi-dimensional vector, take partial derivatives
4. Set derivative(s) to 0 and solve for θ

Maximum Likelihood Estimation - General

- Maximum Likelihood Estimation (MLE) is a common method for solving such problems

$$\begin{aligned} \theta_{MLE} &= arg\ max_\theta\ p(D|\theta) \\ &= arg\ max_\theta\ \ln p(D|\theta) \\ &= arg\ min_\theta\ -\ln p(D|\theta) \end{aligned}$$

MLE Recipe

1. Determine θ, D and expression for likelihood $p(D|\theta)$
2. Take the natural logarithm of the likelihood
3. Take the derivative of $\ln p(D|\theta)$ w.r.t. θ. If θ is a multi-dimensional vector, take partial derivatives
4. Set derivative(s) to 0 and solve for θ

Maximum Likelihood Estimation - General

- Maximum Likelihood Estimation (MLE) is a common method for solving such problems

$$
\begin{aligned}
\theta_{MLE} &= arg\ max_\theta\ p(D|\theta) \\
&= arg\ max_\theta\ \ln p(D|\theta) \\
&= arg\ min_\theta\ -\ln p(D|\theta)
\end{aligned}
$$

MLE Recipe

1. Determine θ, D and expression for likelihood $p(D|\theta)$
2. Take the natural logarithm of the likelihood
3. Take the derivative of $\ln p(D|\theta)$ w.r.t. θ. If θ is a multi-dimensional vector, take partial derivatives
4. Set derivative(s) to 0 and solve for θ

Maximum Likelihood Estimation - General

- Maximum Likelihood Estimation (MLE) is a common method for solving such problems

$$\begin{aligned}\theta_{MLE} &= arg\ max_{\theta}\ p(D|\theta) \\ &= arg\ max_{\theta}\ \ln p(D|\theta) \\ &= arg\ min_{\theta}\ -\ln p(D|\theta)\end{aligned}$$

MLE Recipe

1. Determine θ, D and expression for likelihood $p(D|\theta)$
2. Take the natural logarithm of the likelihood
3. Take the derivative of $\ln p(D|\theta)$ w.r.t. θ. If θ is a multi-dimensional vector, take partial derivatives
4. Set derivative(s) to 0 and solve for θ

Maximum Likelihood Estimation - General

- Maximum Likelihood Estimation (MLE) is a common method for solving such problems

$$\begin{aligned} \theta_{MLE} &= arg\ max_\theta\ p(D|\theta) \\ &= arg\ max_\theta\ \ln p(D|\theta) \\ &= arg\ min_\theta\ - \ln p(D|\theta) \end{aligned}$$

MLE Recipe

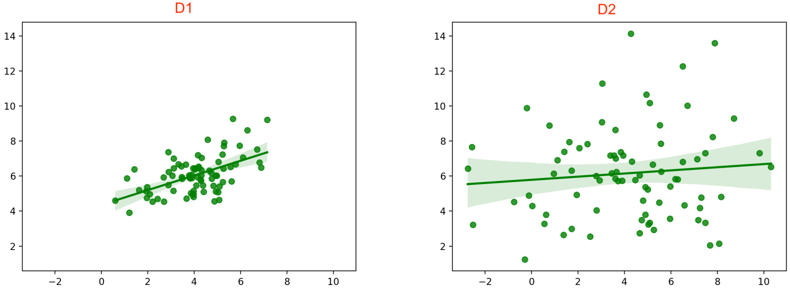
1. Determine θ, D and expression for likelihood $p(D|\theta)$
2. Take the natural logarithm of the likelihood
3. Take the derivative of $\ln p(D|\theta)$ w.r.t. θ. If θ is a multi-dimensional vector, take partial derivatives
4. Set derivative(s) to 0 and solve for θ

Least Squares and MLE for Linear Regression

- In the case of standard linear regression one can prove that the parameters which minimise the squared error are also the MLE parameters.
- Note that in general the MLE recipe just given will only find local maxima of the likelihood (so not necessarily the MLE).
- But in the special case of linear regression it does find the MLE.

