

# Teoria degli Algoritmi

Corso di Laurea Magistrale in Matematica Applicata

a.a. 2020-21



**SAPIENZA**  
UNIVERSITÀ DI ROMA

Gabriele Tolomei

Dipartimento di Informatica

Sapienza Università di Roma

[tolomei@di.uniroma1.it](mailto:tolomei@di.uniroma1.it)

# Recap from Last Lectures

- We described linear regression as a powerful technique to predict real-valued function
- Linear regression tries to fit a straight hyperplane between features (i.e., independent variables) and the target (i.e., dependent variable)
- OLS method to easily estimate the parameters of the model
- More advanced techniques may be applied if the relationship between features and the target is not linear (e.g., polynomial regression)

# Classification vs. Regression

- Very often, the response variable to predict is **qualitative** (categorical)

# Classification vs. Regression

- Very often, the response variable to predict is **qualitative** (categorical)
- **Classification** (as opposed to regression) deals with predicting categorical responses

# Classification vs. Regression

- Very often, the response variable to predict is **qualitative** (categorical)
- **Classification** (as opposed to regression) deals with predicting categorical responses
- Examples:
  - spam vs. non-spam emails
  - click vs. non-click on a web page or an advertisement

# Classification vs. Regression

- Very often, the response variable to predict is **qualitative** (categorical)
- **Classification** (as opposed to regression) deals with predicting categorical responses
- Examples:
  - spam vs. non-spam emails
  - click vs. non-click on a web page or an advertisement
- Classification methods may first predict the probability of each category of a qualitative response to make in turn a decision

# Why Not Linear Regression, Then?

- Suppose we want to predict the health condition of a patient arriving in the ER on the basis of her symptoms

# Why Not Linear Regression, Then?

- Suppose we want to predict the health condition of a patient arriving in the ER on the basis of her symptoms
- Imagine there are only the following 3 possible diagnoses: stroke, drug overdose, and epileptic seizure



# Why Not Linear Regression, Then?

- Suppose we want to predict the health condition of a patient arriving in the ER on the basis of her symptoms
- Imagine there are only the following 3 possible diagnoses: **stroke**, **drug overdose**, and **epileptic seizure**
- We may encode the above values as a categorical response variable  $Y$

$$Y = \begin{cases} 1 & \text{if } \mathbf{stroke}; \\ 2 & \text{if } \mathbf{drug\ overdose}; \\ 3 & \text{if } \mathbf{epileptic\ seizure}. \end{cases}$$

# Why Not Linear Regression, Then?

- With the previous encoding we can fit a linear regression model using OLS from a set of  $n$  features  $x_1, \dots, x_n$

# Why Not Linear Regression, Then?

- With the previous encoding we can fit a linear regression model using OLS from a set of  $n$  features  $x_1, \dots, x_n$
- Unfortunately, this coding implies an **ordering** on the outcomes
  - drug overdose is in between stroke and epileptic seizure
  - the difference between stroke and drug overdose is the same as the difference between drug overdose and epileptic seizure

# Why Not Linear Regression, Then?

- With the previous encoding we can fit a linear regression model using OLS from a set of  $n$  features  $x_1, \dots, x_n$
- Unfortunately, this coding implies an **ordering** on the outcomes
  - drug overdose is in between stroke and epileptic seizure
  - the difference between stroke and drug overdose is the same as the difference between drug overdose and epileptic seizure
- In practice, there is no particular reason to choose the encoding above!

# Why Not Linear Regression, Then?

- With the previous encoding we can fit a linear regression model using OLS from a set of  $n$  features  $x_1, \dots, x_n$
- Unfortunately, this coding implies an **ordering** on the outcomes
  - drug overdose is in between stroke and epileptic seizure
  - the difference between stroke and drug overdose is the same as the difference between drug overdose and epileptic seizure
- In practice, there is no particular reason to choose the encoding above!
- Different (and still legitimate) encodings will produce different models

# Why Not Linear Regression, Then?

- The encoding above would work if the response variable values take on an **equally-spaced natural ordering** (e.g., mild, moderate, and severe)

# Why Not Linear Regression, Then?

- The encoding above would work if the response variable values take on an **equally-spaced natural ordering** (e.g., mild, moderate, and severe)
- In general, there is no natural way to convert a K-ary ( $K > 2$ ) response into a quantitative response that is ready for linear regression

# Why Not Linear Regression, Then?

- The encoding above would work if the response variable values take on an **equally-spaced natural ordering** (e.g., mild, moderate, and severe)
- In general, there is no natural way to convert a K-ary ( $K > 2$ ) response into a quantitative response that is ready for linear regression
- For a binary response with a 0/1 encoding, linear regression by OLS does anyway make sense
  - Predict 1 if the outcome is  $> 0.5$ , 0 otherwise



# Why Not Linear Regression, Then?

- The encoding above would work if the response variable values take on an **equally-spaced natural ordering** (e.g., mild, moderate, and severe)
- In general, there is no natural way to convert a K-ary ( $K > 2$ ) response into a quantitative response that is ready for linear regression
- For a binary response with a 0/1 encoding, linear regression by OLS does anyway make sense
  - Predict 1 if the outcome is  $> 0.5$ , 0 otherwise
- Still, it is preferable to use a classification method which works by design

# LOGISTIC REGRESSION

## Example: Default(Y) vs. Balance(X)

Consider a binary response Default(Y) taking on two values: Yes or No

## Example: Default(Y) vs. Balance(X)

Consider a binary response Default(Y) taking on two values: Yes or No

Suppose we want to predict the value of Y from the value of Balance(X)

## Example: Default(Y) vs. Balance(X)

Consider a binary response Default(Y) taking on two values: Yes or No

Suppose we want to predict the value of Y from the value of Balance(X)

We can model it directly via linear regression (i.e., predicting its value)

## Example: Default(Y) vs. Balance(X)

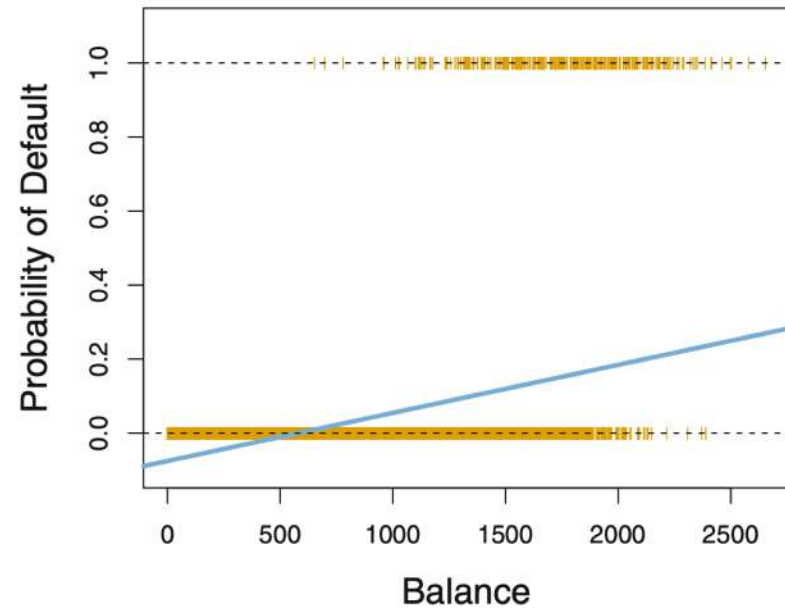
Consider a binary response Default(Y) taking on two values: Yes or No

Suppose we want to predict the value of Y from the value of Balance(X)

We can model it **directly** via linear regression (i.e., predicting its value)

**Logistic Regression** instead models the **probability** that Y belongs to one of the two possible outcome values

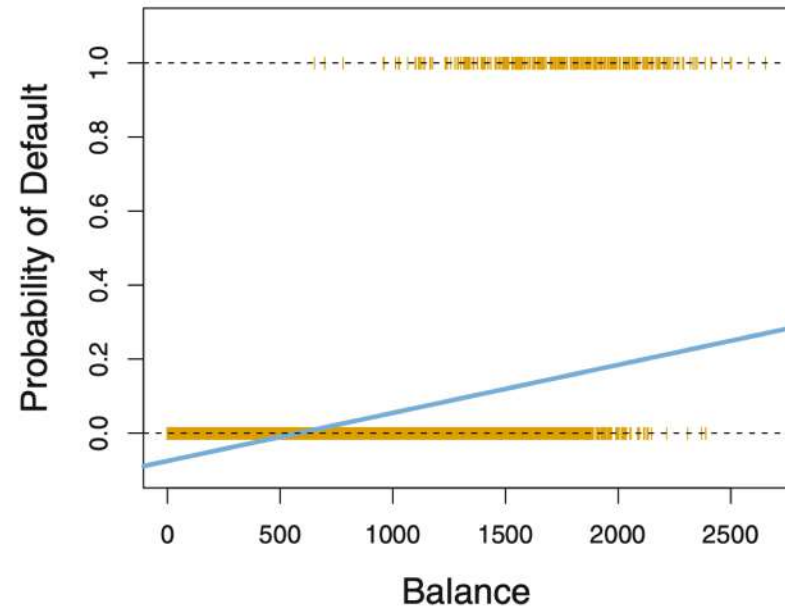
# Example: Default(Y) vs. Balance(X)



Predicted probability using **linear regression**  
(some estimated probabilities are negative!)

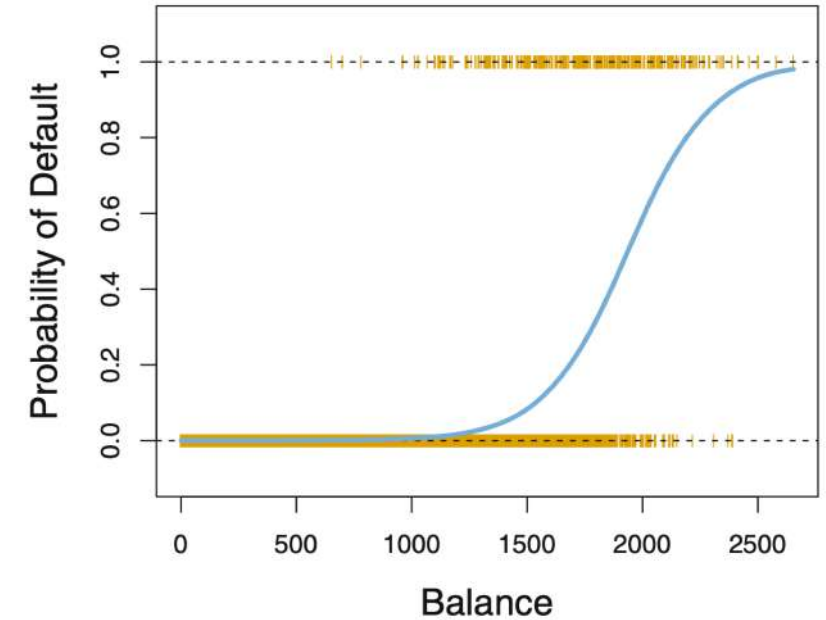
**Linear Regression**

# Example: Default(Y) vs. Balance(X)



Predicted probability using **linear regression**  
(some estimated probabilities are negative!)

