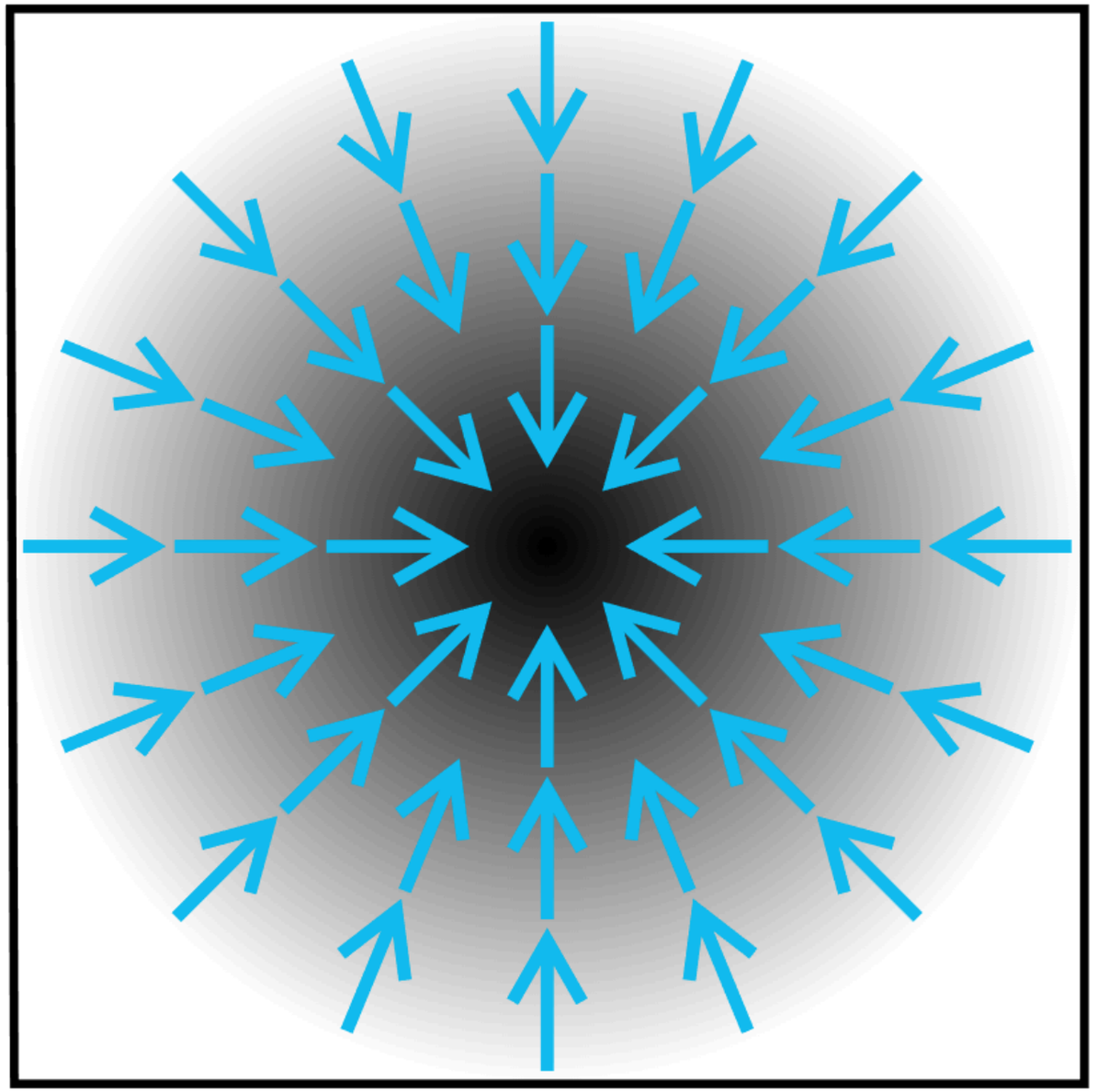


Logistic Regression: Loss Function and Gradient Descent

Leandro L. Minku

Outline

- Logistic regression optimisation problem
  - What problem needs to be solved to learn a logistic regression model?
  - Maximum likelihood estimation.
- Solving the logistic regression optimisation problem
  - Gradient descent.

Outline

- Logistic regression optimisation problem
  - What problem needs to be solved to learn a logistic regression model?
  - Maximum likelihood estimation.
- Solving the logistic regression optimisation problem
  - Gradient descent.

How to Learn $\mathbf{w}$?

$$\text{logit}(p_1) = \mathbf{w}^T\mathbf{x} \begin{cases} \mathbf{w}^T\mathbf{x} \geq 0 \rightarrow \text{class } 1 \\ \mathbf{w}^T\mathbf{x} < 0 \rightarrow \text{class } 0 \end{cases}$$

$$p_1 = \frac{e^{(\mathbf{w}^T\mathbf{x})}}{1 + e^{(\mathbf{w}^T\mathbf{x})}}$$

$$p_0 = 1 - p_1$$

- Given a training set (drawn i.i.d. from the true underlying distribution):

$$\mathcal{T} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

- Maximum likelihood estimation for supervised learning

Principle: the most reasonable values for $\mathbf{w}$ are the ones for which the "probability" of the observed examples is largest.

1. How likely would we get the output y for the input $\mathbf{x}$ for our training examples if the target distribution is really $\{p_1 = p(1 \mid \mathbf{x}, \mathbf{w}), p_0 = p(0 \mid \mathbf{x}, \mathbf{w})\}$?

2. Find the value $\mathbf{w}$ that maximises this quantity.

Likelihood

How likely would we get the output y for the input $\mathbf{x}$ for our training examples if the target distribution is really $\{p_1 = p(1 \mid \mathbf{x}, \mathbf{w}), p_0 = p(0 \mid \mathbf{x}, \mathbf{w})\}$?

$$\mathcal{T} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

For each example $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{T}$,
this is captured as $p_{y^{(i)}} = p(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})$, where $y^{(i)} \in \{0, 1\}$ — conditional likelihood

As examples are drawn i.i.d., jointly we have $\prod_{i=1}^{N} p_{y^{(i)}}$. — joint conditional likelihood

Probability vs likelihood: the term probability is usually used when we assume the model's parameters are reliable. The term likelihood is usually used when we're trying to determine whether the parameters in a model are good given the data.

Likelihood

$$\prod_{i=1}^{N} p_{y^{(i)}} = \prod_{i=1}^{N} p(y^{(i)} \,|\, \mathbf{x}^{(i)}, \mathbf{w}) = \quad p(\mathbf{y} \,|\, \mathbf{X}, \mathbf{w}) = \quad \mathscr{L}(\mathbf{w})$$

$$\mathscr{T} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

Design matrix: $\mathbf{X} = \begin{pmatrix} x_1^{(1)}, x_2^{(1)}, \cdots, x_d^{(1)} \\ x_1^{(2)}, x_2^{(2)}, \cdots, x_d^{(2)} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ x_1^{(N)}, x_2^{(N)}, \cdots, x_d^{(N)} \end{pmatrix}$

Vector of outputs: $\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}$

Maximum Likelihood Estimation

$$\prod_{i=1}^{N} p_{y^{(i)}} = \prod_{i=1}^{N} p(y^{(i)} \,|\, \mathbf{x}^{(i)}, \mathbf{w}) = \quad p(\mathbf{y} \,|\, \mathbf{X}, \mathbf{w}) = \quad \mathscr{L}(\mathbf{w})$$

$$\mathscr{T} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

Design matrix: $\mathbf{X} = \begin{pmatrix} x_1^{(1)}, x_2^{(1)}, \cdots, x_d^{(1)} \\ x_1^{(2)}, x_2^{(2)}, \cdots, x_d^{(2)} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ x_1^{(N)}, x_2^{(N)}, \cdots, x_d^{(N)} \end{pmatrix}$