Data Modelling - Deterministic vs Probabilistic

- **Probabilistic Models** can tell us more
- We could use the same MLE recipe to find σ_{ML}. This would tell us how uncertain our model is about the data D.
- For example: if we apply this method to two datasets (D_1 and D_2) what would the parameters $\theta = \{a_1, \sigma\}$ be?

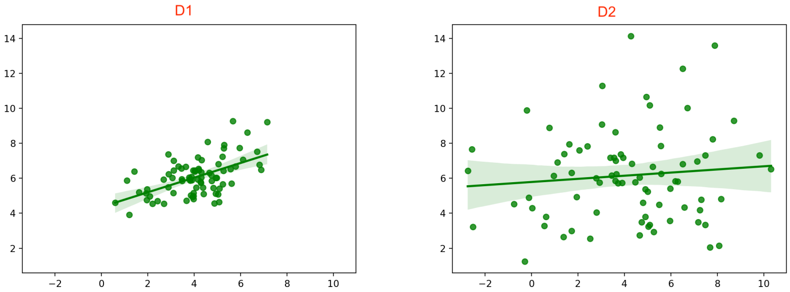


$$a_{1\,ML}^{D_1} > a_{1\,ML}^{D_2} \text{ [slope] and } \sigma_{ML}^{D_1} < \sigma_{ML}^{D_2} \text{ [uncertainty}^3\text{]}$$

[3] The uncertainty (σ) is represented by the light green bar in the plots. Test it yourself.

Data Modelling - Deterministic vs Probabilistic

- **Probabilistic Models** can tell us more
- We could use the same MLE recipe to find σ_{ML}. This would tell us how uncertain our model is about the data D.
- For example: if we apply this method to two datasets (D_1 and D_2) what would the parameters $\theta = \{a_1, \sigma\}$ be?

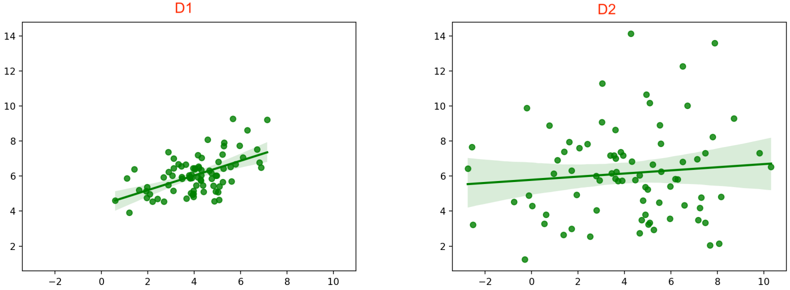


$$a_{1\,ML}^{D_1} > a_{1\,ML}^{D_2} \text{ [slope] and } \sigma_{ML}^{D_1} < \sigma_{ML}^{D_2} \text{ [uncertainty}^3\text{]}$$

[3] The uncertainty (σ) is represented by the light green bar in the plots. Test it yourself.

Data Modelling - Deterministic vs Probabilistic

- **Probabilistic Models** can tell us more
- We could use the same MLE recipe to find σ_{ML}. This would tell us how uncertain our model is about the data D.
- For example: if we apply this method to two datasets (D_1 and D_2) what would the parameters $\theta = \{a_1, \sigma\}$ be?

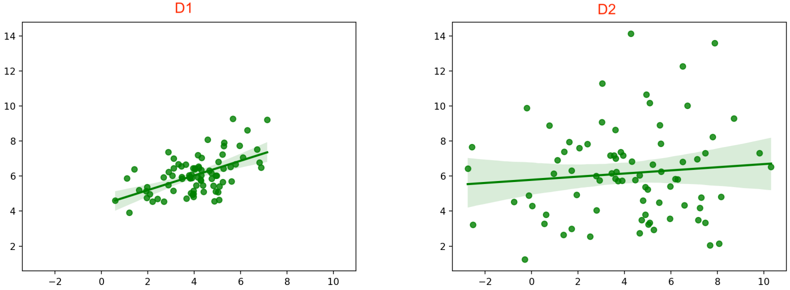


$a_{1\,ML}^{D_1} > a_{1\,ML}^{D_2}$ [slope] and $\sigma_{ML}^{D_1} < \sigma_{ML}^{D_2}$ [uncertainty[3]]

[3] The uncertainty (σ) is represented by the light green bar in the plots. Test it yourself.

