NOVA IMS Information Management School

# Text Mining

## NGrams, TF-IDF, Distance metrics & Classification

Academic Year 2024-2025

Bruno Jardim bjardim@novaims.unl.pt
Rita Oliveira roliveira@novaims.unl.pt

Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

# Lecture Plan

1. N-Grams
2. TF-IDF
3. Distance Metrics
4. Classifiers with Bag-of-Words

NOVA IMS Information Management School

# 1. N-Grams

Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

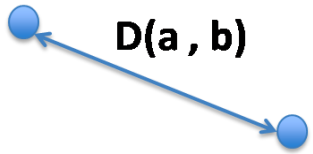
# Bag of Words

Each word is a feature!

Now is the time for all good man to come to the aid of their party.

The quick brown fox jumped over the lazy dog's back.

$$D(a,b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$




| | aid | all | back | brown | come | dog | fox | good | jump | lazy | men | now | over | party | quick | their | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Doc1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Doc2 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

# Problems with Bag of Words

1. Curse of dimensionality

2. No semantic relationships
   (**Word order** is discarded)

3. All words have the same importance

# Problems with Bag of Words

1. Curse of dimensionality

2. No semantic relationships
   **(Word order is discarded)**

3. All words have the same importance

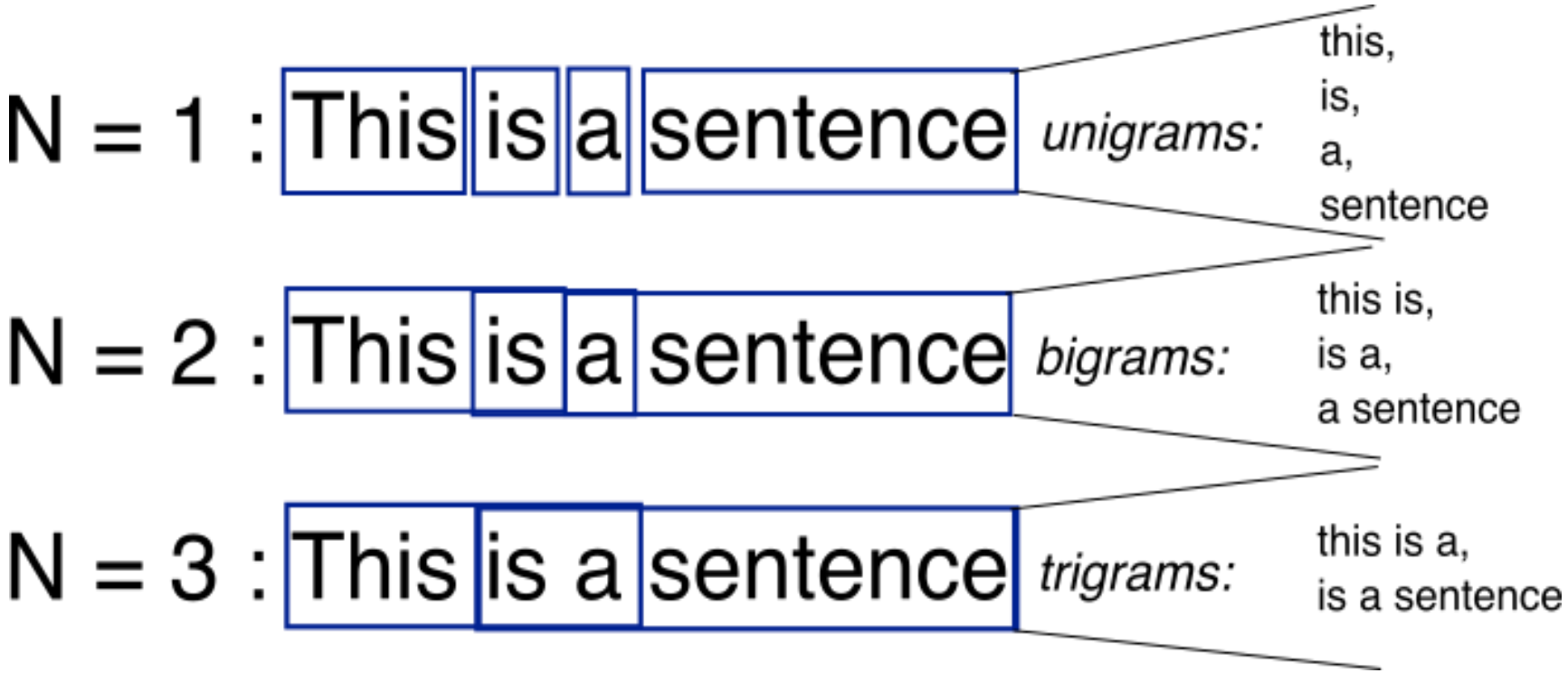
# N-Grams

**Definition:**

An n-gram is a **contiguous sequence of n items** from a given sample (text or speech).