Vector of outputs: $\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}$

Problem: find $\mathbf{w}$ that maximises the likelihood: $\underset{\mathbf{w}}{\operatorname{argmax}}\, \mathscr{L}(\mathbf{w})$

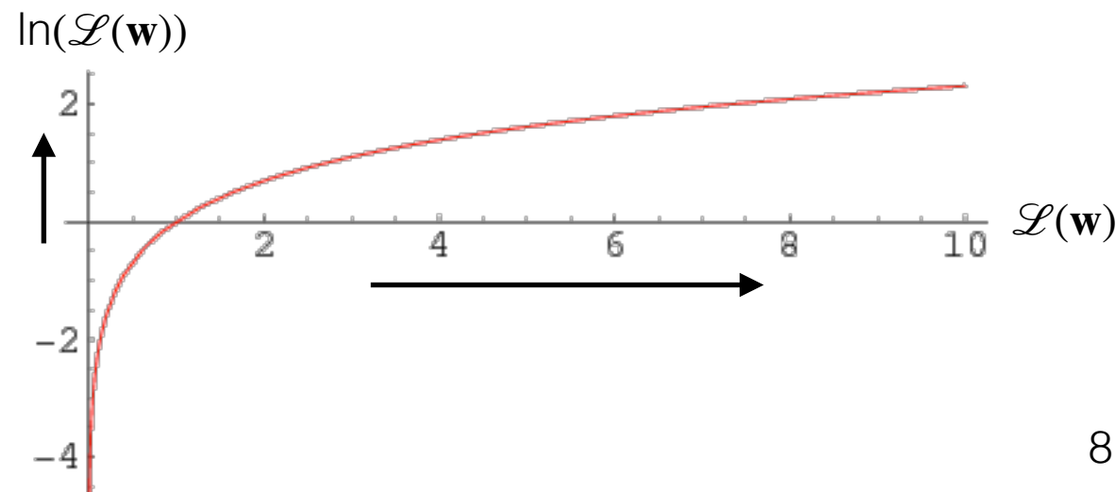
Log-Likelihood

Problem: find $\mathbf{w}$ that maximises the likelihood, $\underset{\mathbf{w}}{\mathbf{argmax}}\, \mathscr{L}(\mathbf{w})$

The products in $\mathscr{L}(\mathbf{w}) = \prod_{i=1}^{N} p_{y^{(i)}}$ can be numerically unstable.

Equivalent to: find $\mathbf{w}$ that maximises the log-likelihood, $\underset{\mathbf{w}}{\mathbf{argmax}}\, \ln(\mathscr{L}(\mathbf{w}))$

$$\ln(\mathscr{L}(\mathbf{w})) = \ln \prod_{i=1}^{N} p_{y^{(i)}} = \sum_{i=1}^{N} \ln\, p_{y^{(i)}}$$



Loss Function

Problem: find $\mathbf{w}$ that maximises the log-likelihood, $\underset{\mathbf{w}}{\mathbf{argmax}}\, \ln(\mathscr{L}(\mathbf{w}))$

$$\ln(\mathscr{L}(\mathbf{w})) = \sum_{i=1}^{N} \ln\, p_{y^{(i)}}$$

Equivalent to: find $\mathbf{w}$ that minimises the loss, $\underset{\mathbf{w}}{\mathbf{argmin}}\, E(\mathbf{w})$

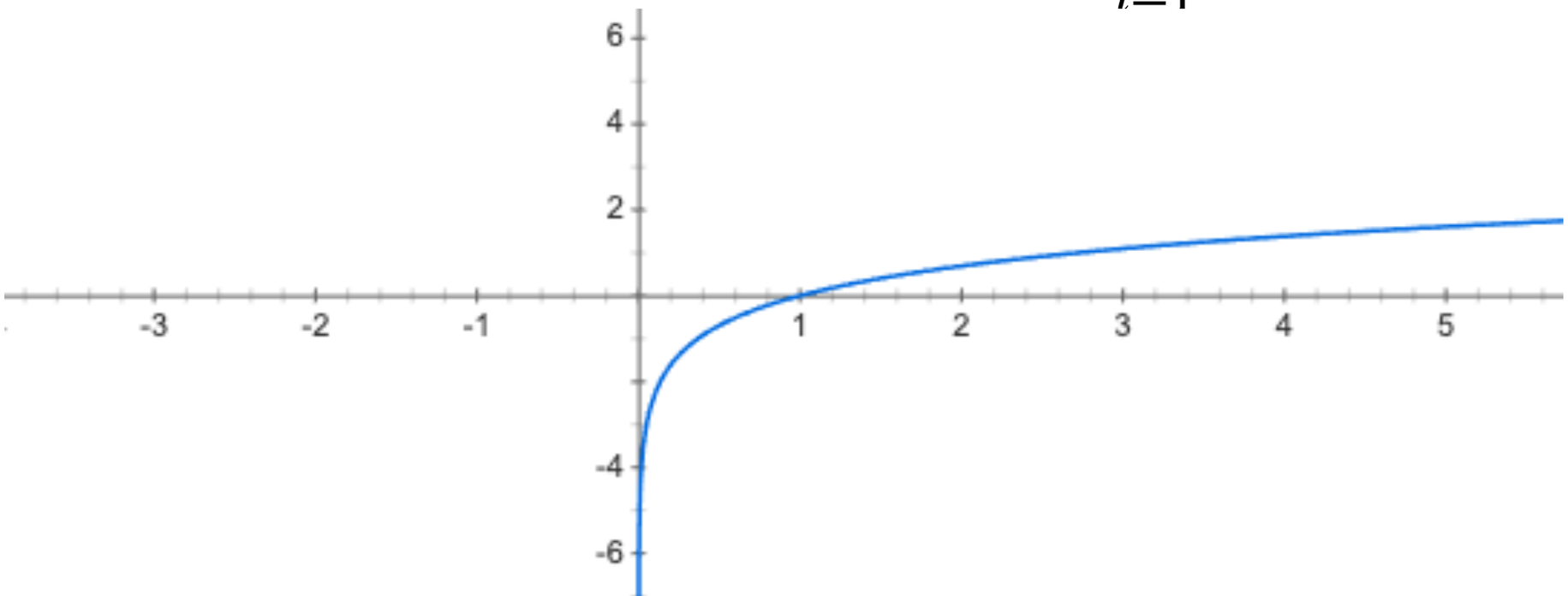
$$E(\mathbf{w}) = -\ln(\mathscr{L}(\mathbf{w})) = -\sum_{i=1}^{N} \ln\, p_{y^{(i)}}$$

Understanding the Loss

Learning $\mathbf{w}$ can be achieved by finding the $\mathbf{w}$ that minimises $E(\mathbf{w})$, calculated based on the training set.

$$\operatorname*{argmin}_{\mathbf{w}} E(\mathbf{w})$$

$$E(\mathbf{w}) = -\ln(\mathscr{L}(\mathbf{w})) = -\sum_{i=1}^{N} \ln p_{y^{(i)}}$$