Data Modelling - Deterministic vs Probabilistic

- **Probabilistic Models** can tell us more
- We could use the same MLE recipe to find σ_{ML}. This would tell us how uncertain our model is about the data D.
- For example: if we apply this method to two datasets (D_1 and D_2) what would the parameters $\theta = \{a_1, \sigma\}$ be?

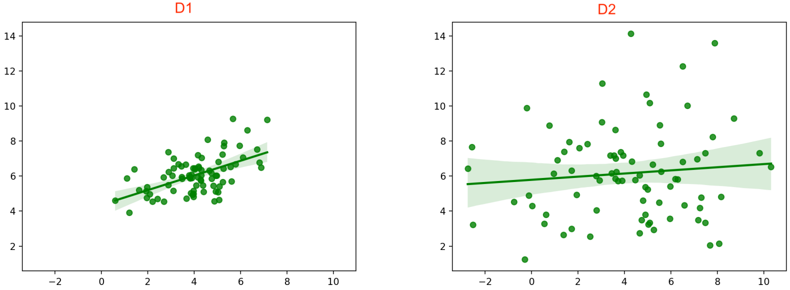


$a_{1}{}^{D_1}_{ML} > a_{1}{}^{D_2}_{ML}$ [slope] and $\sigma^{D_1}_{ML} < \sigma^{D_2}_{ML}$ [uncertainty[3]]

[3] The uncertainty (σ) is represented by the light green bar in the plots. Test it yourself.

Data Modelling - Deterministic vs Probabilistic

- **Probabilistic Models** can tell us more
- We could use the same MLE recipe to find σ_{ML}. This would tell us how uncertain our model is about the data D.
- For example: if we apply this method to two datasets (D_1 and D_2) what would the parameters $\theta = \{a_1, \sigma\}$ be?



$$a_{1\,ML}^{D_1} > a_{1\,ML}^{D_2} \text{ [slope] and } \sigma_{ML}^{D_1} < \sigma_{ML}^{D_2} \text{ [uncertainty}^3\text{]}$$

[3] The uncertainty (σ) is represented by the light green bar in the plots. Test it yourself.

Classification

- It is the classical example of **supervised learning**
- Goal: Classify input data into one of K classes
- Model: *Discriminant function*:
    - A function that takes an input vector x and assigns it to class C_k. For simplicity we will focus on $K = 2$ and will first study linear functions (see Bishop for the general cases).

Classification

- It is the classical example of **supervised learning**
- Goal: Classify input data into one of K classes
- Model: *Discriminant function*:
  - A function that takes an input vector x and assigns it to class C_k. For simplicity we will focus on $K = 2$ and will first study linear functions (see Bishop for the general cases).

Linear discriminant function

- The simplest linear discriminant (LD) is $\boxed{y(x) = w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}}$
  - where y is used to predicted class C_k, x is the input vector (feature values)
  - w_0 is a scalar, which we call *bias*
  - w_T is our vector of parameters, which we call *weights*
- This looks like linear regression! Except the next step...
- For $K = 2$: An input vector x is assigned to class C_1 if $y(x) \geq 0$ and to class C_2 otherwise.
- A direct application of least-squares to choose w_0 and $\mathbf{w}$ does not give great results. See Bishop §4.1.3
- Instead we can assume the data from each class has a Gaussian distribution whose mean is class-specific (but where the covariance matrix for each class is the same), use MLE to find parameters for each of these Gaussians and finally use Bayes theorem to assign classes. See scikit-learn explanation.

Linear discriminant function

- The simplest linear discriminant (LD) is $\boxed{y(x) = w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}}$
  - where y is used to predicted class C_k, x is the input vector (feature values)
  - w_0 is a scalar, which we call *bias*
  - w_T is our vector of parameters, which we call *weights*
- This looks like linear regression! Except the next step...
- For $K = 2$: An input vector x is assigned to class C_1 if $y(x) \geq 0$ and to class C_2 otherwise.
- A direct application of least-squares to choose w_0 and $\mathbf{w}$ does not give great results. See Bishop §4.1.3
- Instead we can assume the data from each class has a Gaussian distribution whose mean is class-specific (but where the covariance matrix for each class is the same), use MLE to find parameters for each of these Gaussians and finally use Bayes theorem to assign classes. See scikit-learn explanation.