Linear Regression



Predicted probability using **logistic regression**  
(all probabilities lie between 0 and 1)

Logistic Regression



# General Concepts

3 components need to be specified

# General Concepts

3 components need to be specified

Model

Defines the space of  
representable hypotheses

# General Concepts

3 components need to be specified

## Model

Defines the space of representable hypotheses

## Error Measure (Cost Function)

Measures the price of misclassification errors

# General Concepts

3 components need to be specified

```
graph TD; A[3 components need to be specified] -- blue arrow --> B[Model]; A -- green arrow --> C[Error Measure]; A -- orange arrow --> D[Learning Algorithm];
```

## Model

Defines the space of representable hypotheses

## Error Measure (Cost Function)

Measures the price of misclassification errors

## Learning Algorithm

Picks the best hypothesis exploring search space

MODEL

# Linear Signal

- Let  $\mathbf{x} = (x_0, x_1, \dots, x_d)$  be the  $(d+1)$ -dimensional input

# Linear Signal

- Let  $\mathbf{x} = (x_0, x_1, \dots, x_d)$  be the  $(d+1)$ -dimensional input
- Let  $F$  be the family of real-valued functions parametrized by  $\boldsymbol{\theta}$  so that  $\boldsymbol{\theta}^T = (\theta_0, \theta_1, \dots, \theta_d)$

$$\mathcal{F} = \{f_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \mapsto \mathbb{R} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \sum_{i=0}^d \theta_i x_i\}$$

# Linear Signal

- Let  $\mathbf{x} = (x_0, x_1, \dots, x_d)$  be the  $(d+1)$ -dimensional input
- Let  $F$  be the family of real-valued functions parametrized by  $\boldsymbol{\theta}$  so that  $\boldsymbol{\theta}^T = (\theta_0, \theta_1, \dots, \theta_d)$

$$\mathcal{F} = \{f_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \mapsto \mathbb{R} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \sum_{i=0}^d \theta_i x_i\}$$

- Each function in  $F$  outputs a real number (i.e., a scalar) as a linear combination of the input  $\mathbf{x}$  with the parameters  $\boldsymbol{\theta}$



# Linear Signal

- Let  $\mathbf{x} = (x_0, x_1, \dots, x_d)$  be the  $(d+1)$ -dimensional input
- Let  $F$  be the family of real-valued functions parametrized by  $\boldsymbol{\theta}$  so that  $\boldsymbol{\theta}^T = (\theta_0, \theta_1, \dots, \theta_d)$

$$\mathcal{F} = \{f_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \mapsto \mathbb{R} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \sum_{i=0}^d \theta_i x_i\}$$

- Each function in  $F$  outputs a real number (i.e., a scalar) as a linear combination of the input  $\mathbf{x}$  with the parameters  $\boldsymbol{\theta}$
- $f_{\boldsymbol{\theta}}(\mathbf{x})$  is referred to as (**linear**) **signal**

# Hypothesis Space (Revisited)

- The signal alone is not enough to define the hypothesis space  $H$

# Hypothesis Space (Revisited)

- The signal alone is not enough to define the hypothesis space  $H$
- Usually the signal is passed through a "filter", i.e. another real-valued function  $g$

# Hypothesis Space (Revisited)

- The signal alone is not enough to define the hypothesis space  $H$
- Usually the signal is passed through a "filter", i.e. another real-valued function  $g$
- $h_{\boldsymbol{\theta}}(\mathbf{x}) = g(f_{\boldsymbol{\theta}}(\mathbf{x}))$  defines the hypothesis space:

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \mapsto \mathbb{R} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = g(f_{\boldsymbol{\theta}}(\mathbf{x})) = g(\boldsymbol{\theta}^T \mathbf{x}) = g(\sum_{i=0}^d \theta_i x_i)\}$$

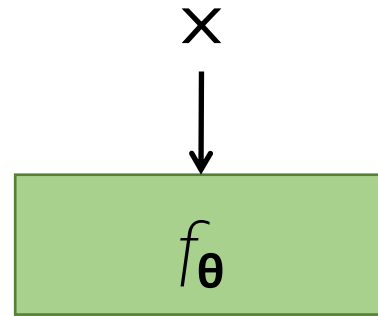
# Hypothesis Space (Revisited)

- The signal alone is not enough to define the hypothesis space  $H$
- Usually the signal is passed through a "filter", i.e. another real-valued function  $g$
- $h_{\boldsymbol{\theta}}(\mathbf{x}) = g(f_{\boldsymbol{\theta}}(\mathbf{x}))$  defines the hypothesis space:

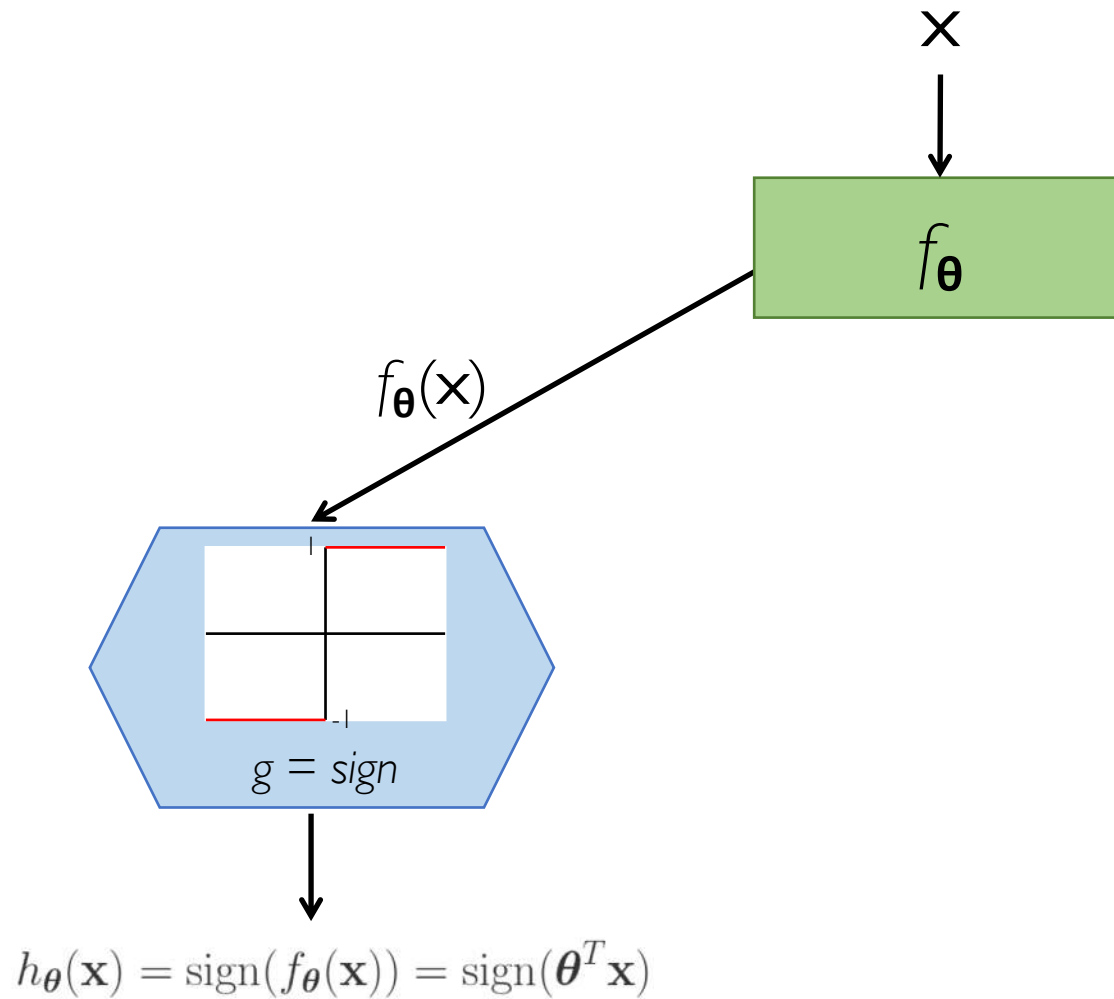
$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \mapsto \mathbb{R} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = g(f_{\boldsymbol{\theta}}(\mathbf{x})) = g(\boldsymbol{\theta}^T \mathbf{x}) = g(\sum_{i=0}^d \theta_i x_i)\}$$

The set of possible hypotheses  $H$  changes depending on the parametric model ( $f_{\boldsymbol{\theta}}$ ) and on the **thresholding function** ( $g$ )

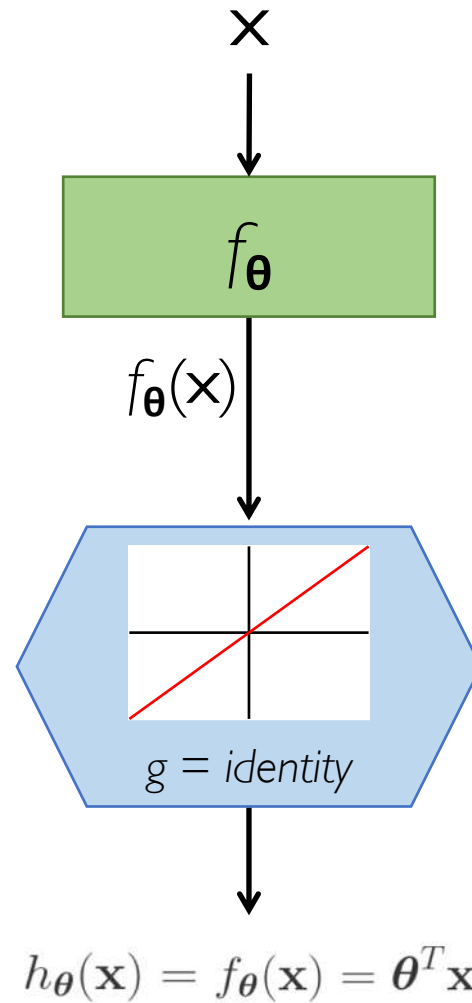
# Filtering (Thresholding)