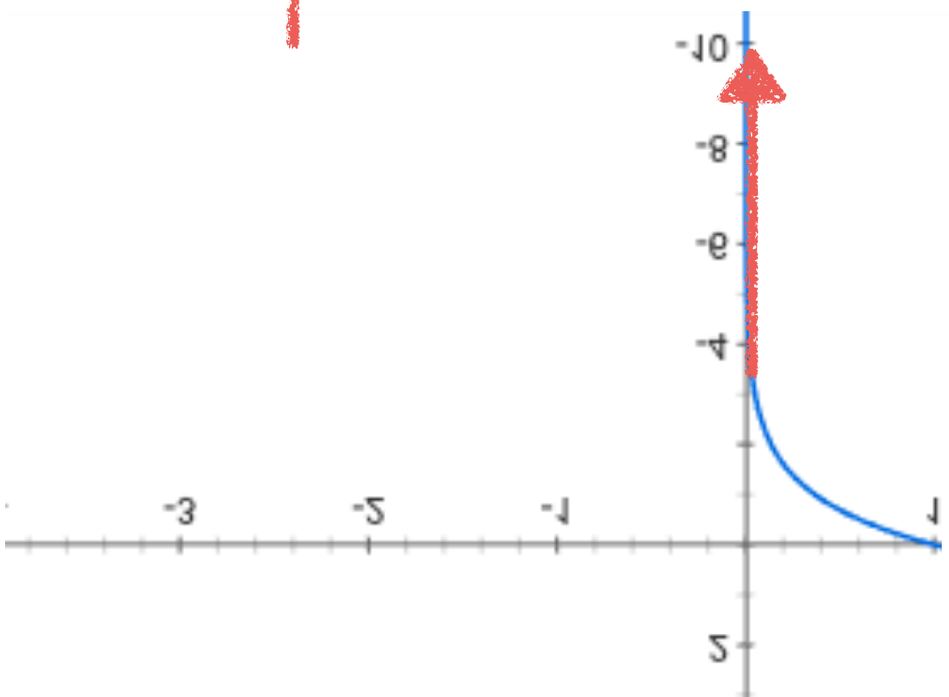


Understanding the Loss

Learning $\mathbf{w}$ can be achieved by finding the $\mathbf{w}$ that minimises $E(\mathbf{w})$, calculated based on the training set.

$$\operatorname{argmin}_{\mathbf{w}} E(\mathbf{w})$$

$$E(\mathbf{w}) = -\ln(\mathscr{L}(\mathbf{w})) = -\sum_{i=1}^{N} \ln p_{y^{(i)}}$$



Assume $(\mathbf{x}^{(i)}, y^{(i)} = 1)$

$p_1 = 0.99 \rightarrow -0.01 \rightarrow +0.01$

$p_1 = 0.5 \rightarrow -0.69 \rightarrow +0.69$

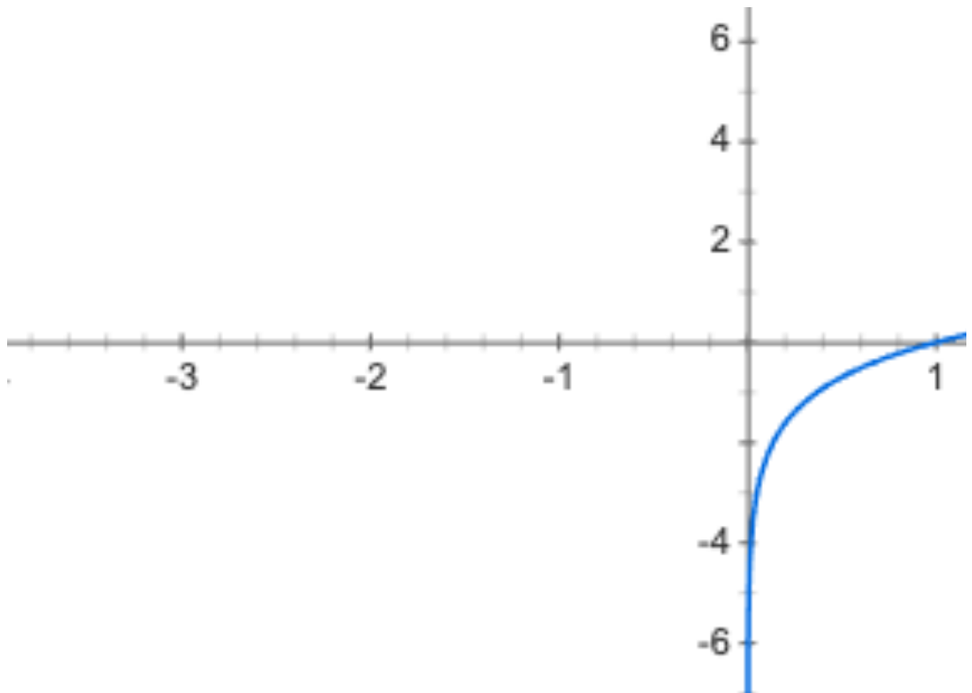
$p_1 = 0.01 \rightarrow -4.6 \rightarrow +4.6$

Understanding the Loss

Learning $\mathbf{w}$ can be achieved by finding the $\mathbf{w}$ that minimises $E(\mathbf{w})$, calculated based on the training set.

$$\operatorname*{argmin}_{\mathbf{w}} E(\mathbf{w})$$

$$E(\mathbf{w}) = -\ln(\mathscr{L}(\mathbf{w})) = -\sum_{i=1}^{N} \ln p_{y^{(i)}}$$



For examples of class 1, we sum $-\ln p_1$.

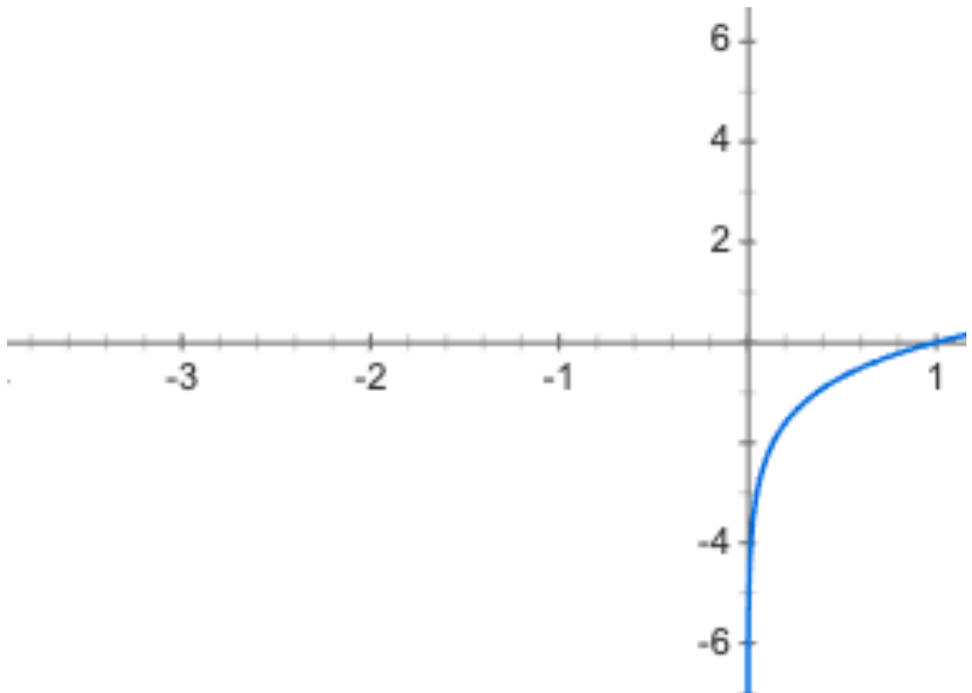
The closer p_1 is to 1, the smaller the value that we sum to $E(\mathbf{w})$, and so the loss is smaller.

Understanding the Loss

Learning $\mathbf{w}$ can be achieved by finding the $\mathbf{w}$ that minimises $E(\mathbf{w})$, calculated based on the training set.

$$\operatorname*{argmin}_{\mathbf{w}} E(\mathbf{w})$$

$$E(\mathbf{w}) = -\ln(\mathscr{L}(\mathbf{w})) = -\sum_{i=1}^{N} \ln p_{y^{(i)}}$$



For examples of class 1, we sum $-\ln p_1$.