- Although N-grams can be sequences of anything (characters, words, lemmas, etc.) we will see mostly **sequences of words**.

# N-Grams


N = 1 : This is a sentence
unigrams:
this,
is,
a,
sentence
N = 2 : This is a sentence
bigrams:
this is,
is a,
a sentence
N = 3 : This is a sentence
trigrams:
this is a,
is a sentence

# N-Grams Bag of Words

**The BoW can also be done with N–grams (ex: 2-grams):**

Doc 1 (d1) – “The dog is on the table”

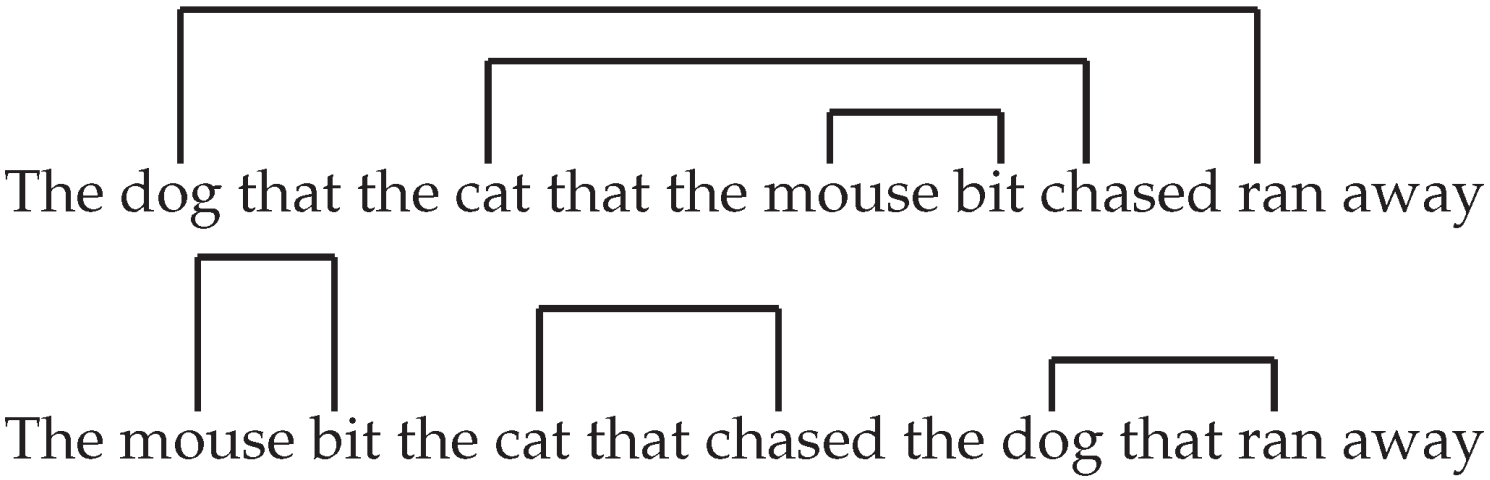
Doc 2 (d2) – “ The cat is on the table”

| | (The, dog) | (dog, is) | (is, on) | (on, the) | (the, table) | (The, cat) | (cat, is) |
|---|---|---|---|---|---|---|---|
| Term Frequency d1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Term Frequency d2 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

# N-Grams Problems

**N-Gram size:**

Sometimes, the occurrence of one word, depends on another word that came way before in text.



# N-Grams Problems

**Data sparseness:**

One of the major drawbacks with N-grams is that they **don't deal well with long distance dependencies**.

E.g:

"*Gollum loves his precious*" could be a frequent 4-gram in Lord of the Rings, but for some reason , our text might only contain things such as: "Gollum loves in a very sick way precious", thus the initial 4-gram will never be found.