Linear discriminant function

- The simplest linear discriminant (LD) is $\boxed{y(x) = w_0 + \boldsymbol{w^T x}}$
  - where y is used to predicted class C_k, x is the input vector (feature values)
  - w_0 is a scalar, which we call *bias*
  - w_T is our vector of parameters, which we call *weights*
- This looks like linear regression! Except the next step...
- For $K = 2$: An input vector x is assigned to class C_1 if $y(x) \geq 0$ and to class C_2 otherwise.
- A direct application of least-squares to choose w_0 and $\mathbf{w}$ does not give great results. See Bishop §4.1.3
- Instead we can assume the data from each class has a Gaussian distribution whose mean is class-specific (but where the covariance matrix for each class is the same), use MLE to find parameters for each of these Gaussians and finally use Bayes theorem to assign classes. See scikit-learn explanation.

Linear discriminant function

- The simplest linear discriminant (LD) is $\boxed{y(x) = w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}}$
  - where y is used to predicted class C_k, x is the input vector (feature values)
  - w_0 is a scalar, which we call *bias*
  - w_T is our vector of parameters, which we call *weights*
- This looks like linear regression! Except the next step...
- For $K = 2$: An input vector x is assigned to class C_1 if $y(x) \geq 0$ and to class C_2 otherwise.
- A direct application of least-squares to choose w_0 and $\mathbf{w}$ does not give great results. See Bishop §4.1.3
- Instead we can assume the data from each class has a Gaussian distribution whose mean is class-specific (but where the covariance matrix for each class is the same), use MLE to find parameters for each of these Gaussians and finally use Bayes theorem to assign classes. See scikit-learn explanation.

Linear discriminant function

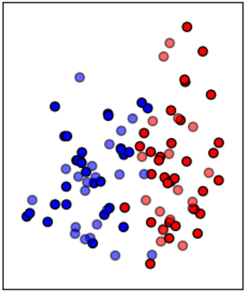
- The simplest linear discriminant (LD) is $\boxed{y(x) = w_0 + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}}$
  - where y is used to predicted class C_k, x is the input vector (feature values)
  - w_0 is a scalar, which we call *bias*
  - w_T is our vector of parameters, which we call *weights*

- This looks like linear regression! Except the next step...
- For $K = 2$: An input vector x is assigned to class C_1 if $y(x) \geq 0$ and to class C_2 otherwise.
- A direct application of least-squares to choose w_0 and **w** does not give great results. See Bishop §4.1.3
- Instead we can assume the data from each class has a Gaussian distribution whose mean is class-specific (but where the covariance matrix for each class is the same), use MLE to find parameters for each of these Gaussians and finally use Bayes theorem to assign classes. See scikit-learn explanation.

LD and linear separability

Example

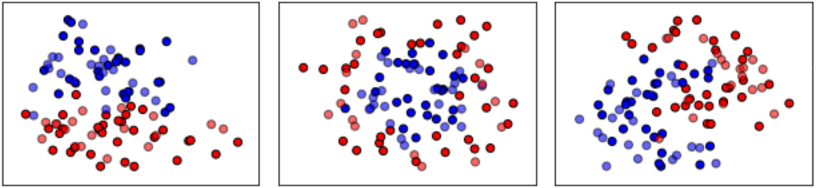
Linear separability is when two sets of points are separable by a line. We generated two sets of points using two Gaussians to illustrate this point, which can easily be fit by a LD. A *decision boundary* is the boundary that separates the two given classes, which our models will try to find.



Linear separability vs nonlinear separability

Example

Which datasets **are** and **are not** linearly separable[4]?



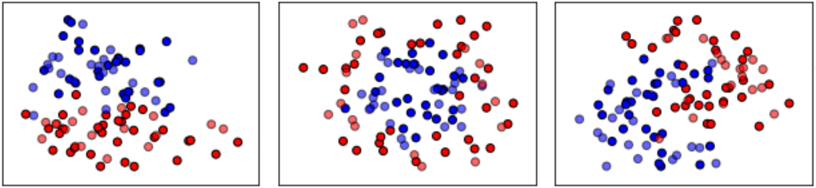
Only the first dataset is linearly separable!