The closer p_1 is to 0, the larger the value that we sum to $E(\mathbf{w})$, and we strongly penalise being too close to 0.

Cross Entropy Loss

$$E(\mathbf{w}) = -\ln(\mathscr{L}(\mathbf{w})) = -\sum_{i=1}^{N} \ln\ p_{y^{(i)}}$$

equivalent to:

$$E(\mathbf{w}) = -\sum_{i=1}^{N} y^{(i)}\ \ln\ p(1\,|\,\mathbf{x}^{(i)}, \mathbf{w}) + (1 - y^{(i)})\ \ln\ (1 - p(1\,|\,\mathbf{x}^{(i)}, \mathbf{w}))$$

Cross-entropy is a measure of dissimilarity between two probability distributions.

Here, it is used to measure the dissimilarity between the true (target) distribution $P(y\,|\,\mathbf{x})$ and learned distribution $p(y\,|\,\mathbf{x}, \mathbf{w})$, estimated based on the training examples.

Summary So Far

$$\text{logit}(p_1) = \mathbf{w}^T\mathbf{x} \begin{cases} \mathbf{w}^T\mathbf{x} \geq 0 \rightarrow \text{class } 1 \\ \mathbf{w}^T\mathbf{x} < 0 \rightarrow \text{class } 0 \end{cases}$$

$$g(\mathbf{x}) = p_1 = p(1 \mid \mathbf{x}, \mathbf{w}) \begin{cases} p_1 \geq 0.5 \rightarrow \text{class } 1 \\ p_1 < 0.5 \rightarrow \text{class } 0 \end{cases}$$

Optimisation problem: $\underset{\mathbf{w}}{\text{argmin}}\, E(\mathbf{w})$

$$E(\mathbf{w}) = -\sum_{i=1}^{N} y^{(i)} \ln p(1 \mid \mathbf{x}^{(i)}, \mathbf{w}) + (1 - y^{(i)}) \ln (1 - p(1 \mid \mathbf{x}^{(i)}, \mathbf{w}))$$

How to solve this optimisation problem?

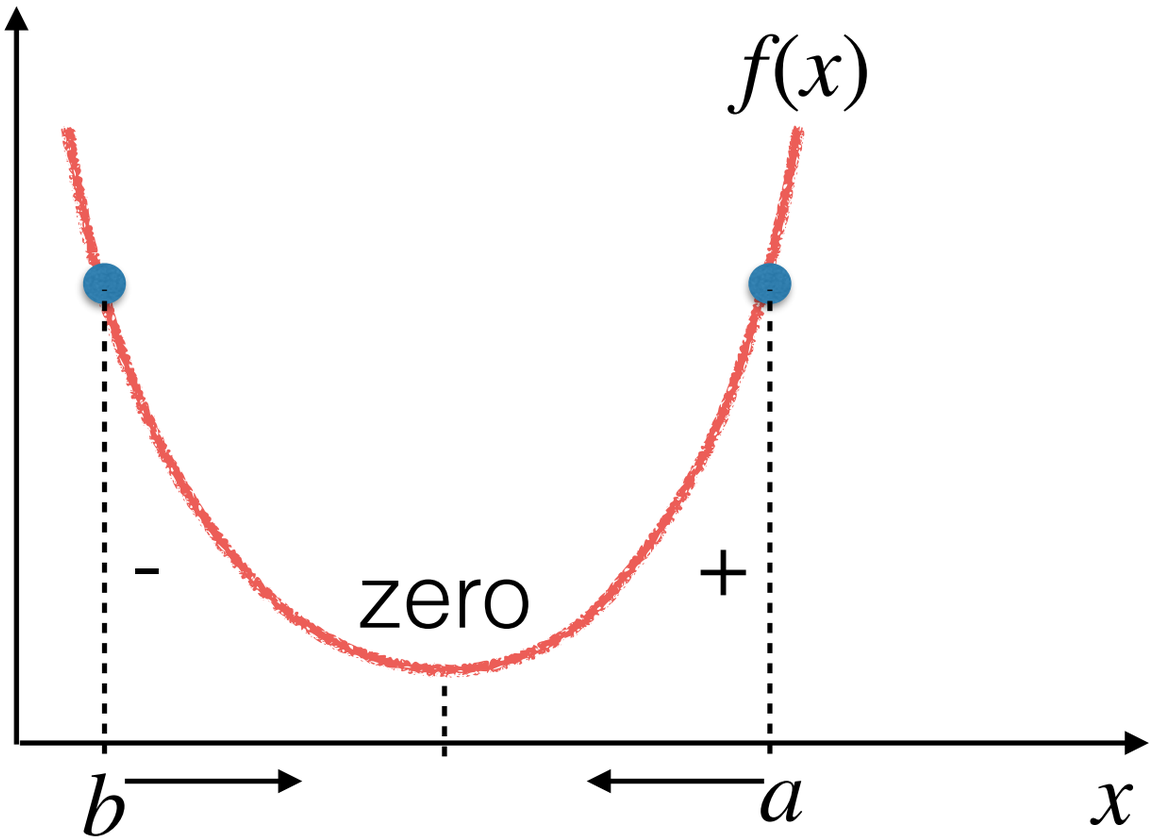
Outline

- Logistic regression optimisation problem
  - What problem needs to be solved to learn a logistic regression model?
  - Maximum likelihood estimation.
- Solving the logistic regression optimisation problem
  - Gradient descent.

General Idea of Gradient Descent

Gradient descent adjusts $\mathbf{w}$ iteratively in the direction that leads to the biggest decrease (steepest descent) in $E(\mathbf{w})$.

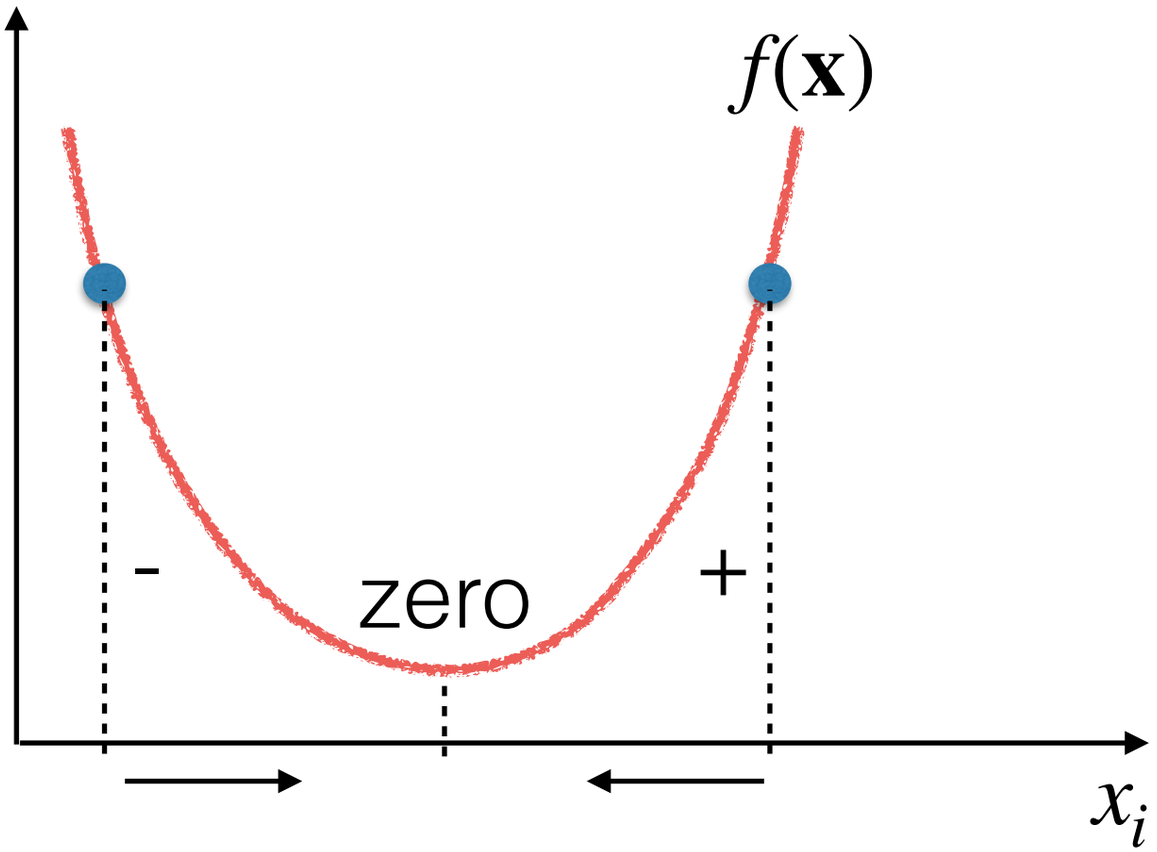
Some Relevant Properties of Derivatives



This can be used to search for the minimum of a function!