# Filtering (Thresholding)

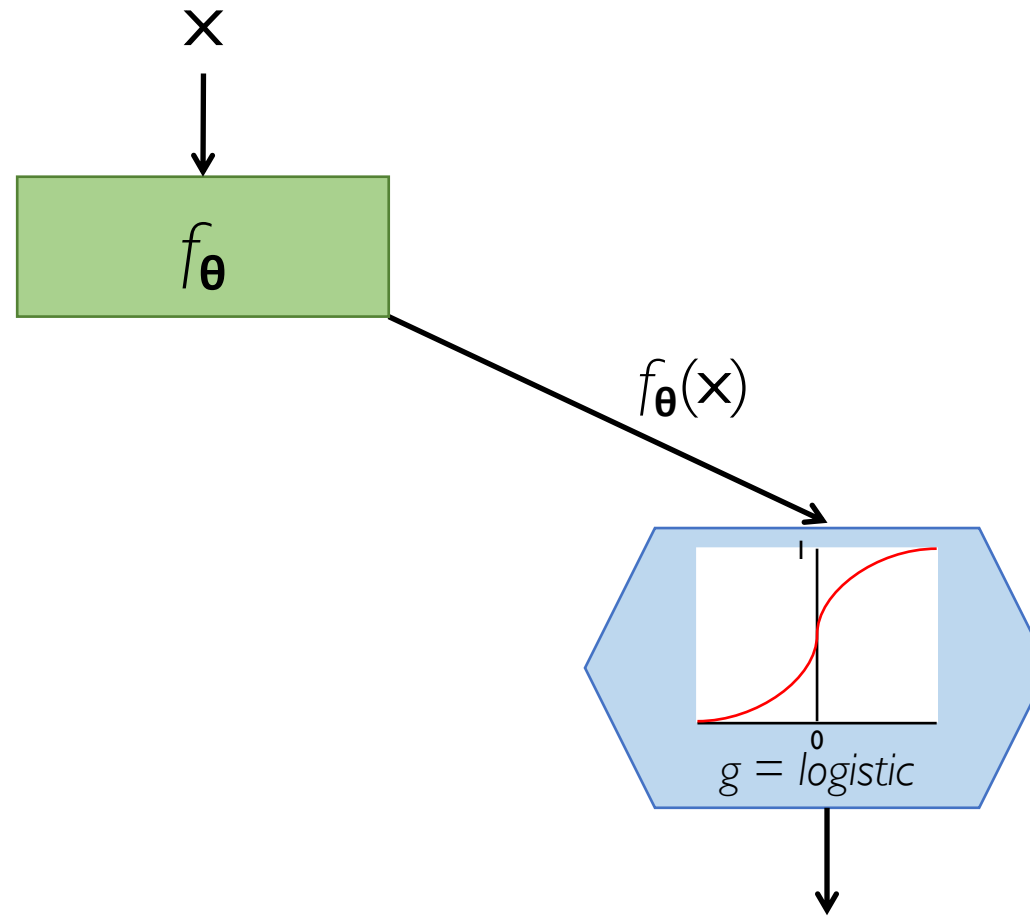


# Filtering (Thresholding)



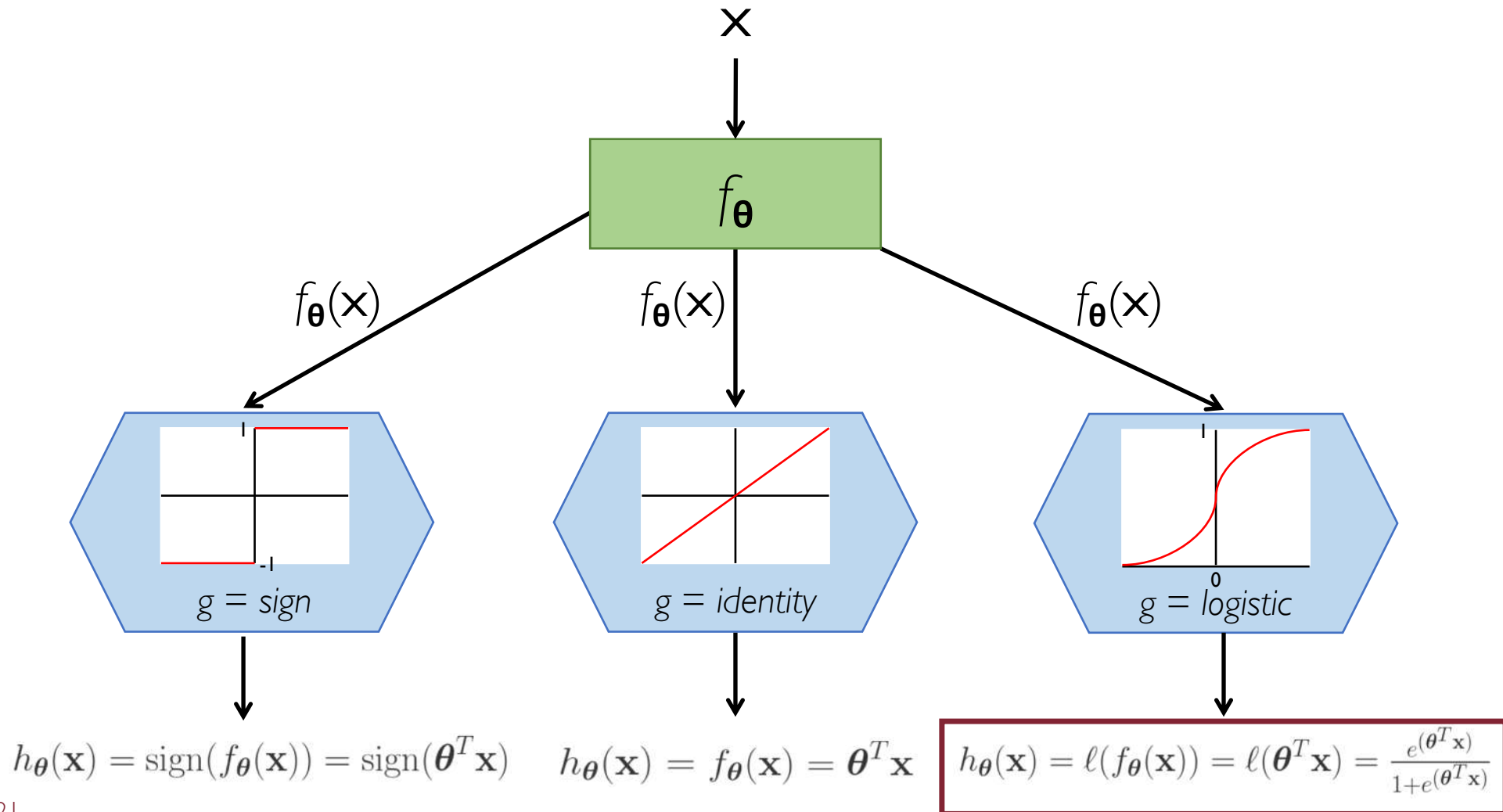


# Filtering (Thresholding)

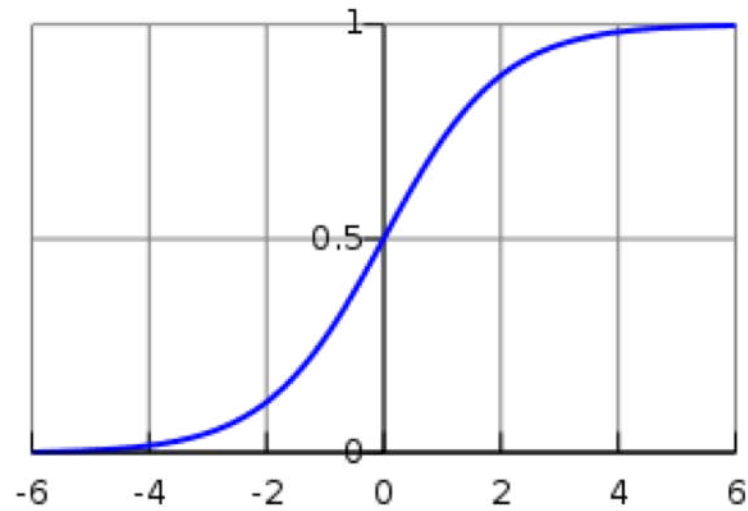


$$h_{\theta}(\mathbf{x}) = \ell(f_{\theta}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{\boldsymbol{\theta}^T \mathbf{x}}}{1 + e^{\boldsymbol{\theta}^T \mathbf{x}}}$$

# Filtering (Thresholding)

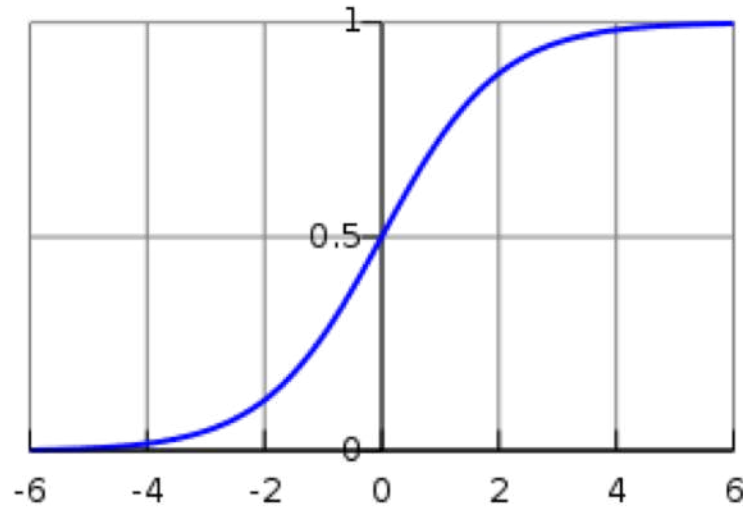


# The Logistic Function



$$l(z) = \frac{e^z}{1+e^z}$$

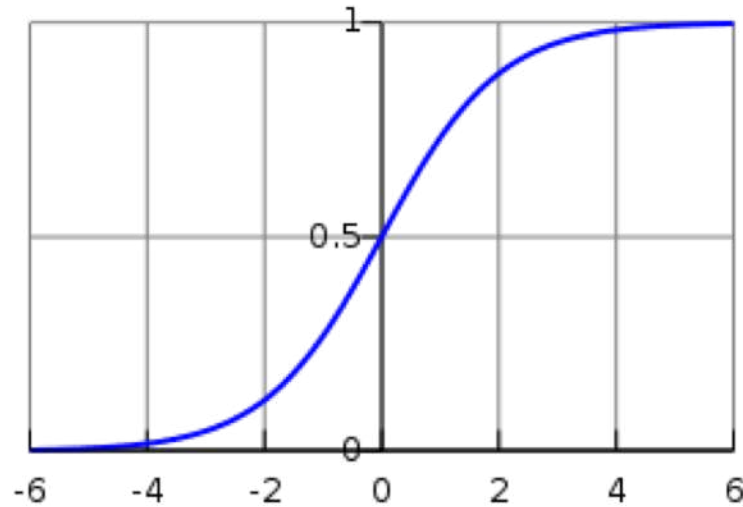
# The Logistic Function



$$\ell(z) = \frac{e^z}{1+e^z}$$

- Also known as sigmoid function do to its "S" shape or soft threshold (compared to hard threshold imposed by *sign*)

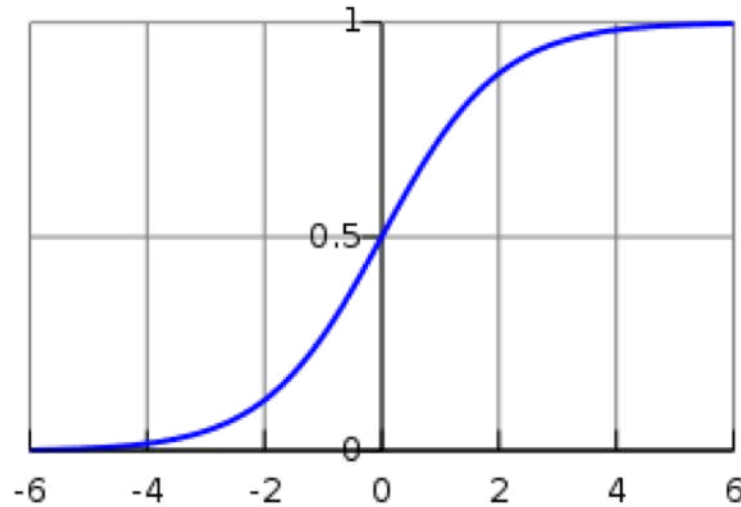
# The Logistic Function



$$\ell(z) = \frac{e^z}{1+e^z}$$

- Also known as sigmoid function do to its "S" shape or soft threshold (compared to hard threshold imposed by *sign*)
- When  $z = \boldsymbol{\theta}^T \mathbf{x}$  we are applying a *non-linear* transformation to our *linear* signal

# The Logistic Function



$$\ell(z) = \frac{e^z}{1+e^z}$$

- Also known as sigmoid function do to its "S" shape or soft threshold (compared to hard threshold imposed by *sign*)
- When  $z = \boldsymbol{\theta}^T \mathbf{x}$  we are applying a *non-linear* transformation to our *linear* signal
- Output can be *genuinely* interpreted as a probability value

# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

- Describing the set of hypotheses using the **logistic function** is not enough to state that the output can be interpreted as a probability



# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

- Describing the set of hypotheses using the **logistic function** is not enough to state that the output can be interpreted as a probability
- All we know is that the logistic function always produce a real value between 0 and 1

# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

- Describing the set of hypotheses using the **logistic function** is not enough to state that the output can be interpreted as a probability
- All we know is that the logistic function always produce a real value between 0 and 1
- Other functions may have the same property [e.g.,  $1/\pi \arctan(x) + 1/2$ ]

# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

- The key points here are:

# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

- The key points here are:
  - the output of the logistic function can be interpreted as a probability even during learning

# Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T \mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}}$$

- The key points here are:
  - the output of the logistic function can be interpreted as a probability even during learning
  - the logistic function is mathematically convenient!

## Additional Notes

[https://github.com/gtolomei/theory-of-algorithms/raw/main/extras/Notes\\_on\\_Logistic\\_Regression.pdf](https://github.com/gtolomei/theory-of-algorithms/raw/main/extras/Notes_on_Logistic_Regression.pdf)

# Odds

- Let  $p$  (resp.,  $q = 1-p$ ) be the probability of success (resp., failure) of an event

# Odds

- Let  $p$  (resp.,  $q = 1-p$ ) be the probability of success (resp., failure) of an event
- $\text{odds}(\text{success}) = p/q = p/(1-p)$

# Odds

- Let  $p$  (resp.,  $q = 1-p$ ) be the probability of success (resp., failure) of an event
- $\text{odds}(\text{success}) = p/q = p/(1-p)$
- $\text{odds}(\text{failure}) = q/p = 1/p/q = 1/\text{odds}(\text{success})$



# Odds

- Let  $p$  (resp.,  $q = 1-p$ ) be the probability of success (resp., failure) of an event
- $\text{odds}(\text{success}) = p/q = p/(1-p)$
- $\text{odds}(\text{failure}) = q/p = 1/p/q = 1/\text{odds}(\text{success})$
- $\text{logit}(p) = \ln(\text{odds}(\text{success})) = \ln(p/q) = \ln(p/1-p) = \ln(p) - \ln(1-p)$

# Odds

Logistic Regression is in fact an ordinary linear regression where the logit is the response variable!

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

The coefficients of logistic regression are expressed in terms of the natural logarithm of odds

# Why Odds instead of Probability?

Odds are defined on the range  $[0, +\infty]$

# Why Odds instead of Probability?

Odds are defined on the range  $[0, +\infty]$

Logit (i.e., natural log of odds) are defined on the range  $[-\infty, +\infty]$

# Why Odds instead of Probability?

Odds are defined on the range  $[0, +\infty]$

Logit (i.e., natural log of odds) are defined on the range  $[-\infty, +\infty]$

Therefore we can use "standard" regression equation:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

# Why Odds instead of Probability?

Odds are defined on the range  $[0, +\infty]$

Logit (i.e., natural log of odds) are defined on the range  $[-\infty, +\infty]$

Therefore we can use "standard" regression equation:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

For any value of the regression coefficients and features a valid value for the odds are predicted

# Why Odds instead of Probability?

Odds are defined on the range  $[0, +\infty]$

Logit (i.e., natural log of odds) are defined on the range  $[-\infty, +\infty]$

Therefore we can use "standard" regression equation:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

For any value of the regression coefficients and features a valid value for the odds are predicted

Probabilities are only defined on the range  $[0, 1]$

It would need very complicated constraints on the regression coefficients to work with probability

# From Odds to Probability

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$



# From Odds to Probability

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

$$e^{\text{logit}(p)} = e^{\ln\left(\frac{p}{1-p}\right)} = \frac{p}{1-p} = e^{(\boldsymbol{\theta}^T \mathbf{x})}$$

$$p = e^{(\boldsymbol{\theta}^T \mathbf{x})}(1 - p) = e^{(\boldsymbol{\theta}^T \mathbf{x})} - e^{(\boldsymbol{\theta}^T \mathbf{x})}p$$

$$p + e^{(\boldsymbol{\theta}^T \mathbf{x})}p = e^{(\boldsymbol{\theta}^T \mathbf{x})}$$

$$p(1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}) = e^{(\boldsymbol{\theta}^T \mathbf{x})}$$

$$p = \frac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}} = \frac{1}{e^{-(\boldsymbol{\theta}^T \mathbf{x})} + 1}$$

# Odds Ratio

Using (log) odds rather than actual probabilities provides an easier interpretation of the model's coefficients learned

$$\ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

$$\left(\frac{p}{1-p}\right) = e^{\theta_0 + \theta_1 x_1 + \dots + \theta_d x_d} = e^{\boldsymbol{\theta}^T \mathbf{x}}$$

# Odds Ratio

Using (log) odds rather than actual probabilities provides an easier interpretation of the model's coefficients learned

$$\ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

$$\left(\frac{p}{1-p}\right) = e^{\theta_0 + \theta_1 x_1 + \dots + \theta_d x_d} = e^{\boldsymbol{\theta}^T \mathbf{x}}$$