Typically **when we increase N** our **results became sparse** (aka we get a lot of zeros in the BoW).

NOVA IMS Information Management School

# 2. TF-IDF

# Problems with BoW (cont.)

1. Curse of dimensionality
2. No semantic relationships
3. **All words have the same importance**

# Problems with BoW (cont.)

**All words have the same importance…**

Possible solution:

- Count the number of times a word appears.

# Problems with BoW (cont.)

**All words have the same importance…**

Possible solution:

- Count the number of times a word appears.

  **Unfortunately, words that appear most typically are not the most important ones…**

# TF-IDF

## Term Frequency – Inverse Document Frequency (TF-IDF)

$$\text{TF-IDF} = \text{TF}(t, d) * \text{IDF}(t, D)$$

$$\text{TF}(t, d) = \text{raw count of } t \text{ in } d$$

$$\text{IDF}(t, D) = log(\frac{|D|}{n_t})$$

$n_t$ = number of documents where the term appears

# TF-IDF

Doc 1 (d1) – “The dog is on the table”

Doc 2 (d2) – “ The cat is on the table”

# TF-IDF

Doc 1 (d1) – “The dog is on the table”

Doc 2 (d2) – “ The cat is on the table”

**TF(“word”, d)** – Count “word” in document **d**

**IDF(“word”, D)** – $\log(|D|/n_t)$, where **|D|** is the number of documents in the corpus (in this example, we have 2 documents) and $\mathbf{n_t}$ is the number of documents **n** with the word **t**

# TF-IDF

Doc 1 (d1) – “The dog is on the table”

Doc 2 (d2) – “The cat is on the table”

$$\mathrm{TF}(t, d) = \text{raw count of } t \text{ in } d$$

| | The | dog | is | on | table | cat |
|---|---|---|---|---|---|---|
| Term Frequency d1 | 2 | 1 | 1 | 1 | 1 | 0 |
| Term Frequency d2 | 2 | 0 | 1 | 1 | 1 | 1 |

# TF-IDF

Doc 1 (d1) – “The dog is on the table”

Doc 2 (d2) – “The cat is on the table”

| | The | dog | is | on | table | cat |
|---|---|---|---|---|---|---|
| Term Frequency d1 | 2 | 1 | 1 | 1 | 1 | 0 |
| Term Frequency d2 | 2 | 0 | 1 | 1 | 1 | 1 |
| **Documents with word** | **2** | **1** | **2** | **2** | **2** | **1** |

# TF-IDF

Doc 1 (d1) – “The dog is on the table”

Doc 2 (d2) – “The cat is on the table”

$$\mathrm{IDF}(t, D) = log(\frac{|D|}{n_t})$$

| | The | dog | is | on | table | cat |
|---|---|---|---|---|---|---|
| Term Frequency d1 | 2 | 1 | 1 | 1 | 1 | 0 |
| Term Frequency d2 | 2 | 0 | 1 | 1 | 1 | 1 |
| Documents with word ($n_t$) | 2 | 1 | 2 | 2 | 2 | 1 |
| **IDF (\|D\| = 2)** | **0** | **0.3** | **0** | **0** | **0** | **0.3** |

# TF-IDF

Doc 1 (d1) – “The dog is on the table”

Doc 2 (d2) – “The cat is on the table”

$$\text{TF-IDF} = \text{TF}(t, d) * \text{IDF}(t, D)$$

| | The | dog | is | on | table | cat |
|---|---|---|---|---|---|---|
| Term Frequency d1 | 2 | 1 | 1 | 1 | 1 | 0 |
| Term Frequency d2 | 2 | 0 | 1 | 1 | 1 | 1 |
| Documents with word | 2 | 1 | 2 | 2 | 2 | 1 |
| IDF (\|D\| = 2) | 0 | 0.3 | 0 | 0 | 0 | 0.3 |
| **TF-IDF d1** | **0** | **0.3** | **0** | **0** | **0** | **0** |
| **TF-IDF d2** | **0** | **0** | **0** | **0** | **0** | **0.3** |