$$x = x - \eta \frac{df}{dx} \qquad \text{where } \eta > 0$$

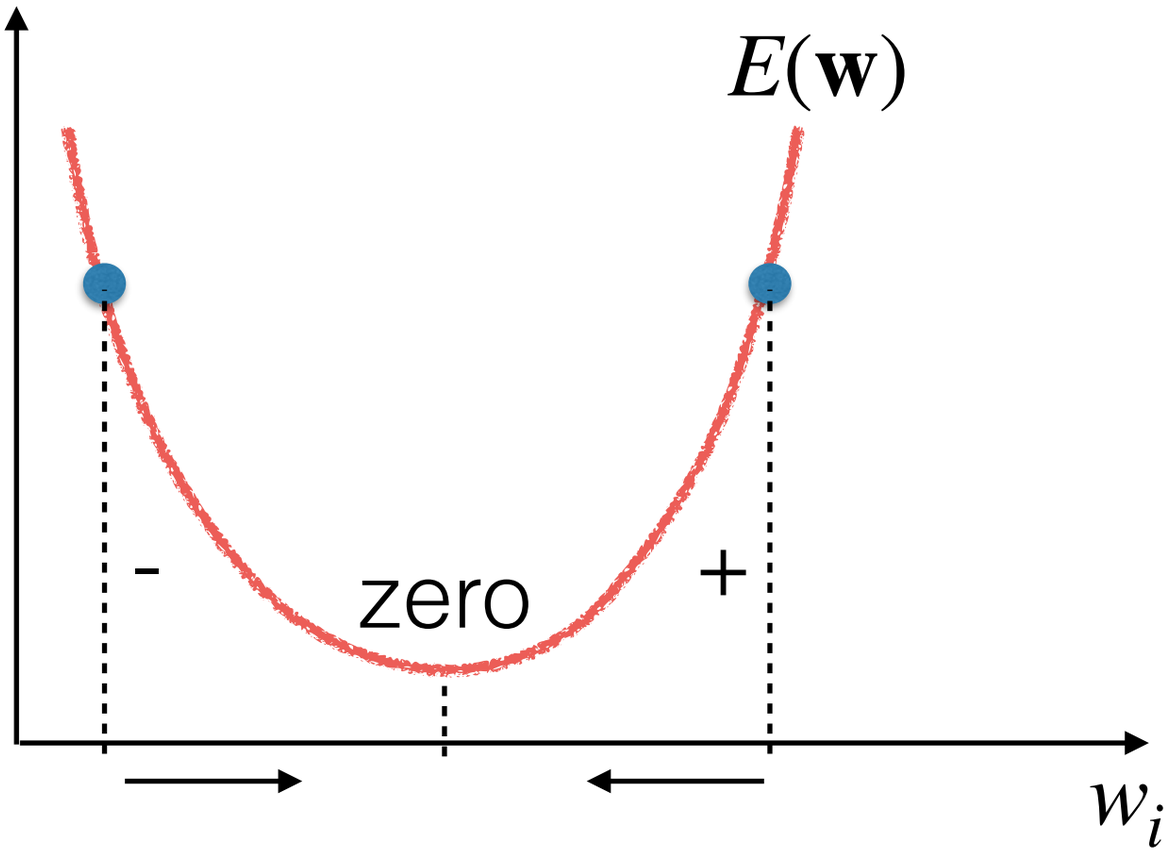
Some Relevant Properties of (Partial) Derivatives



This can be used to search for the minimum of a function!

$$x_i = x_i - \eta \frac{\partial f}{\partial x_i} \quad \text{where } \eta > 0$$

Some Relevant Properties of (Partial) Derivatives



$$w_i = w_i - \eta \frac{\partial E}{\partial w_i} \quad \text{where } \eta > 0$$

This can be used to search for the weights $\mathbf{w}$ that minimise our cross-entropy loss $E(\mathbf{w})$!

Note: The function drawn here is just for illustration purposes. The cross-entropy for logistic regression is not a quadratic function.

Adjusting $\mathbf{w}$ In The Direction that Reduces $E(\mathbf{w})$

$$w_0 = w_0 - \eta \frac{\partial E}{\partial w_0} \qquad w_1 = w_1 - \eta \frac{\partial E}{\partial w_1} \qquad \cdots \qquad w_d = w_d - \eta \frac{\partial E}{\partial w_d}$$

$$\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_d} \end{bmatrix}$$

$$\mathbf{w} = \mathbf{w} - \eta \nabla E(\mathbf{w})$$ "gradient"

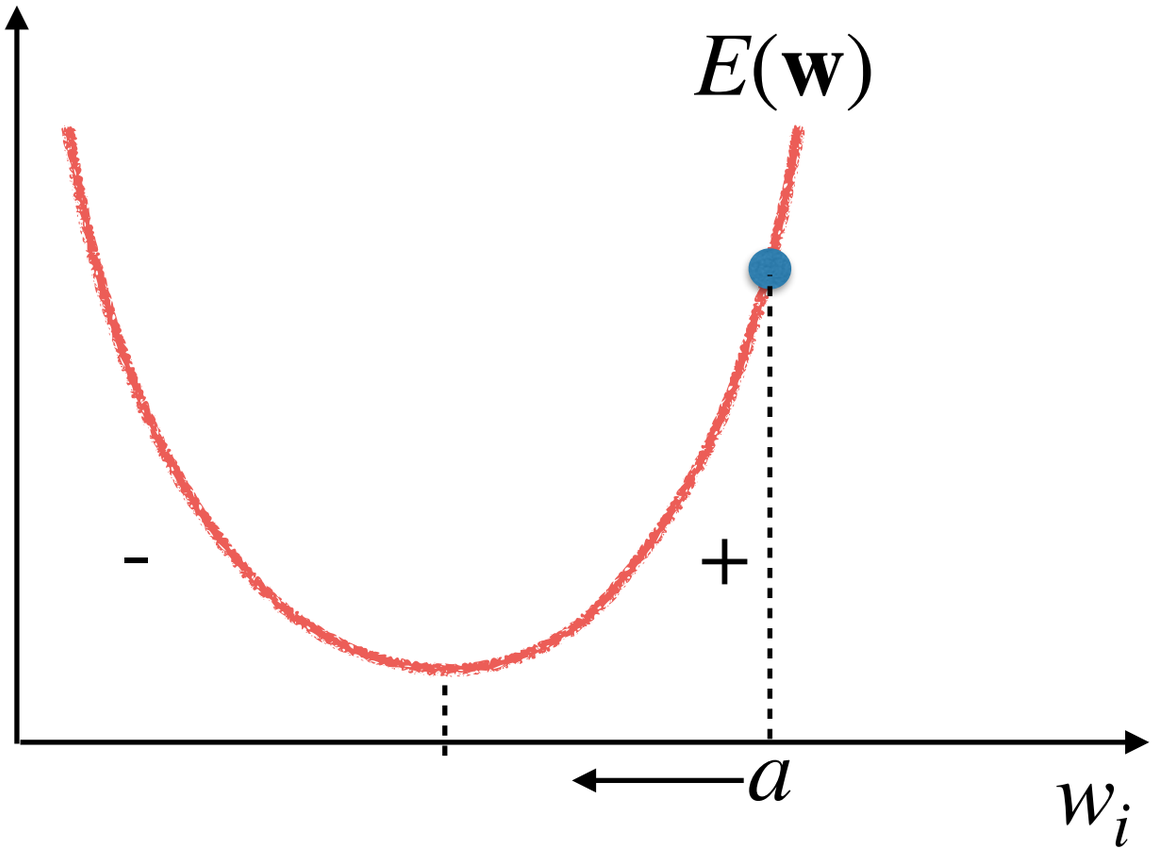
Gradient Descent (Batch Version)