Suppose we want to measure the effect of a unit increase in one of the predictors to the output response

# Odds Ratio

Let's measure the ratio between the odds computed at a certain input  $\mathbf{x}$  and the odds computed at a different point  $\mathbf{x}'$

# Odds Ratio

Let's measure the ratio between the odds computed at a certain input  $\mathbf{x}$  and the odds computed at a different point  $\mathbf{x}'$

$$\mathbf{x} = (x_1, \dots, x_i, \dots, x_d)$$
$$\mathbf{x}' = (x_1, \dots, x_i + 1, \dots, x_d)$$

$\mathbf{x}'$  is just the same as  $\mathbf{x}$  where the  $i$ -th predictor/feature is increased by 1 unit

# Odds Ratio

Let's measure the ratio between the odds computed at a certain input  $\mathbf{x}$  and the odds computed at a different point  $\mathbf{x}'$

$$\mathbf{x} = (x_1, \dots, x_i, \dots, x_d)$$
$$\mathbf{x}' = (x_1, \dots, x_i + 1, \dots, x_d)$$

$\mathbf{x}'$  is just the same as  $\mathbf{x}$  where the  $i$ -th predictor/feature is increased by 1 unit

$$\frac{e^{\theta^T \mathbf{x}'}}{e^{\theta^T \mathbf{x}}}$$

Odds Ratio

# Odds Ratio

$$\frac{e^{\boldsymbol{\theta}^T \mathbf{x}'}}{e^{\boldsymbol{\theta}^T \mathbf{x}}} =$$

# Odds Ratio

$$\frac{e^{\boldsymbol{\theta}^T \mathbf{x}'}}{e^{\boldsymbol{\theta}^T \mathbf{x}}} =$$

$$\frac{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i (x_i + 1) + \dots + \theta_d x_d}}{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}}$$



# Odds Ratio

$$\frac{e^{\boldsymbol{\theta}^T \mathbf{x}'}}{e^{\boldsymbol{\theta}^T \mathbf{x}}} =$$

$$\frac{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i (x_i + 1) + \dots + \theta_d x_d}}{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} = \frac{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d} * e^{\theta_i}}{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}}$$

# Odds Ratio

$$\frac{e^{\theta^T \mathbf{x}'}}{e^{\theta^T \mathbf{x}}} =$$

$$\frac{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i (x_i + 1) + \dots + \theta_d x_d}}{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} = \frac{\cancel{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} * e^{\theta_i}}{\cancel{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}}}$$

# Odds Ratio

$$\frac{e^{\theta^T \mathbf{x}'}}{e^{\theta^T \mathbf{x}}} =$$

$$\frac{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i (x_i + 1) + \dots + \theta_d x_d}}{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} = \frac{\cancel{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} * e^{\theta_i}}{\cancel{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}}}$$

$$= e^{\theta_i}$$

The ratio of the odds for 1-unit increase in  $x_i$

# Odds Ratio

$$\frac{e^{\theta^T \mathbf{x}'}}{e^{\theta^T \mathbf{x}}} =$$

$$\frac{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i (x_i + 1) + \dots + \theta_d x_d}}{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} = \frac{\cancel{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}} * e^{\theta_i}}{\cancel{e^{\theta_0 + \theta_1 x_1 + \dots + \theta_i x_i + \dots + \theta_d x_d}}}$$

$$= e^{\theta_i}$$

The ratio of the odds for 1-unit increase in  $x_i$

or

$\theta_i$  is the ratio of the natural log(odds) for 1-unit increase in  $x_i$

# Why Odds Ratio?

This ratio is **constant**: it does not change according to the value of the other  $x_j$  because they cancel out in the calculation

# Why Odds Ratio?

This ratio is **constant**: it does not change according to the value of the other  $x_j$  because they cancel out in the calculation

If we used probability rather than odds this wouldn't be constant!

# Why Odds Ratio?

This ratio is **constant**: it does not change according to the value of the other  $x_j$  because they cancel out in the calculation

If we used probability rather than odds this wouldn't be constant!

The effect of  $x_i$  on the probability of success  $p$  is different depending on the value of  $x_i$

# Why Odds Ratio?

This ratio is **constant**: it does not change according to the value of the other  $x_j$  because they cancel out in the calculation

If we used probability rather than odds this wouldn't be constant!

The effect of  $x_i$  on the probability of success  $p$  is different depending on the value of  $x_i$

## Example

An odds ratio of 1.08 will give an 8% increase in the odds at **any** value of  $x_i$



# Probabilistically-Generated Data

As with any other supervised learning problem we are given a finite set  $D$  of  $m$  i.i.d. labelled examples which we can try to learn from

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$$

where each  $y_i$  is a binary variable taking on two values (e.g.,  $\{-1, +1\}$ )

# Probabilistically-Generated Data

That means we **do not** have access to the individual probability associated with each training sample!

# Probabilistically-Generated Data

That means we **do not** have access to the individual probability associated with each training sample!

The data we observe from  $D$  is actually generated by an underlying and unknown probability function (**noisy target**) which we want to estimate

$$P(y|\mathbf{x}) = \begin{cases} \phi(\mathbf{x}) & \text{if } y = +1 \\ 1 - \phi(\mathbf{x}) & \text{if } y = -1 \end{cases}$$

# Deterministic vs. Noisy Target

- Deterministic function: given  $\mathbf{x}$  as input it always outputs either  $y = +1$  or  $y = -1$  (mutually exclusive)

# Deterministic vs. Noisy Target

- Deterministic function: given  $\mathbf{x}$  as input it always outputs either  $y = +1$  or  $y = -1$  (mutually exclusive)
- Noisy target function: given  $\mathbf{x}$  as input it always outputs both  $y = +1$  and  $y = -1$ , each with a "degree of certainty" associated

# Deterministic vs. Noisy Target

- Deterministic function: given  $\mathbf{x}$  as input it always outputs either  $y = +1$  or  $y = -1$  (mutually exclusive)
- Noisy target function: given  $\mathbf{x}$  as input it always outputs both  $y = +1$  and  $y = -1$ , each with a "degree of certainty" associated

## Goal

$\phi: \mathbb{R}^{d+1} \rightarrow [0,1]$  is the unknown noisy target which generates our examples, our aim is to find an estimate  $\phi^*$  which best approximates  $\phi$

# Estimating Noisy Target

$$P(y|\mathbf{x}) = \begin{cases} \phi^*(\mathbf{x}) & \text{if } y = +1 \\ 1 - \phi^*(\mathbf{x}) & \text{if } y = -1 \end{cases}$$

# Estimating Noisy Target

$$P(y|\mathbf{x}) = \begin{cases} \phi^*(\mathbf{x}) & \text{if } y = +1 \\ 1 - \phi^*(\mathbf{x}) & \text{if } y = -1 \end{cases}$$

We claim that the best estimate  $\phi^*$  of  $\phi$  is  $h_{\boldsymbol{\theta}^*}^*(\mathbf{x})$ , which in turn is picked from the set of hypotheses defined by logistic function

$$\phi^*(\mathbf{x}) = h_{\boldsymbol{\theta}^*}^*(\mathbf{x}) = \ell(\boldsymbol{\theta}^{*T} \mathbf{x}) \approx \phi(\mathbf{x})$$



# Hypothesized Noisy Target

- How do we estimate  $h^*_{\theta}(\mathbf{x})$ ?

# Hypothesized Noisy Target

- How do we estimate  $h^*_{\theta}(\mathbf{x})$ ?
- We will use the same general framework introduced for the supervised learning problem!

# Hypothesized Noisy Target

- How do we estimate  $h^*_{\theta}(\mathbf{x})$ ?
- We will use the same general framework introduced for the supervised learning problem!
- We already fixed the set of hypothesis function to select from

# Hypothesized Noisy Target

- How do we estimate  $h^*_{\theta}(\mathbf{x})$ ?
- We will use the same general framework introduced for the supervised learning problem!
- We already fixed the set of hypothesis function to select from
- We still need:
  - A training set  $D$
  - An error measure (cost function) to minimize

# COST FUNCTION

# Finding The Best Hypothesis

$$\overbrace{P(h_{\boldsymbol{\theta}} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\boldsymbol{\theta}})}^{\text{likelihood}} \times \overbrace{P(h_{\boldsymbol{\theta}})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

# Finding The Best Hypothesis

$$\overbrace{P(h_{\theta} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\theta})}^{\text{likelihood}} \times \overbrace{P(h_{\theta})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

← Bayes Rule

# Finding The Best Hypothesis

$$\overbrace{P(h_{\boldsymbol{\theta}} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\boldsymbol{\theta}})}^{\text{likelihood}} \times \overbrace{P(h_{\boldsymbol{\theta}})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

2 main ways to find the estimate of the best hypothesis parameters  $\boldsymbol{\theta}^*$



# Finding The Best Hypothesis

$$\overbrace{P(h_{\theta} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\theta})}^{\text{likelihood}} \times \overbrace{P(h_{\theta})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

2 main ways to find the estimate of the best hypothesis parameters  $\theta^*$

Maximum Likelihood Estimate  
(MLE)

Frequentist approach

# Finding The Best Hypothesis

$$\overbrace{P(h_{\theta} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\theta})}^{\text{likelihood}} \times \overbrace{P(h_{\theta})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

2 main ways to find the estimate of the best hypothesis parameters  $\theta^*$

Maximum A Posteriori  
(MAP)

Bayesian approach

# Finding The Best Hypothesis

$$\overbrace{P(h_{\theta} | \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} | h_{\theta})}^{\text{likelihood}} \times \overbrace{P(h_{\theta})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

2 main ways to find the estimate of the best hypothesis parameters  $\theta^*$

Maximum Likelihood Estimate  
(MLE)

Frequentist approach

Maximum A Posteriori  
(MAP)

Bayesian approach

# Finding The Best Hypothesis

$$\overbrace{P(h_{\theta} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\theta})}^{\text{likelihood}} \times \overbrace{P(h_{\theta})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

**MLE** returns the set of parameters that **maximize** the **likelihood**

$$h_{\theta}^* = h_{\theta}^{\text{MLE}} = \operatorname{argmax}_{h_{\theta} \in \mathcal{H}} P(\mathcal{D} \mid h_{\theta})$$

# Finding The Best Hypothesis

$$\overbrace{P(h_{\theta} \mid \mathcal{D})}^{\text{posterior}} = \frac{\overbrace{P(\mathcal{D} \mid h_{\theta})}^{\text{likelihood}} \times \overbrace{P(h_{\theta})}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{evidence}}}$$

**MAP** returns the set of parameters that **maximize** the **posterior**

$$\begin{aligned} h_{\theta}^* &= h_{\theta}^{\text{MAP}} = \operatorname{argmax}_{h_{\theta} \in \mathcal{H}} P(h_{\theta} \mid \mathcal{D}) \\ &= \operatorname{argmax}_{h_{\theta} \in \mathcal{H}} \frac{P(\mathcal{D} \mid h_{\theta}) \times P(h_{\theta})}{P(\mathcal{D})} \\ &= \operatorname{argmax}_{h_{\theta} \in \mathcal{H}} P(\mathcal{D} \mid h_{\theta}) \times P(h_{\theta}) \end{aligned}$$

# MLE vs. MAP

MLE is just a special case of MAP where priors are uniform  
(i.e., every hypothesis is equiprobable)

# MLE vs. MAP

MLE is just a special case of MAP where priors are uniform  
(i.e., every hypothesis is equiprobable)

Both MLE and MAP are point estimators: they return a single value for the optimal parameter vector  $\theta^*$

# MLE vs. MAP

MLE is just a special case of MAP where priors are uniform  
(i.e., every hypothesis is equiprobable)

Both MLE and MAP are point estimators: they return a single value for the optimal parameter vector  $\theta^*$

## Note

A full Bayesian estimation is also possible, where the full posterior distribution (i.e., probability density/mass function) is estimated, although this turns out to be often computationally intractable



# MLE: Maximizing The Likelihood Function

We measure the error we are making by assuming that  $h_{\theta}^*(\mathbf{x})$  approximates the true noisy target  $\phi$

# MLE: Maximizing The Likelihood Function

We measure the error we are making by assuming that  $h^*_{\theta}(\mathbf{x})$  approximates the true noisy target  $\phi$

How likely is that the observed data  $D$  have been generated by our selected hypothesis  $h^*_{\theta}(\mathbf{x})$ ?

# MLE: Maximizing The Likelihood Function

We measure the error we are making by assuming that  $h^*_{\theta}(\mathbf{x})$  approximates the true noisy target  $\phi$

How likely is that the observed data  $D$  have been generated by our selected hypothesis  $h^*_{\theta}(\mathbf{x})$ ?