[4] Example from Sklearn here.

Linear separability vs nonlinear separability

Example

Which datasets **are** and **are not** linearly separable[4]?



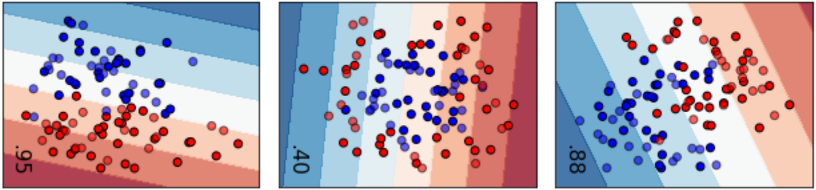
Only the first dataset is linearly separable!

[4] Example from Sklearn here.

Linear discriminant

Example

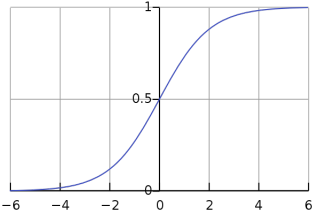
Using sklearn we fitted a LD to the data:



As expected, the LD model only does a good job in finding a good separation in the first dataset.

Logistic regression

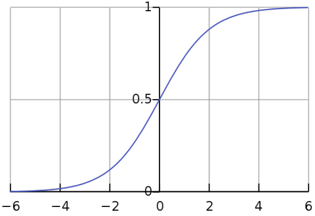
- We use a logistic function to obtain the probability of class C_k:
  $\boxed{y(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x})}$
  where σ denotes the logistic sigmoid function (s-shaped), for example:



- such that when $y \to 0$ we choose class 2 and $y \to 1$ class 1.
- Taking a probabilistic view:
  $p(C_1|\boldsymbol{x}) = y(\boldsymbol{x})$, and $p(C_2|\boldsymbol{x}) = 1 - p(C_1|\boldsymbol{x})$.

Logistic regression

- We use a logistic function to obtain the probability of class C_k:
  $\boxed{y(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})}$
  where σ denotes the logistic sigmoid function (s-shaped), for example:



- such that when $y \to 0$ we choose class 2 and $y \to 1$ class 1.
- Taking a probabilistic view:
  $p(C_1|\boldsymbol{x}) = y(\boldsymbol{x})$, and $p(C_2|\boldsymbol{x}) = 1 - p(C_1|\boldsymbol{x})$.

Logistic regression – maximum likelihood estimation

Follow MLE recipe:

1. Define likelihood: For a dataset $\{x_n, t_n\}$, where the targets $t_n \in \{0, 1\}$ we have $p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$ where $y_n = p(C_1|x_n)$. [5]
2. Take negative logarithm of the likelihood [6]:
$$-ln\, p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\, y_n + (1 - t_n) ln(1 - y_n)\}$$
3. Calculate the derivative w.r.t. the parameters $\boldsymbol{w}$:[7]
$$\frac{d\; ln\, p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w})}{d\boldsymbol{w}} = \sum_{n=1}^{N}(y_n - t_n)x_n$$
4. Now we can use Eq. above to directly update $\boldsymbol{w}$ using the data x.

---

[5] The exponent selects the probability of the target class (i.e. if $t_n = 1$ we get y_n; if $t_n = 0$ we get $1 - y_n$).

[6] Note that we used the logarithm product and power rule.

[7] This solution makes sense since we want to optimise the difference between the model output y and the desired targets t.

Logistic regression – maximum likelihood estimation

Follow MLE recipe:

1. Define likelihood: For a dataset $\{x_n, t_n\}$, where the targets $t_n \in \{0,1\}$ we have $p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$ where $y_n = p(C_1|x_n)$. [5]
2. Take negative logarithm of the likelihood [6]:
   $$-ln\ p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\ y_n + (1-t_n) ln(1-y_n)\}$$
3. Calculate the derivative w.r.t. the parameters $\boldsymbol{w}$:[7]
   $$\frac{d\ ln\ p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{w})}{d\boldsymbol{w}} = \sum_{n=1}^{N}(y_n - t_n)x_n$$
4. Now we can use Eq. above to directly update $\boldsymbol{w}$ using the data x.