Initialise $\mathbf{w}$ with zeroes or random values near zero.

Repeat for a given number of iterations or until $\nabla E(\mathbf{w})$ is a vector of zeroes:

$$\mathbf{w} = \mathbf{w} - \eta \nabla E(\mathbf{w})$$ where $\eta > 0$ is the learning rate.

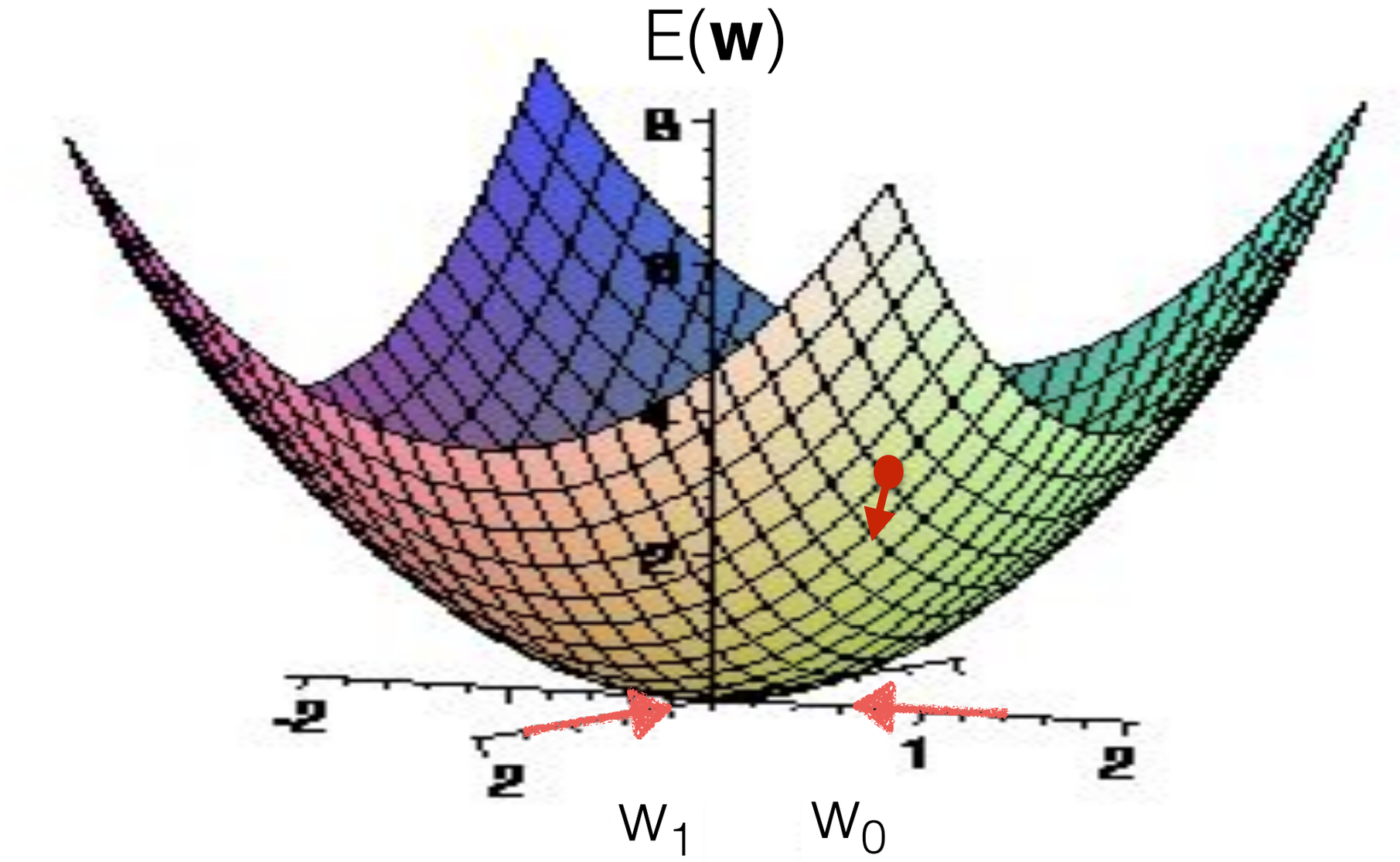
Steepest Descent



$$\mathbf{w} = \mathbf{w} - \eta \nabla E(\mathbf{w})$$

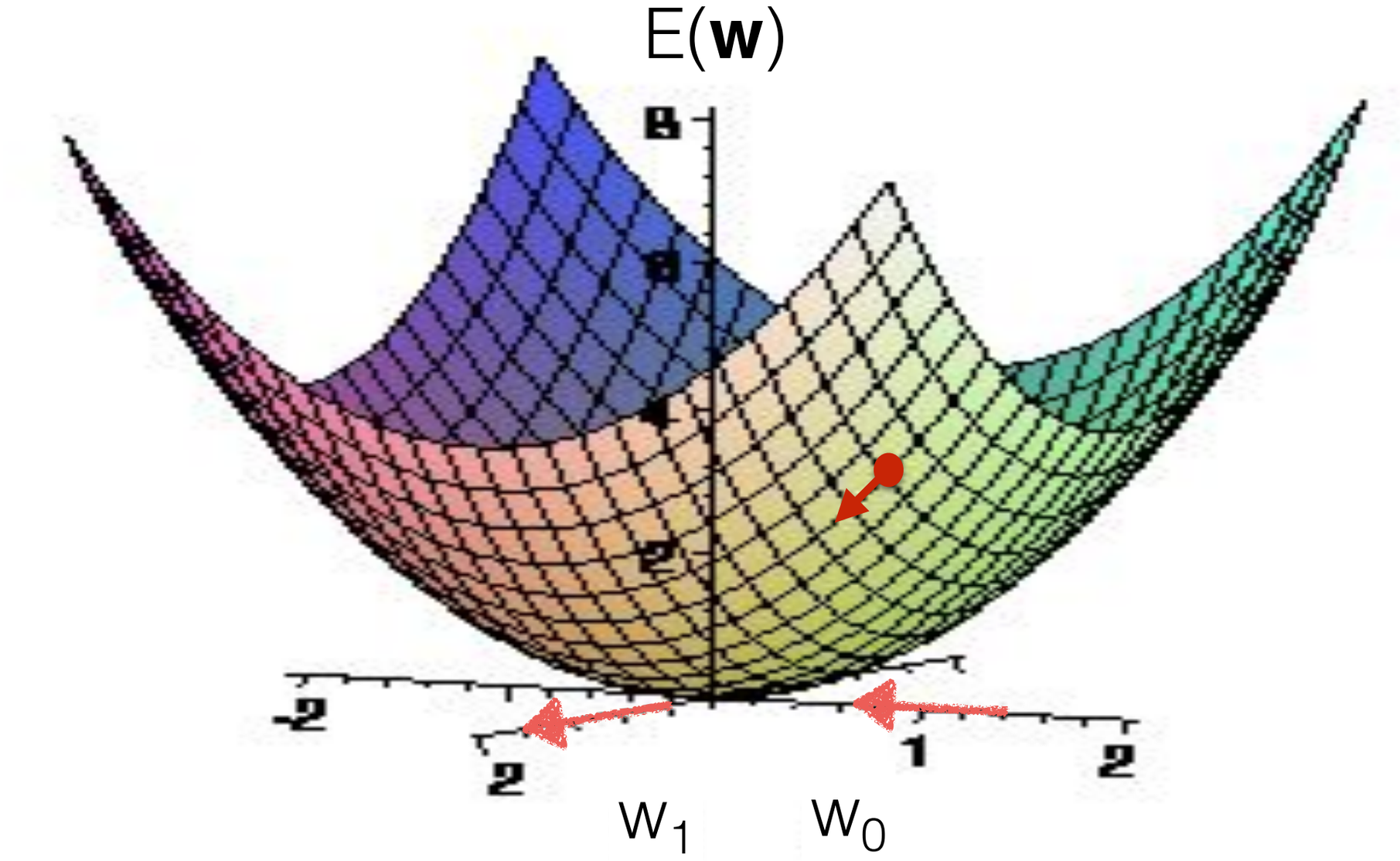
Changes coefficients $\mathbf{w}$ in the direction of the **steepest** descent, i.e., the direction that causes the largest reduction in $E(\mathbf{w})$.