Find the hypothesis which maximizes the probability of the observed data  $D$  given a particular hypothesis

$$h^*_{\theta} = \operatorname{argmax}_{h_{\theta} \in \mathcal{H}} P(\mathcal{D} | h_{\theta})$$

# The Likelihood Function

Given the generic training example  $(\mathbf{x}, y)$  and assuming it has been generated by a hypothesis  $h_{\theta}(\mathbf{x})$  the likelihood function is:

$$P(y|\mathbf{x}) = \begin{cases} h_{\theta}(\mathbf{x}) & \text{if } y = +1 \\ 1 - h_{\theta}(\mathbf{x}) & \text{if } y = -1 \end{cases}$$

where  $\phi$  has been replaced with our hypothesis

# The Likelihood Function

If we assume the hypothesis is the logistic function

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(\boldsymbol{\theta}^T \mathbf{x})$$

# The Likelihood Function

If we assume the hypothesis is the logistic function

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(\boldsymbol{\theta}^T \mathbf{x})$$

And by noticing that logistic function is symmetric, i.e.,  $\ell(-z) = 1 - \ell(z)$ , the likelihood for a single example is:

$$P(y \mid \mathbf{x}) = \ell(y\boldsymbol{\theta}^T \mathbf{x})$$

# The Likelihood Function

Having access to a full set of  $m$  i.i.d. training examples  $D$

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$$

The overall likelihood function is computed as:

$$\prod_{i=1}^m P(y_i \mid \mathbf{x}_i) = \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$$

# Why Does Likelihood Make Sense?

How does the likelihood  $\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$  changes w.r.t. the sign of  $y_i$  and  $\boldsymbol{\theta}^T \mathbf{x}_i$ ?

	$\boldsymbol{\theta}^T \mathbf{x}_i > 0$	$\boldsymbol{\theta}^T \mathbf{x}_i < 0$
$y_i > 0$	$\approx 1$	$\approx 0$
$y_i < 0$	$\approx 0$	$\approx 1$



# Why Does Likelihood Make Sense?

How does the likelihood  $\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$  changes w.r.t. the sign of  $y_i$  and  $\boldsymbol{\theta}^T \mathbf{x}_i$ ?

	$\boldsymbol{\theta}^T \mathbf{x}_i > 0$	$\boldsymbol{\theta}^T \mathbf{x}_i < 0$
$y_i > 0$	$\approx 1$	$\approx 0$
$y_i < 0$	$\approx 0$	$\approx 1$

If the label is **concordant** with the signal (either positively or negatively)  
then  $\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$  approaches to 1

prediction agrees with the true label

# Why Does Likelihood Make Sense?

How does the likelihood  $\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$  changes w.r.t. the sign of  $y_i$  and  $\boldsymbol{\theta}^T \mathbf{x}_i$ ?

	$\boldsymbol{\theta}^T \mathbf{x}_i > 0$	$\boldsymbol{\theta}^T \mathbf{x}_i < 0$
$y_i > 0$	$\approx 1$	$\approx 0$
$y_i < 0$	$\approx 0$	$\approx 1$

If the label is **disoncordant** with the signal then  $\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$  approaches to 0

prediction disagrees with the true label

# Maximum Likelihood Estimate (MLE)

Find the vector of parameters  $\boldsymbol{\theta}$  such that the likelihood function is maximum

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \prod_{i=1}^m P(y_i \mid \mathbf{x}_i) \right) = \operatorname{argmax}_{\boldsymbol{\theta}} \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right)$$

# From MLE to In-Sample Error

Given a hypothesis  $h_{\theta}$  and a training set  $D$  of  $m$  labelled samples we are interested in measuring the "in-sample" (i.e. *training*) error

# From MLE to In-Sample Error

Given a hypothesis  $h_{\boldsymbol{\theta}}$  and a training set  $D$  of  $m$  labelled samples we are interested in measuring the "in-sample" (i.e. *training*) error

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m e(h_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$$

where  $e()$  measures how "far" the chosen hypothesis is from the true observed value

# From MLE to In-Sample Error

Given a hypothesis  $h_{\boldsymbol{\theta}}$  and a training set  $D$  of  $m$  labelled samples we are interested in measuring the "in-sample" (i.e. *training*) error

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m e(h_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$$

where  $e()$  measures how "far" the chosen hypothesis is from the true observed value

How we can "transform" MLE to the "in-sample" error above?

# Negative Log-Likelihood

$$\operatorname{argmax}_{\theta} \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right)$$

# Negative Log-Likelihood

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right)$$

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$



# Negative Log-Likelihood

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right)$$

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right) = \operatorname{argmin}_{\boldsymbol{\theta}} \left( -\frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

# Negative Log-Likelihood

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right)$$

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right) = \operatorname{argmin}_{\boldsymbol{\theta}} \left( -\frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

$$= \operatorname{argmin}_{\boldsymbol{\theta}} \left( -\frac{1}{m} \ln \left( \ell(y_1 \boldsymbol{\theta}^T \mathbf{x}_1) \right) - \dots - \frac{1}{m} \ln \left( \ell(y_m \boldsymbol{\theta}^T \mathbf{x}_m) \right) \right)$$

$$\text{as } k \ln(a \cdot b) = k(\ln(a) + \ln(b)) = k \ln(a) + k \ln(b).$$

# Negative Log-Likelihood

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \qquad \operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

$$\operatorname{argmax}_{\boldsymbol{\theta}} \left( \frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right) = \operatorname{argmin}_{\boldsymbol{\theta}} \left( -\frac{1}{m} \ln \left( \prod_{i=1}^m \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

$$= \operatorname{argmin}_{\boldsymbol{\theta}} \left( -\frac{1}{m} \ln \left( \ell(y_1 \boldsymbol{\theta}^T \mathbf{x}_1) \right) - \dots - \frac{1}{m} \ln \left( \ell(y_m \boldsymbol{\theta}^T \mathbf{x}_m) \right) \right)$$

$$\text{as } k \ln(a \cdot b) = k(\ln(a) + \ln(b)) = k \ln(a) + k \ln(b).$$

$$= \operatorname{argmin}_{\boldsymbol{\theta}} \left( \frac{1}{m} \sum_{i=1}^m -\ln \left( \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) \right) \right)$$

$$= \operatorname{argmin}_{\boldsymbol{\theta}} \left( \frac{1}{m} \sum_{i=1}^m \ln \left( \frac{1}{\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)} \right) \right)$$

$$\text{as } -\ln(a) = \ln\left(\frac{1}{a}\right).$$

# Cross-Entropy Error

$$\operatorname{argmin}_{\boldsymbol{\theta}} \left( \frac{1}{m} \sum_{i=1}^m \ln \left( \frac{1}{\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)} \right) \right)$$

# Cross-Entropy Error

$$\operatorname{argmin}_{\boldsymbol{\theta}} \left( \frac{1}{m} \sum_{i=1}^m \ln \left( \frac{1}{\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)} \right) \right)$$

By noticing that logistic function can be rewritten as follows:

$$\ell(z) = \frac{e^z}{1+e^z} = \frac{1}{e^{-z}+1}$$

We can finally write the "in-sample" error to be minimized:

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

# Cross-Entropy Error

$$\operatorname{argmin}_{\boldsymbol{\theta}} \left( \frac{1}{m} \sum_{i=1}^m \ln \left( \frac{1}{\ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)} \right) \right)$$

By noticing that logistic function can be rewritten as follows:

$$\ell(z) = \frac{e^z}{1+e^z} = \frac{1}{e^{-z}+1}$$

We can finally write the "in-sample" error to be minimized:

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

Cross-Entropy Error

# Cross-Entropy (a.k.a. Log-Loss) Formulations

2 formulations of cross-entropy can be found depending on the labeling chosen for the (binary) response  $y$

# Cross-Entropy (a.k.a. Log-Loss) Formulations

2 formulations of cross-entropy can be found depending on the labeling chosen for the (binary) response  $y$

$$\frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$y = \{-1, +1\}$$



# Cross-Entropy (a.k.a. Log-Loss) Formulations

2 formulations of cross-entropy can be found depending on the labeling chosen for the (binary) response  $y$

$$\frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$y = \{-1, +1\}$$

$$-\frac{1}{m} \sum_{i=1}^m y_i \ln(p) + (1 - y_i) \ln(1 - p)$$

$$p = \frac{e^{\boldsymbol{\theta}^T \mathbf{x}}}{e^{\boldsymbol{\theta}^T \mathbf{x}} + 1} = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

$$y = \{0, 1\}$$

# Cross-Entropy (a.k.a. Log-Loss) Formulations

$$Y = \{0, 1\}$$
$$Y \sim \text{Bernoulli}(p)$$

$$\boxed{f_Y(y; p)} = \boxed{L_Y(p; y)} = \begin{cases} p & \text{if } y = 1 \\ q = 1 - p & \text{if } y = 0 \end{cases}$$

Probability density function of a Bernoulli-distributed random variable with known parameter  $p$

Likelihood of an observed Bernoulli-distributed random variable (parameter  $p$  is unknown)

# Likelihood Function

Likelihood function of  $m$  **i.i.d.** observations of  $Y$

$$L_Y(p; y_1 \dots y_m) = \prod_{i=1}^m p^{y_i} (1-p)^{(1-y_i)}$$

# Likelihood Function

Likelihood function of  $m$  **i.i.d.** observations of  $Y$

$$L_Y(p; y_1 \dots y_m) = \prod_{i=1}^m p^{y_i} (1 - p)^{(1-y_i)}$$

Here the unknown is the parameter  $p$  and we use the observations  $y_1, \dots, y_m$  to find  $p$  so as to maximize the likelihood

$$p^* = \operatorname{argmax}_p \left\{ \prod_{i=1}^m p^{y_i} (1 - p)^{(1-y_i)} \right\}$$

# Negative Log-Likelihood Function

$$p^* = \operatorname{argmin}_p \left\{ -\ln \left[ \prod_{i=1}^m p^{y_i} (1-p)^{(1-y_i)} \right] \right\}$$

# Negative Log-Likelihood Function

$$p^* = \operatorname{argmin}_p \left\{ -\ln \left[ \prod_{i=1}^m p^{y_i} (1-p)^{(1-y_i)} \right] \right\}$$

$$p^* = \operatorname{argmin}_p \left\{ -\sum_{i=1}^m \ln \left[ p^{y_i} (1-p)^{(1-y_i)} \right] \right\}$$

# Negative Log-Likelihood Function

$$p^* = \operatorname{argmin}_p \left\{ -\ln \left[ \prod_{i=1}^m p^{y_i} (1-p)^{(1-y_i)} \right] \right\}$$

$$p^* = \operatorname{argmin}_p \left\{ -\sum_{i=1}^m \ln \left[ p^{y_i} (1-p)^{(1-y_i)} \right] \right\}$$

$$p^* = \operatorname{argmin}_p \left\{ -\sum_{i=1}^m \ln(p^{y_i}) + \ln \left( (1-p)^{(1-y_i)} \right) \right\}$$

# Negative Log-Likelihood Function

$$p^* = \operatorname{argmin}_p \left\{ - \sum_{i=1}^m \ln(p^{y_i}) + \ln\left((1-p)^{(1-y_i)}\right) \right\}$$



# Negative Log-Likelihood Function

$$p^* = \operatorname{argmin}_p \left\{ - \sum_{i=1}^m \ln(p^{y_i}) + \ln\left((1-p)^{(1-y_i)}\right) \right\}$$

$$p^* = \operatorname{argmin}_p \left\{ - \sum_{i=1}^m y_i \ln(p) + (1-y_i) \ln(1-p) \right\}$$

# Negative Log-Likelihood Function

$$p^* = \operatorname{argmin}_p \left\{ - \sum_{i=1}^m \ln(p^{y_i}) + \ln((1-p)^{(1-y_i)}) \right\}$$

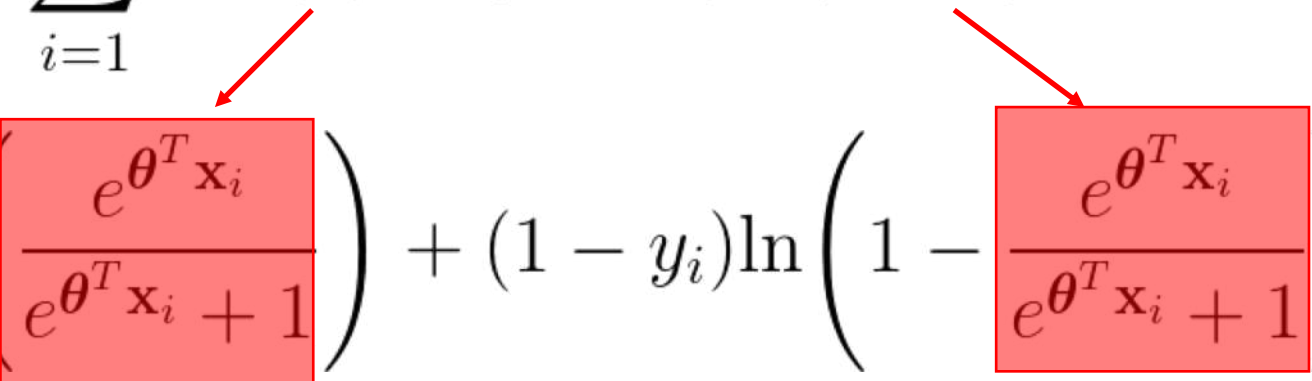
$$p^* = \operatorname{argmin}_p \left\{ - \sum_{i=1}^m y_i \ln(p) + (1-y_i) \ln(1-p) \right\}$$