[5] The exponent selects the probability of the target class (i.e. if $t_n = 1$ we get y_n; if $t_n = 0$ we get $1 - y_n$).

[6] Note that we used the logarithm product and power rule.

[7] This solution makes sense since we want to optimise the difference between the model output y and the desired targets t.

Logistic regression – maximum likelihood estimation

Follow MLE recipe:

1. Define likelihood: For a dataset $\{x_n, t_n\}$, where the targets $t_n \in \{0,1\}$ we have $p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$ where $y_n = p(C_1|x_n)$. [5]
2. Take negative logarithm of the likelihood [6]:
$$-ln\ p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\ y_n + (1-t_n) ln(1-y_n)\}$$
3. Calculate the derivative w.r.t. the parameters $\boldsymbol{w}$:[7]
$$\frac{d\ ln\ p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{w})}{d\boldsymbol{w}} = \sum_{n=1}^{N}(y_n - t_n)x_n$$
4. Now we can use Eq. above to directly update $\boldsymbol{w}$ using the data x.

---

[5] The exponent selects the probability of the target class (i.e. if $t_n = 1$ we get y_n; if $t_n = 0$ we get $1 - y_n$).

[6] Note that we used the logarithm product and power rule.

[7] This solution makes sense since we want to optimise the difference between the model output y and the desired targets t.

Logistic regression – maximum likelihood estimation

Follow MLE recipe:

1. Define likelihood: For a dataset $\{x_n, t_n\}$, where the targets $t_n \in \{0, 1\}$ we have $p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$ where $y_n = p(C_1|x_n)$. [5]
2. Take negative logarithm of the likelihood [6]:
$$-ln\, p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\, y_n + (1-t_n) ln(1-y_n)\}$$
3. Calculate the derivative w.r.t. the parameters $\boldsymbol{w}$:[7]
$$\frac{d\, ln\, p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w})}{d\boldsymbol{w}} = \sum_{n=1}^{N}(y_n - t_n)x_n$$
4. Now we can use Eq. above to directly update $\boldsymbol{w}$ using the data x.

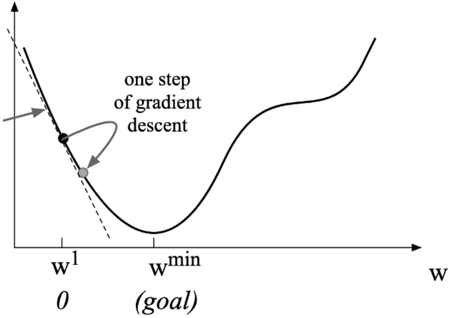
---

[5] The exponent selects the probability of the target class (i.e. if $t_n = 1$ we get y_n; if $t_n = 0$ we get $1 - y_n$).

[6] Note that we used the logarithm product and power rule.

[7] This solution makes sense since we want to optimise the difference between the model output y and the desired targets t.

MLE using Gradient Descent



- Start with random weight values
- We want to adjust each weight w to minimise negative log likelihood: move downhill to the minimum
- The derivative represents the slope: $\frac{d \ ln \ p(\mathbf{t}|\mathbf{x},\mathbf{w})}{d\mathbf{w}} = \sum_{n=1}^{N}(y_n - t_n)x_n$
- Increase or decrease w by a small amount in the downward direction
- Graph above illustrates gradient descent in general. In the particular case of logistic regression the error function is *convex* which means it has a unique minimum.

Logistic regression – maximum likelihood estimation

More details on calculating the derivative:

1. From here $-ln\, p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\, y_n + (1-t_n) ln(1-y_n)\}$
2. We get $\sum_{n=1}^{N}\{-\frac{t_n}{y_n} + \frac{(1-t_n)}{1-y_n}\}\{y_n(1-y_n)\}x_n$ [8]
3. The above simplifies to $\sum_{n=1}^{N}\{-t_n(1-y_n) + (1-t_n)y_n\}x_n$
4. And in turn to $\sum_{n=1}^{N}\{y_n - t_n\}x_n$ [9]

---

[8] We used the chain rule and $d\, ln(x) = 1/x$. We also used the derivative of the sigmoid $dy_n = y(1-y_n)$.