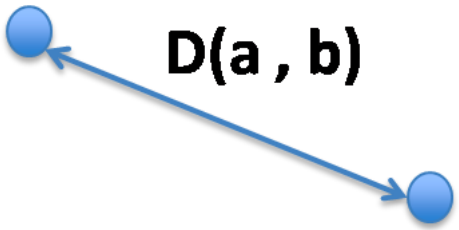
NOVA IMS Information Management School

# 3. Distance Metrics

Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

# Distance Metrics

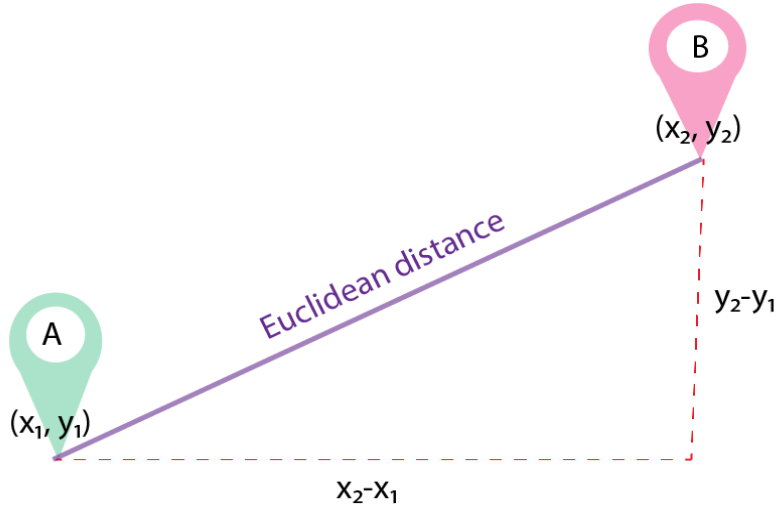
- Given 2 documents **we can transform them into feature space** and **compare them with simple distance metrics**.
- The feature space can be binary values or raw counts or TF-IDF of any kind of N-grams.
- If they **share a lot of features, they will be close to each other**.



# Euclidean Distance

The **Euclidean Distance** is the most common use of distance. When **data is dense** or **continuous**, this is the best proximity measure.

The Euclidean distance between two points is the **length of the path connecting them**. The Pythagorean theorem gives this distance between two points.



# Euclidean Distance

**By representing documents as vectors, we can calculate, for example, the distance between documents…**

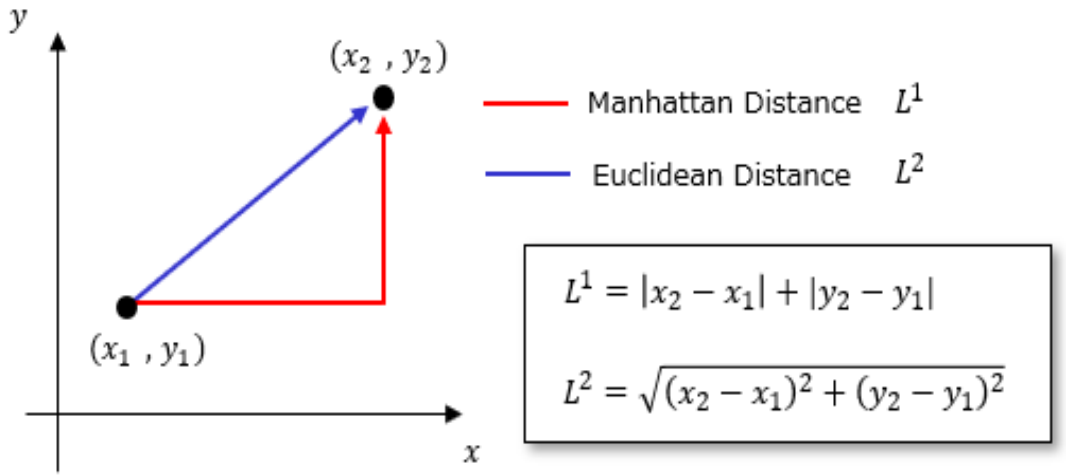
| | The | dog | is | on | table | cat |
|---|---|---|---|---|---|---|
| TF-IDF Doc 1 | 0 | 0.3 | 0 | 0 | 0 | 0 |
| TF-IDF Doc 2 | 0 | 0 | 0 | 0 | 0 | 0.3 |

**Euclidean:**

$$D(d1,d2) = \sqrt{(0-0)^2 + (0.3-0)^2} + \cdots = \sqrt{(0.3)^2 + (-0.3)^2} = 0.42$$

# Manhattan Distance

**Manhattan distance** is a metric in which **the distance between two points is the sum of the absolute differences of their Cartesian coordinates**. In a simple way of saying it is the total sum of the difference between the x-coordinates and y-coordinates.




https://taketake2.com/N102_en.html

# Cosine Distance

**Cosine distance** finds the **normalized dot product of the two attributes**. By determining the cosine similarity, we would effectively try to find the **cosine of the angle between the two objects**. The cosine of 0° is 1, and it is less than 1 for any other angle.