Except for the  $1/m$  factor this is **exactly** the second formulation we gave for the cross-entropy error

# Substituting $p$

$$-\sum_{i=1}^m y_i \ln(p) + (1 - y_i) \ln(1 - p)$$

# Substituting $p$

$$-\sum_{i=1}^m y_i \ln(p) + (1 - y_i) \ln(1 - p)$$

$$-\sum_{i=1}^m y_i \ln\left(\frac{e^{\theta^T \mathbf{x}_i}}{e^{\theta^T \mathbf{x}_i} + 1}\right) + (1 - y_i) \ln\left(1 - \frac{e^{\theta^T \mathbf{x}_i}}{e^{\theta^T \mathbf{x}_i} + 1}\right)$$

# Substituting $p$

$$\begin{aligned} & - \sum_{i=1}^m y_i \ln(p) + (1 - y_i) \ln(1 - p) \\ & - \sum_{i=1}^m y_i \ln \left( \frac{e^{\theta^T \mathbf{x}_i}}{e^{\theta^T \mathbf{x}_i} + 1} \right) + (1 - y_i) \ln \left( 1 - \frac{e^{\theta^T \mathbf{x}_i}}{e^{\theta^T \mathbf{x}_i} + 1} \right) \\ & - \sum_{i=1}^m y_i [\ln(e^{\theta^T \mathbf{x}_i}) - \ln(e^{\theta^T \mathbf{x}_i} + 1)] + (1 - y_i) [\ln(1) - \ln(e^{\theta^T \mathbf{x}_i} + 1)] \end{aligned}$$

## Substituting $p$

$$-\sum_{i=1}^m y_i [\ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)] + (1 - y_i) [\ln(\cancel{1}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)]$$

0

## Substituting $p$

$$\begin{aligned} & - \sum_{i=1}^m y_i [\ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)] + (1 - y_i) [\cancel{\ln(1)} - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)] \\ & \qquad \qquad \qquad 0 \\ & - \sum_{i=1}^m y_i \boldsymbol{\theta}^T \mathbf{x}_i - y_i \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1) + y_i \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1) \end{aligned}$$

## Substituting $p$

$$- \sum_{i=1}^m y_i [\ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)] + (1 - y_i) [\cancel{\ln(1)} - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)]$$
$$- \sum_{i=1}^m y_i \boldsymbol{\theta}^T \mathbf{x}_i - \cancel{y_i \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)} - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1) + \cancel{y_i \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)}$$



# Substituting $p$

$$- \sum_{i=1}^m y_i [\ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)] + (1 - y_i) [\cancel{\ln(1)} - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)]$$

0

$$- \sum_{i=1}^m y_i \boldsymbol{\theta}^T \mathbf{x}_i - \cancel{y_i \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)} - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1) + \cancel{y_i \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)}$$

$$- \sum_{i=1}^m y_i \boldsymbol{\theta}^T \mathbf{x}_i - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

# Equivalence Between 2 Formulations

We want to show the 2 formulations below lead to the same function to be minimized

$$\sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$y = \{-1, +1\}$$

$$-\sum_{i=1}^m y_i \boldsymbol{\theta}^T \mathbf{x}_i - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$y = \{0, 1\}$$

# Equivalence Between 2 Formulations

We want to show the 2 formulations below lead to the same function to be minimized

$$\boxed{\sum_{i=1}^m \ln(e^{\theta^T \mathbf{x}_i} + 1)}_{y = -1} = \boxed{\sum_{i=1}^m \ln(e^{\theta^T \mathbf{x}_i} + 1)}_{y = 0}$$

# Equivalence Between 2 Formulations

We want to show the 2 formulations below lead to the same function to be minimized

$$\sum_{i=1}^m \ln(e^{-\boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$y = 1$$

$\stackrel{?}{=}$

$$-\sum_{i=1}^m \boldsymbol{\theta}^T \mathbf{x}_i - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$y = 1$$

# Equivalence Between 2 Formulations

$$\boxed{\sum_{i=1}^m \ln(e^{-\boldsymbol{\theta}^T \mathbf{x}_i} + 1)} = \sum_{i=1}^m \ln\left(\frac{1}{e^{\boldsymbol{\theta}^T \mathbf{x}_i}} + 1\right) = \sum_{i=1}^m \ln\left(\frac{1 + e^{\boldsymbol{\theta}^T \mathbf{x}_i}}{e^{\boldsymbol{\theta}^T \mathbf{x}_i}}\right)$$

# Equivalence Between 2 Formulations

$$\boxed{\sum_{i=1}^m \ln(e^{-\boldsymbol{\theta}^T \mathbf{x}_i} + 1)} = \sum_{i=1}^m \ln\left(\frac{1}{e^{\boldsymbol{\theta}^T \mathbf{x}_i}} + 1\right) = \sum_{i=1}^m \ln\left(\frac{1 + e^{\boldsymbol{\theta}^T \mathbf{x}_i}}{e^{\boldsymbol{\theta}^T \mathbf{x}_i}}\right)$$
$$= \sum_{i=1}^m \ln(1 + e^{\boldsymbol{\theta}^T \mathbf{x}_i}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i})$$

# Equivalence Between 2 Formulations

$$\begin{aligned} \boxed{\sum_{i=1}^m \ln(e^{-\boldsymbol{\theta}^T \mathbf{x}_i} + 1)} &= \sum_{i=1}^m \ln\left(\frac{1}{e^{\boldsymbol{\theta}^T \mathbf{x}_i}} + 1\right) = \sum_{i=1}^m \ln\left(\frac{1 + e^{\boldsymbol{\theta}^T \mathbf{x}_i}}{e^{\boldsymbol{\theta}^T \mathbf{x}_i}}\right) \\ &= \sum_{i=1}^m \ln(1 + e^{\boldsymbol{\theta}^T \mathbf{x}_i}) - \ln(e^{\boldsymbol{\theta}^T \mathbf{x}_i}) \\ &= \boxed{-\sum_{i=1}^m \boldsymbol{\theta}^T \mathbf{x}_i - \ln(1 + e^{\boldsymbol{\theta}^T \mathbf{x}_i})} \end{aligned}$$

# LEARNING ALGORITHM



# Picking the Best Hypothesis

- So far, we have defined:
  - The model (logistic function)
  - The error measure (cross-entropy)

# Picking the Best Hypothesis

- So far, we have defined:
  - The model (logistic function)
  - The error measure (cross-entropy)

To actually select the best hypothesis, we have to pick the vector of parameters  $\boldsymbol{\theta}^*$  so that the error measure is minimized

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

# Mean Squared Error vs. Cross-Entropy

In the case of linear regression we have a similar expression for the error measure, i.e. Mean Squared Error (MSE)

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m (\boldsymbol{\theta}^T \mathbf{x}_i - y_i)^2$$

# Mean Squared Error vs. Cross-Entropy

In the case of linear regression we have a similar expression for the error measure, i.e. Mean Squared Error (MSE)

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m (\boldsymbol{\theta}^T \mathbf{x}_i - y_i)^2$$

Minimising MSE through Ordinary Least Squares (OLS) leads to a **closed-form solution** often referred to as the OLS estimator for  $\boldsymbol{\theta}^*$

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

# Mean Squared Error vs. Cross-Entropy

The problem is that using Cross-Entropy as error measure we **cannot** find a closed-form solution to the minimization problem

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

# Mean Squared Error vs. Cross-Entropy

The problem is that using Cross-Entropy as error measure we **cannot** find a closed-form solution to the minimization problem

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

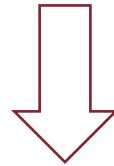
Yet, Cross-Entropy is **convex** w.r.t. the parameters  $\boldsymbol{\theta}$

# Mean Squared Error vs. Cross-Entropy

The problem is that using Cross-Entropy as error measure we **cannot** find a closed-form solution to the minimization problem

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

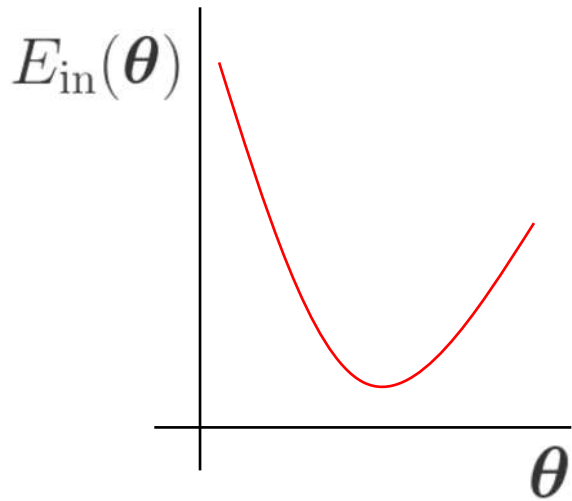
Yet, Cross-Entropy is **convex** w.r.t. the parameters  $\boldsymbol{\theta}$



Iterative Solution

# (Batch) Gradient Descent

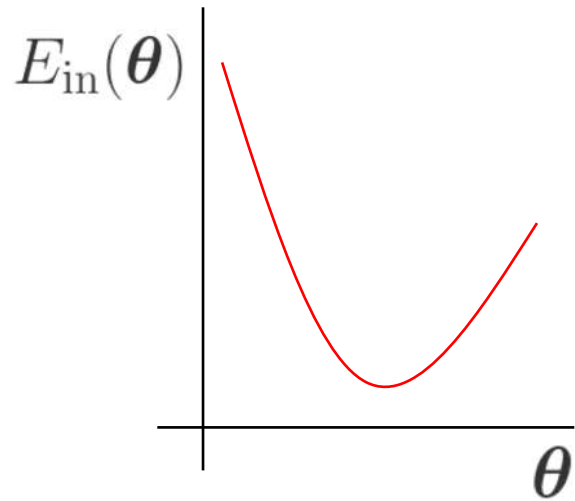
General iterative method for any nonlinear optimization





# (Batch) Gradient Descent

General iterative method for any nonlinear optimization

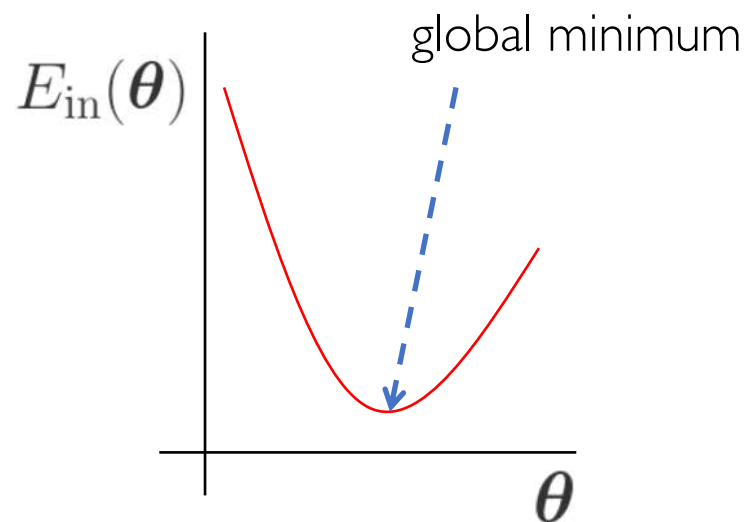


The method **guarantees the convergence to a local minimum**

(Under specific assumptions on the objective function and learning rate)

# (Batch) Gradient Descent

General iterative method for any nonlinear optimization



The method **guarantees the convergence to a local minimum**

(Under specific assumptions on the objective function and learning rate)

If the objective function is **convex** (like cross-entropy)  
then the local minimum is also the **global minimum**

# Gradient Descent: The Main Idea

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$

# Gradient Descent: The Main Idea

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$
2. Repeat until convergence:
  - a. Update the current vector of parameters  $\boldsymbol{\theta}(t)$  by taking a "step" along the "steepest" slope:  $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$
  - b. Return to 2.

# Gradient Descent: The Main Idea

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$
2. Repeat until convergence:
  - a. Update the current vector of parameters  $\boldsymbol{\theta}(t)$  by taking a "step" along the "steepest" slope:  $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$
  - b. Return to 2.

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$$

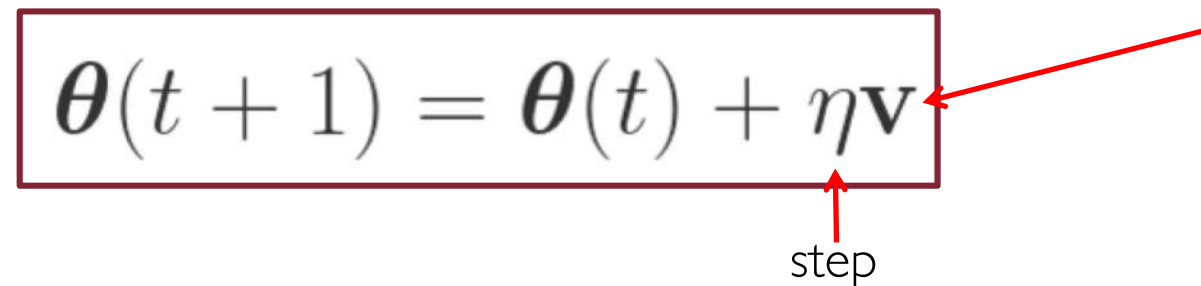
Unit vector representing the direction of the steepest slope

step

# Gradient Descent: The Main Idea

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$
2. Repeat until convergence:
  - a. Update the current vector of parameters  $\boldsymbol{\theta}(t)$  by taking a "step" along the "steepest" slope:  $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$
  - b. Return to 2.