[9] You can find the full derivation here.

Logistic regression – maximum likelihood estimation

More details on calculating the derivative:

1. From here $-ln\ p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\ y_n + (1-t_n) ln(1-y_n)\}$
2. We get $\sum_{n=1}^{N}\{-\frac{t_n}{y_n} + \frac{(1-t_n)}{1-y_n}\}\{y_n(1-y_n)\}x_n$ [8]
3. The above simplifies to $\sum_{n=1}^{N}\{-t_n(1-y_n) + (1-t_n)y_n\}x_n$
4. And in turn to $\sum_{n=1}^{N}\{y_n - t_n\}x_n$ [9]

[8] We used the chain rule and $d\ ln(x) = 1/x$. We also used the derivative of the sigmoid $dy_n = y(1-y_n)$.

[9] You can find the full derivation here.

Logistic regression – maximum likelihood estimation

More details on calculating the derivative:

1. From here $-ln\ p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\ y_n + (1-t_n) ln(1-y_n)\}$
2. We get $\sum_{n=1}^{N}\{-\frac{t_n}{y_n} + \frac{(1-t_n)}{1-y_n}\}\{y_n(1-y_n)\}x_n$ [8]
3. The above simplifies to $\sum_{n=1}^{N}\{-t_n(1-y_n) + (1-t_n)y_n\}x_n$
4. And in turn to $\sum_{n=1}^{N}\{y_n - t_n\}x_n$ [9]

[8] We used the chain rule and $d\ ln(x) = 1/x$. We also used the derivative of the sigmoid $dy_n = y(1-y_n)$.

[9] You can find the full derivation here.

Logistic regression – maximum likelihood estimation

More details on calculating the derivative:

1. From here $-ln\ p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n ln\ y_n + (1-t_n) ln(1-y_n)\}$
2. We get $\sum_{n=1}^{N}\{-\frac{t_n}{y_n} + \frac{(1-t_n)}{1-y_n}\}\{y_n(1-y_n)\}x_n$ [8]
3. The above simplifies to $\sum_{n=1}^{N}\{-t_n(1-y_n) + (1-t_n)y_n\}x_n$
4. And in turn to $\sum_{n=1}^{N}\{y_n - t_n\}x_n$ [9]

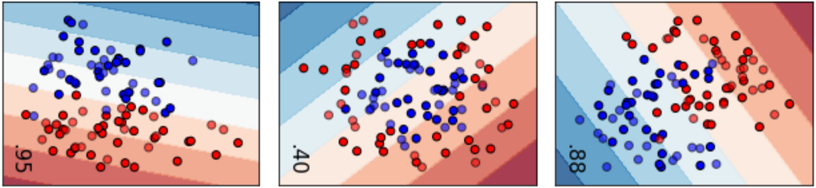
[8] We used the chain rule and $d\ ln(x) = 1/x$. We also used the derivative of the sigmoid $dy_n = y(1-y_n)$.

[9] You can find the full derivation here.

Logistic regression

Example

Using sklearn we fitted a logistic regression classifier to the data:



As you can see the results are very similar to LD, but because of probabilistic formulation we have an explicit probability of belonging to one or the other class (not shown); this can be very useful in real-world applications (e.g. self-driving cars or cancer detection).

Optimisation algorithms for logistic regression

- Although we have looked at simple gradient descent for logistic regression,
- the traditional optimisation algorithm (*iterative reweighted least squares*) also uses the Hessian - the matrix of second partial derivatives.
- For logistic regression, scikit-learn offers you a choice of no fewer than 6 optimisation algorithms to choose from!

Reading

- Bishop §3.1 up to end of §3.1.1.
- Bishop Chapter 4 up to end of §4.1.1.
- Bishop §4.3.2
- Murphy §4.2.1–§4.2.2
- Murphy §10.1–§10.2.3.1
- Murphy §11.1–§11.2.2.1

Problems and quizzes

- No problems.
- Quizzes:
  - Week 1: Regression
  - Week 1: Classification