Steepest Descent



Changes coefficients $\mathbf{w}$ in the direction of the **steepest** descent, i.e., the direction that causes the largest reduction in $E(\mathbf{w})$.

Steepest Descent



Changes coefficients $\mathbf{w}$ in the direction of the **steepest** descent, i.e., the direction that causes the largest reduction in $E(\mathbf{w})$.

Applying Gradient Descent (Batch Version) To Logistic Regression

Initialise $\mathbf{w}$ with zeroes or random values near zero.

Repeat for a given number of iterations or until $\nabla E(\mathbf{w})$ is a vector of zeroes:

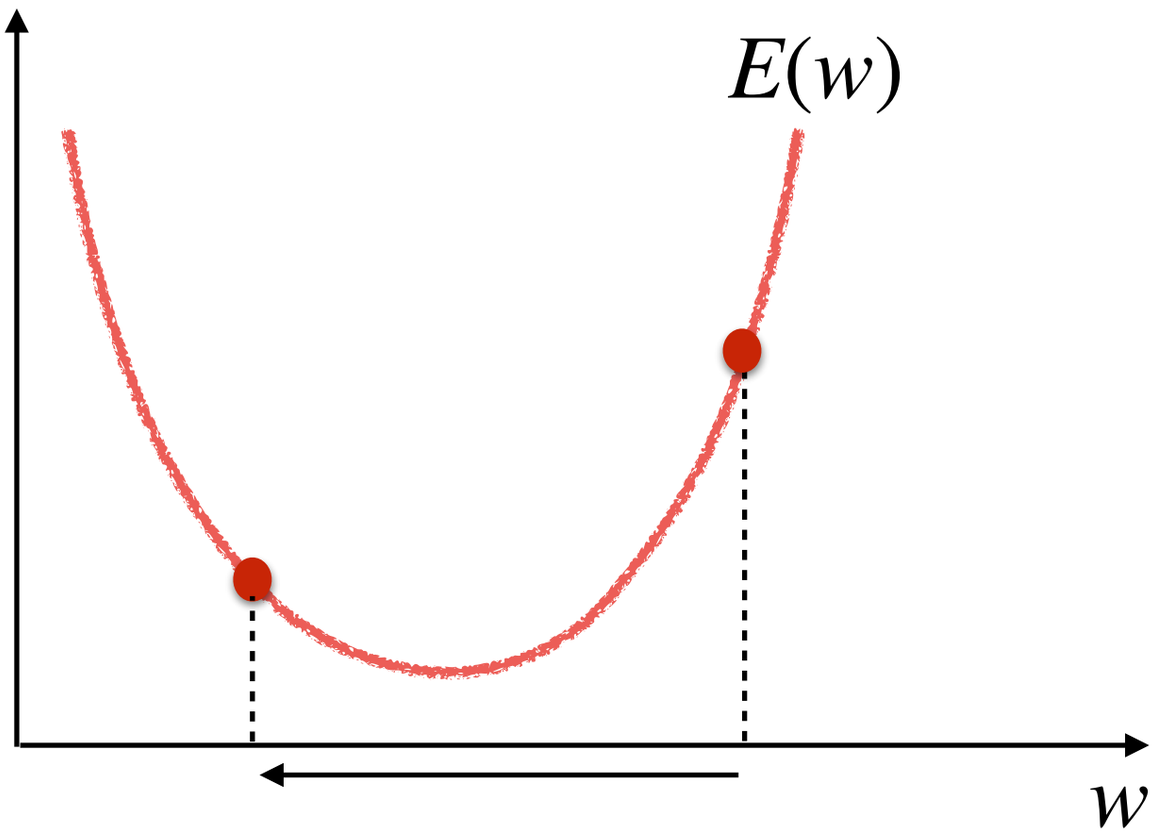
$\mathbf{w} = \mathbf{w} - \eta \nabla E(\mathbf{w})$ where $\eta > 0$ is the learning rate.

$$E(\mathbf{w}) = -\sum_{i=1}^{N} y^{(i)} \ln p(1 \mid \mathbf{x}^{(i)}, \mathbf{w}) + (1 - y^{(i)}) \ln (1 - p(1 \mid \mathbf{x}^{(i)}, \mathbf{w}))$$

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (p(1 \mid \mathbf{x}^{(i)}, \mathbf{w}) - y^{(i)}) \mathbf{x}^{(i)}$$

The Effect of the Hyperparameter η

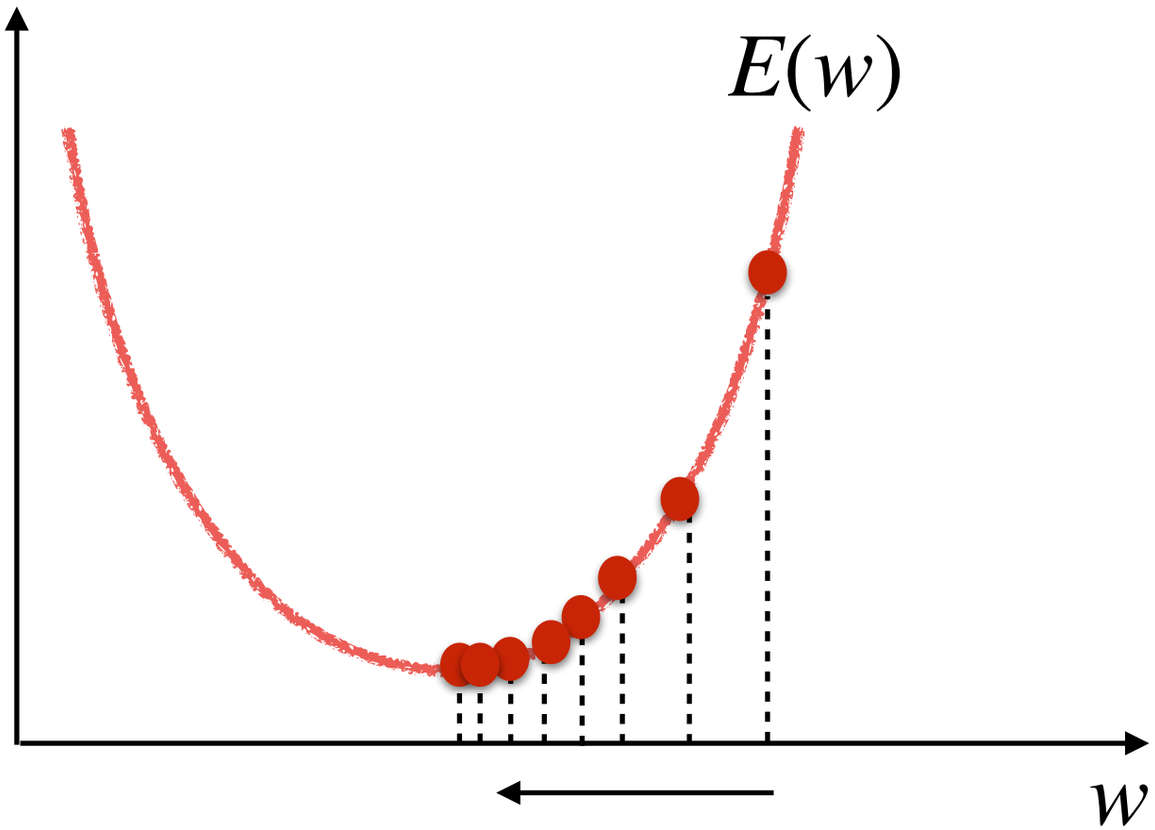
$$\mathbf{w} = \mathbf{w} - \eta \nabla E$$