Unit vector representing the direction of the steepest slope



The diagram shows the equation  $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$  enclosed in a red rectangular box. A red arrow points from the text 'Unit vector representing the direction of the steepest slope' to the vector  $\mathbf{v}$  in the equation. Another red arrow points from the text 'step' to the scalar  $\eta$  in the equation.

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$$

How do we determine the direction  $\mathbf{v}$ ?

# Gradient Descent: The Direction $\mathbf{v}$

- We already intuitively said that the direction  $\mathbf{v}$  should be that of the "steepest" slope

# Gradient Descent: The Direction $\mathbf{v}$

- We already intuitively said that the direction  $\mathbf{v}$  should be that of the "steepest" slope
- Concretely, this means moving along the direction which mostly reduces the in-sample error function

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t)) - E_{\text{in}}(\boldsymbol{\theta}(t - 1))$$



# Gradient Descent: The Direction $\mathbf{v}$

- We already intuitively said that the direction  $\mathbf{v}$  should be that of the "steepest" slope
- Concretely, this means moving along the direction which mostly reduces the in-sample error function

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t)) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

We want  $\Delta E_{\text{in}}$  to be **as negative as possible**, which means that we are actually reducing the error w.r.t. the previous iteration  $t-1$

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

---

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

---

Let's first assume we are in the **univariate** case, i.e.,  $\boldsymbol{\theta} = \vartheta$  in  $\mathbb{R}$

$$f = E_{\text{in}}$$

$$x_0 = \boldsymbol{\theta}(t-1)$$

$$x = \boldsymbol{\theta}(t)$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the **univariate** case, i.e.,  $\boldsymbol{\theta} = \vartheta$  in  $\mathbb{R}$

$$f = E_{\text{in}}$$

$$x_0 = \boldsymbol{\theta}(t-1)$$

$$x = \boldsymbol{\theta}(t)$$

$$\delta f = \Delta E_{\text{in}} = f(x) - f(x_0)$$

$$\delta x = x - x_0 = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \eta \mathbf{v}$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the **univariate** case, i.e.,  $\boldsymbol{\theta} = \vartheta$  in  $\mathbb{R}$

$$f = E_{\text{in}}$$

$$x_0 = \boldsymbol{\theta}(t-1)$$

$$x = \boldsymbol{\theta}(t)$$

$$\delta f = \Delta E_{\text{in}} = f(x) - f(x_0)$$

$$\delta x = x - x_0 = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \eta \mathbf{v}$$

$$f'(x_0) = \lim_{\delta x \rightarrow 0} \frac{f(x_0 + \delta x) - f(x_0)}{\delta x}$$

$$f'(x_0) = \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \approx \frac{\delta f}{\delta x}$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the **univariate** case, i.e.,  $\boldsymbol{\theta} = \vartheta$  in  $\mathbb{R}$

$$f = E_{\text{in}}$$

$$x_0 = \boldsymbol{\theta}(t-1)$$

$$x = \boldsymbol{\theta}(t)$$

$$\delta f = \Delta E_{\text{in}} = f(x) - f(x_0)$$

$$\delta x = x - x_0 = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \eta \mathbf{v}$$

$$f'(x_0) = \lim_{\delta x \rightarrow 0} \frac{f(x_0 + \delta x) - f(x_0)}{\delta x}$$

$$f'(x_0) = \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \approx \frac{\delta f}{\delta x}$$

$$\delta f = f(x) - f(x_0) \approx f'(x_0) \delta x = f'(x_0)(x - x_0)$$

# Gradient Descent: The Direction $\mathbf{v}$

$$f(x) - f(x_0) \approx f'(x_0)(x - x_0)$$

# Gradient Descent: The Direction $\mathbf{v}$

$$f(x) - f(x_0) \approx f'(x_0)(x - x_0)$$

$$f(x) = \underbrace{f(x_0) + f'(x_0)(x - x_0)}_{\text{First-order Taylor approximation}} + \underbrace{O((x - x_0)^2)}_{\text{Second-order error term}}$$



# Gradient Descent: The Direction $\mathbf{v}$

$$f(x) - f(x_0) \approx f'(x_0)(x - x_0)$$

$$f(x) = \underbrace{f(x_0) + f'(x_0)(x - x_0)}_{\text{First-order Taylor approximation}} + \underbrace{O((x - x_0)^2)}_{\text{Second-order error term}}$$

To summarize and generalize to the multivariate case of  $\boldsymbol{\theta}$ :

$$\delta f = f(x) - f(x_0) = \Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + O(\eta^2)$$

The greek letter *nabla* indicates the gradient

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + O(\eta^2)$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + O(\eta^2)$$

The unit vector  $\mathbf{v}$  only contributes to the **direction** and not to the magnitude of the iterative step

# Gradient Descent: The Direction $\mathbf{v}$

$$\Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + \cancel{O(\eta^2)}$$

The unit vector  $\mathbf{v}$  only contributes to the **direction** and not to the magnitude of the iterative step

The second-order approximation term is negligible  
(when the step size is small)

# Gradient Descent: The Direction $\mathbf{v}$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$
$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$
$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \underbrace{\|\mathbf{v}\|}_{=1} \cos(\alpha) = \|\mathbf{u}\| \cos(\alpha)$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$
$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \underbrace{\|\mathbf{v}\|}_{=1} \cos(\alpha) = \|\mathbf{u}\| \cos(\alpha) \quad -1 \leq \cos(\alpha) \leq 1$$

# Gradient Descent: The Direction $\mathbf{v}$

$$\begin{aligned}\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T &= \mathbf{u} \\ \Delta E_{\text{in}} &= \eta \mathbf{u} \cdot \mathbf{v}\end{aligned}$$

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \underbrace{\|\mathbf{v}\|}_{=1} \cos(\alpha) = \|\mathbf{u}\| \cos(\alpha) \quad -1 \leq \cos(\alpha) \leq 1$$


$$\begin{aligned}-\|\mathbf{u}\| &\leq \mathbf{u} \cdot \mathbf{v} \leq \|\mathbf{u}\| \\ -\eta\|\mathbf{u}\| &\leq \underbrace{\eta \mathbf{u} \cdot \mathbf{v}}_{\Delta E_{\text{in}}} \leq \eta\|\mathbf{u}\|\end{aligned}$$



# Gradient Descent: The Direction $\mathbf{v}$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$
$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \underbrace{\|\mathbf{v}\|}_{=1} \cos(\alpha) = \|\mathbf{u}\| \cos(\alpha) \quad -1 \leq \cos(\alpha) \leq 1$$

$$-\|\mathbf{u}\| \leq \mathbf{u} \cdot \mathbf{v} \leq \|\mathbf{u}\|$$
$$-\eta \|\mathbf{u}\| \leq \underbrace{\eta \mathbf{u} \cdot \mathbf{v}}_{\Delta E_{\text{in}}} \leq \boxed{\eta \|\mathbf{u}\|}$$


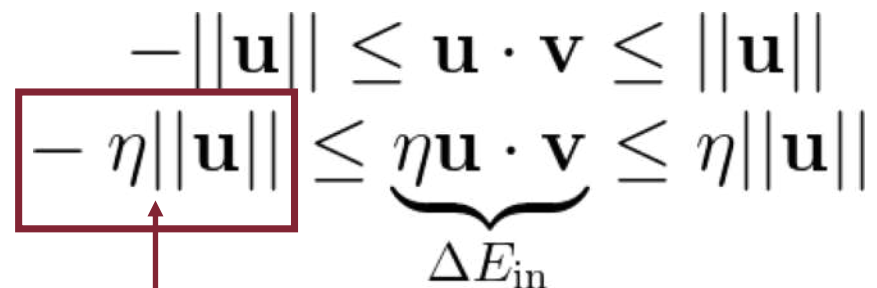
The most **positive**  $\Delta E_{\text{in}}$  when  $\cos(\alpha) = 1$  (i.e.,  $\alpha = 0^\circ$ )

Both error and step vectors have the same direction

# Gradient Descent: The Direction $\mathbf{v}$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$
$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \underbrace{\|\mathbf{v}\|}_{=1} \cos(\alpha) = \|\mathbf{u}\| \cos(\alpha) \quad -1 \leq \cos(\alpha) \leq 1$$

$$-\|\mathbf{u}\| \leq \mathbf{u} \cdot \mathbf{v} \leq \|\mathbf{u}\|$$
$$\boxed{-\eta\|\mathbf{u}\|} \leq \underbrace{\eta \mathbf{u} \cdot \mathbf{v}}_{\Delta E_{\text{in}}} \leq \eta\|\mathbf{u}\|$$


The most **negative**  $\Delta E_{\text{in}}$  when  $\cos(\alpha) = -1$  (i.e.,  $\alpha = 180^\circ$ )

The error and step vectors have opposite direction

# Gradient Descent: The Direction $\mathbf{v}$

At each iteration  $t$ , we want the unit vector  $\mathbf{v}$  which makes exactly **the most negative**  $\Delta E_{\text{in}}$

$$\eta \mathbf{u} \cdot \mathbf{v} = -\eta ||\mathbf{u}||$$

# Gradient Descent: The Direction $\mathbf{v}$

At each iteration  $t$ , we want the unit vector  $\mathbf{v}$  which makes exactly **the most negative**  $\Delta E_{\text{in}}$

$$\eta \mathbf{u} \cdot \mathbf{v} = -\eta ||\mathbf{u}||$$

$$\begin{aligned}\mathbf{u} \cdot \mathbf{v} &= -||\mathbf{u}|| \\ \mathbf{u}^T \cdot \mathbf{u} \cdot \mathbf{v} &= -||\mathbf{u}|| \mathbf{u}^T\end{aligned}$$

$$\mathbf{v} = -\frac{||\mathbf{u}|| \mathbf{u}^T}{||\mathbf{u}||^2} = -\frac{\mathbf{u}^T}{||\mathbf{u}||} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))}{||\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))||}$$

# Gradient Descent: The Direction $\mathbf{v}$

At each iteration  $t$ , we want the unit vector  $\mathbf{v}$  which makes exactly **the most negative**  $\Delta E_{\text{in}}$

$$\eta \mathbf{u} \cdot \mathbf{v} = -\eta \|\mathbf{u}\|$$

$$\begin{aligned} \mathbf{u} \cdot \mathbf{v} &= -\|\mathbf{u}\| \\ \mathbf{u}^T \cdot \mathbf{u} \cdot \mathbf{v} &= -\|\mathbf{u}\| \mathbf{u}^T \end{aligned}$$

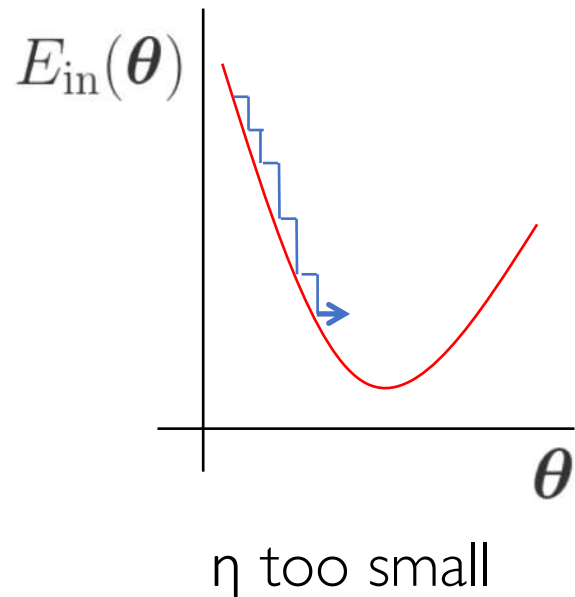
$$\mathbf{v} = -\frac{\|\mathbf{u}\| \mathbf{u}^T}{\|\mathbf{u}\|^2} = -\frac{\mathbf{u}^T}{\|\mathbf{u}\|} = \boxed{-\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))\|}}$$

# Gradient Descent: The Step $\eta$

How the step magnitude  $\eta$  affects the convergence?

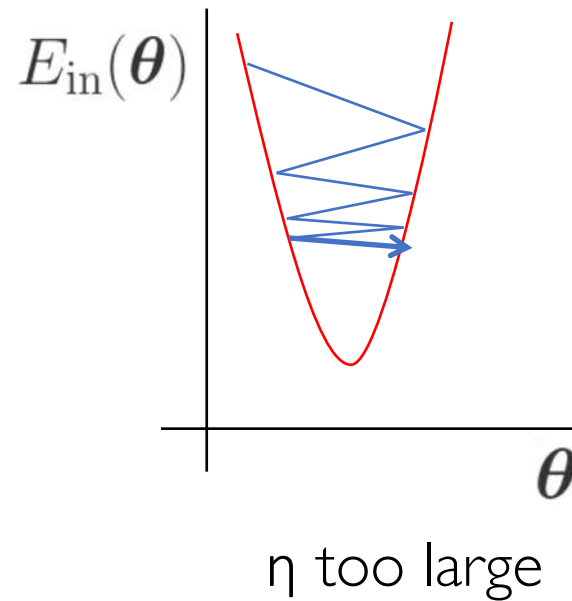
# Gradient Descent: The Step $\eta$

How the step magnitude  $\eta$  affects the convergence?



# Gradient Descent: The Step $\eta$

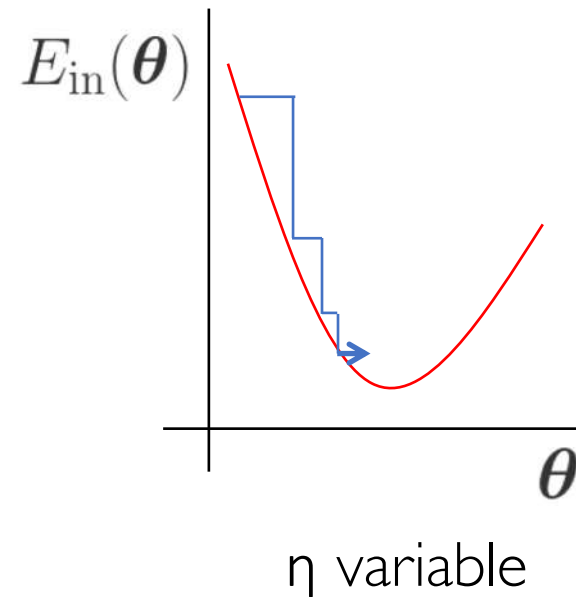
How the step magnitude  $\eta$  affects the convergence?





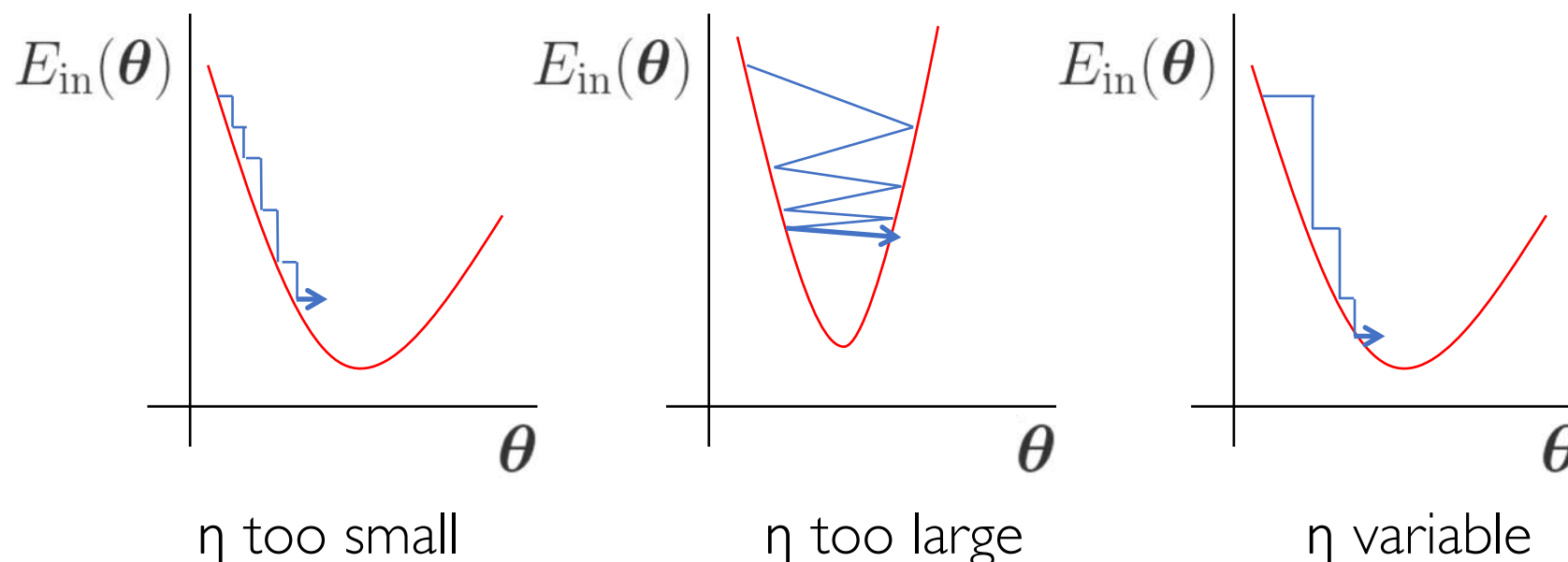
# Gradient Descent: The Step $\eta$

How the step magnitude  $\eta$  affects the convergence?



# Gradient Descent: The Step $\eta$

How the step magnitude  $\eta$  affects the convergence?



Rule of thumb

Dynamically change  $\eta$  proportionally to the gradient!

# Gradient Descent: The Step $\eta$

Remember that at each iteration the update strategy is:

$$\boldsymbol{\theta}(t + 1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step $\eta$

Remember that at each iteration the update strategy is:

$$\boldsymbol{\theta}(t + 1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

At each iteration  $t$ , the step  $\eta$  is fixed

$$\boldsymbol{\theta}(t + 1) = \boldsymbol{\theta}(t) - \eta \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step $\eta$

Instead of having a fixed  $\eta$  at each iteration, use a variable  $\eta_t$  as function of  $\eta$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

# Gradient Descent: The Step $\eta$

Instead of having a fixed  $\eta$  at each iteration, use a variable  $\eta_t$  as function of  $\eta$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

Let's take: 
$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta k \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step $\eta$

Instead of having a fixed  $\eta$  at each iteration, use a variable  $\eta_t$  as function of  $\eta$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

Let's take: 
$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta k \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$



$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \boxed{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|} \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step $\eta$

Instead of having a fixed  $\eta$  at each iteration, use a variable  $\eta_t$  as function of  $\eta$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \quad \eta_t = \eta k$$

Let's take: 
$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta k \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \cancel{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|} \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\cancel{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}}$$

$$\boxed{\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t))}$$



# Computing the Gradient of Cross-Entropy

$$\nabla E_{\text{in}}(\boldsymbol{\theta}) = \nabla \left[ \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]$$

# Computing the Gradient of Cross-Entropy

$$\begin{aligned}\nabla E_{\text{in}}(\boldsymbol{\theta}) &= \nabla \left[ \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] \\ &= \left[ \frac{1}{m} \sum_{i=1}^m \nabla \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]\end{aligned}$$

# Computing the Gradient of Cross-Entropy

$$\begin{aligned}\nabla E_{\text{in}}(\boldsymbol{\theta}) &= \nabla \left[ \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] \\ &= \left[ \frac{1}{m} \sum_{i=1}^m \nabla \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1} \nabla (e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]\end{aligned}$$

chain rule of derivative

# Computing the Gradient of Cross-Entropy

$$\begin{aligned}\nabla E_{\text{in}}(\boldsymbol{\theta}) &= \nabla \left[ \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] \\ &= \left[ \frac{1}{m} \sum_{i=1}^m \nabla \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1} \nabla (e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]\end{aligned}$$

chain rule of derivative

$$= \frac{1}{m} \sum_{i=1}^m \frac{-y_i \mathbf{x}_i e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i}}{e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1}$$

# Computing the Gradient of Cross-Entropy

$$\begin{aligned}\nabla E_{\text{in}}(\boldsymbol{\theta}) &= \nabla \left[ \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] \\ &= \left[ \frac{1}{m} \sum_{i=1}^m \nabla \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right] = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1} \nabla (e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]\end{aligned}$$

chain rule of derivative

$$= \frac{1}{m} \sum_{i=1}^m \frac{-y_i \mathbf{x}_i e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i}}{e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1} = -\frac{1}{m} \sum_{i=1}^m \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}^T \mathbf{x}_i}}$$

# Gradient Descent: The Algorithm

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$

# Gradient Descent: The Algorithm

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$
2. For  $t = 0, 1, 2, \dots$  until stop:

a. Compute the gradient of the cross-entropy error

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = -\frac{1}{m} \sum_{i=1}^m \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}(t)^T \mathbf{x}_i}}$$

# Gradient Descent: The Algorithm

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$
2. For  $t = 0, 1, 2, \dots$  until stop:

- a. Compute the gradient of the cross-entropy error  $E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = -\frac{1}{m} \sum_{i=1}^m \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}(t)^T \mathbf{x}_i}}$$

- b. Update the vector of parameters:  $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t))$
- c. Return to 2.



# Gradient Descent: The Algorithm

1. At  $t = 0$  initialize the (guessed) vector of parameters  $\boldsymbol{\theta}$  to  $\boldsymbol{\theta}(0)$

2. For  $t = 0, 1, 2, \dots$  until stop:

a. Compute the gradient of the cross-entropy error

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1)$$

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = -\frac{1}{m} \sum_{i=1}^m \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}(t)^T \mathbf{x}_i}}$$

b. Update the vector of parameters:  $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t))$

c. Return to 2.

3. Return the final vector of parameters  $\boldsymbol{\theta}(\infty)$

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters  $\theta(0)$ ?

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters  $\theta(0)$ ?
- Typically, random initialization!

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters  $\boldsymbol{\theta}(0)$ ?
- Typically, random initialization!
- If the function is convex we are guaranteed to reach the global minimum no matter what is the initial value of  $\boldsymbol{\theta}(0)$

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters  $\boldsymbol{\theta}(0)$ ?
- Typically, random initialization!
- If the function is convex we are guaranteed to reach the global minimum no matter what is the initial value of  $\boldsymbol{\theta}(0)$
- In general, we may get to the local minimum nearest to  $\boldsymbol{\theta}(0)$

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize **non-convex** objectives

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize **non-convex** objectives
- Problem: non-convex functions may have several local minima

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize **non-convex** objectives
- Problem: non-convex functions may have several local minima
- A bad initialization might cause GD to end up into a "bad" local minimum and miss "better" ones (or even the global if it exists)



# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize **non-convex** objectives
- Problem: non-convex functions may have several local minima
- A bad initialization might cause GD to end up into a "bad" local minimum and miss "better" ones (or even the global if it exists)
- Solution (heuristic): repeating GD 100÷1,000 times each time with a different  $\theta(0)$  may reduce the chance the above issue occurs

# Gradient Descent: Stopping Criterion

- If the function is convex GD reaches the global minimum when

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = 0$$

# Gradient Descent: Stopping Criterion

- If the function is convex GD reaches the global minimum when  $\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = 0$
- In general, we don't know if eventually the gradient gets to 0 therefore we can use several criteria of termination:
  - stop whenever the difference between two iterations is "small enough"  $\rightarrow$  may converge "prematurely"
  - stop when the error equals to  $\varepsilon \rightarrow$  may not converge if the target error is not achievable
  - stop after  $T$  iterations
  - combinations of the above in practice works...

# Gradient Descent: Advanced Topics

- Gradient Descent using second-order approximation
  - Better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)

# Gradient Descent: Advanced Topics

- Gradient Descent using second-order approximation
  - Better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)
- Stochastic vs. Mini-Batch Gradient Descent (SGD vs. MBGD)
  - At each iteration, compute the gradient only from one instance (SGD) or a sample of  $k$  instances (MBGD) rather than the full dataset

# Gradient Descent: Advanced Topics

- Gradient Descent using second-order approximation
  - Better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)
- Stochastic vs. Mini-Batch Gradient Descent (SGD vs. MBGD)
  - At each iteration, compute the gradient only from one instance (SGD) or a sample of  $k$  instances (MBGD) rather than the full dataset
- Regularization
  - Include the L1- or L2-norm of the vector of parameters  $\theta$  in the cross-entropy error to avoid overfitting

# Take-Home Message of Today

- Logistic Regression is a powerful tool for predicting binary variables through probability of each class

# Take-Home Message of Today

- Logistic Regression is a powerful tool for predicting binary variables through probability of each class
- It fits a regression line between input (features) and output (logarithm of the odds), assuming probability takes the form of a sigmoid function



# Take-Home Message of Today

- Logistic Regression is a powerful tool for predicting binary variables through probability of each class
- It fits a regression line between input (features) and output (logarithm of the odds), assuming probability takes the form of a sigmoid function
- Parameter estimation is typically done via MLE (i.e., by minimizing Cross-Entropy error)

# Take-Home Message of Today

- Logistic Regression is a powerful tool for predicting binary variables through probability of each class
- It fits a regression line between input (features) and output (logarithm of the odds), assuming probability takes the form of a sigmoid function
- Parameter estimation is typically done via MLE (i.e., by minimizing Cross-Entropy error)
- No closed-form solution → iterative Gradient Descent