Too large values of η may result in jumping across the optimum.

The Effect of η

$$\mathbf{w} = \mathbf{w} - \eta \nabla E$$

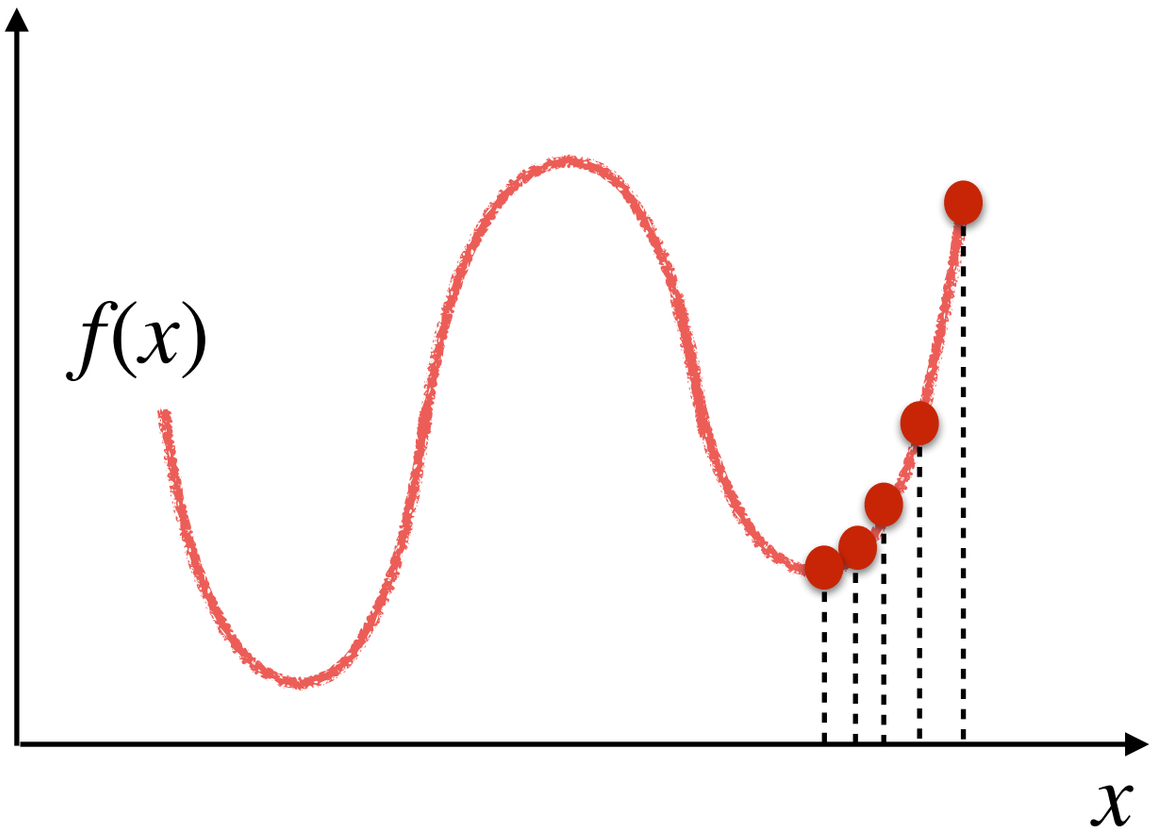


Too small values of η may result in longer time to reach the optimum.

A Note on Local Minima

Gradient descent is a general purpose optimisation algorithm.

But is likely to get stuck in local minima.



For logistic regression using cross-entropy loss, this is not a problem, as its $E(\mathbf{w})$ is strictly convex with respect to $\mathbf{w}$, having a single unique minimum.

Summary

- We can use maximum likelihood estimation to formulate the optimisation problem to be solved for learning a logistic regression model.
- Maximum likelihood estimation is about finding the parameters (coefficients $\mathbf{w}$) that maximise the likelihood of the observed training examples.
- Finding the coefficients $\mathbf{w}$ that maximise the likelihood is equivalent to finding the coefficients $\mathbf{w}$ that minimise the cross-entropy loss.
- We can use Gradient Descent to find good values for $\mathbf{w}$.
- Gradient descent iteratively updates the coefficients $\mathbf{w}$ in the direction of the the steepest descent of $E(\mathbf{w})$, which is the opposite direction of the gradient.
- Next: is Gradient Descent the best optimisation algorithm for us to use?

Further Reading: Loss Function

The reading materials can be found at the module's resource list and elsewhere on the web, except for Iain Style's notes, which can be found in the links provided below.

Essential reading: Abu-Mostafa et al.'s Learning from Data: A Short Course. Section 3.3 (Logistic Regression) until page 92. **==> Note that the authors are using -1 and +1 to represent the different categories, instead of 0 and 1 like in this lecture.**

Recommended reading: Iain Styles's Notes on Logistic Regression:

https://canvas.bham.ac.uk/files/15585285/download?download_frd=1

Optional reading:

Chris Bishop's Pattern Recognition and Machine Learning, Chapter 4.3.2 (Logistic Regression).

Iain Styles's Notes on Multi-class Logistic Regression:

https://canvas.bham.ac.uk/files/15585286/download?download_frd=1

Chris Bishop's Pattern Recognition and Machine Learning, Chapter 4.3.4 (Multiclass Logistic Regression).

Further Reading: Gradient Descent

The reading materials can be found at the module's resource list and elsewhere on the web, except for Leandro Minku's notes on Logistic Regression via Gradient Descent, which are in the links below.

Essential reading: Leandro Minku's Notes on Logistic Regression via Gradient Descent (Read Pages 1 and 2, stopping before the text explaining Online Gradient Descent):

https://canvas.bham.ac.uk/files/15593086/download?download_frd=1

Recommended reading:

Leandro Minku's Notes on Logistic Regression via Gradient Descent (Read Pages 1 to 3, stopping before the text explaining Instantaneous Direction of Best Movement):

https://canvas.bham.ac.uk/files/15593086/download?download_frd=1

Abu-Mostafa et al.'s Learning from Data: A Short Course. Section 3.3.2 (Gradient Descent).

Optional reading:

Chris Bishop's Pattern Recognition and Machine Learning, Chapter 4.3.2 (Logistic Regression).