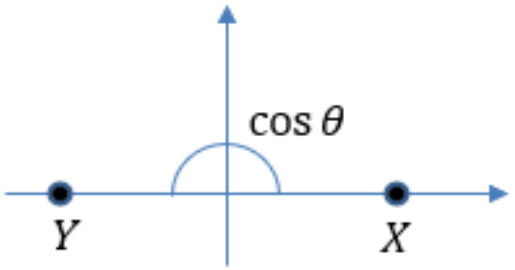
One of the reasons for the popularity of cosine similarity is that it is **very efficient to evaluate, especially for sparse vectors.**

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

# Cosine Distance

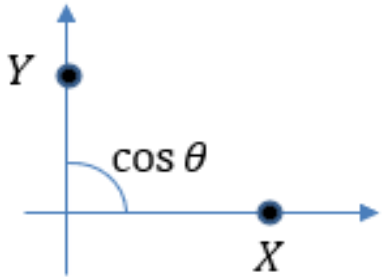
$\cos\theta = r = -1$

⇒ X and Y are negatively correlated



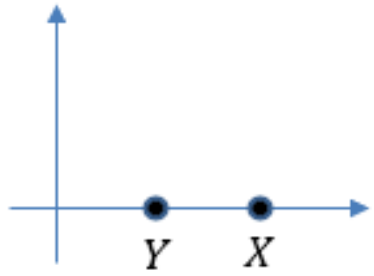

$\cos\theta = r = 0$

⇒ X and Y have no correlation




$\cos\theta = r = 1$

⇒ X and Y are correlated




https://taketake2.com/N102_en.html

# Cosine Distance

| | The | dog | is | on | table | cat |
|---|---|---|---|---|---|---|
| TF-IDF Doc 1 | 0 | 0.3 | 0 | 0 | 0 | 0 |
| TF-IDF Doc 2 | 0 | 0 | 0 | 0 | 0 | 0.3 |

**Cosine Distance:**

$$\text{Cos}(\theta) = \frac{v1 \cdot v2}{\|v1\|\|v2\|} = \frac{\mathbf{0}}{\mathbf{0.3}*\mathbf{0.3}} = \mathbf{0}\ (\mathbf{90°})$$

$$v1 \cdot v2 = (0 * 0) + (0.3 * 0) + (0 * 0) + (0 * 0) + (0 * 0) + (0 * 0.3)$$

$$||v1|| = \sqrt{0^2 + 0.3^2 \quad +0^{\,2}} \ \ldots = \ \sqrt{0.3^2} = \sqrt{0.09} = 0.3$$

$$||v2|| = \sqrt{0^2 + 0^{\,2} + \ \ldots + 0.3^2} = 0.3$$

# Comparing words & documents

2 ways:

**Phonetics** – words

**Orthography** – words and documents

# Phonetics

**Soundex** is a phonetic algorithm for indexing names by sound, as pronounced in English.

Example:

Smith -> S530

Smythe -> S530

George -> G620

Garage -> G620

# Orthography

**Minimum Edit Distance (Levenshtein)** counts the minimum number of operations required to transform one string into the other.

- **Insertion** of a single symbol. If $a = uv$, then inserting the symbol $x$ produces $uxv$. This can also be denoted $\varepsilon \rightarrow x$, using $\varepsilon$ to denote the empty string.
- **Deletion** of a single symbol changes $uxv$ to $uv$ ($x \rightarrow \varepsilon$).
- **Substitution** of a single symbol $x$ for a symbol $y \neq x$ changes $uxv$ to $uyv$ ($x \rightarrow y$)

# Jaccard coefficient

**Jaccard coefficient** measures similarity between finite sample sets, and is defined as the size of the **intersection** divided by the size of the **union** of the sample sets:

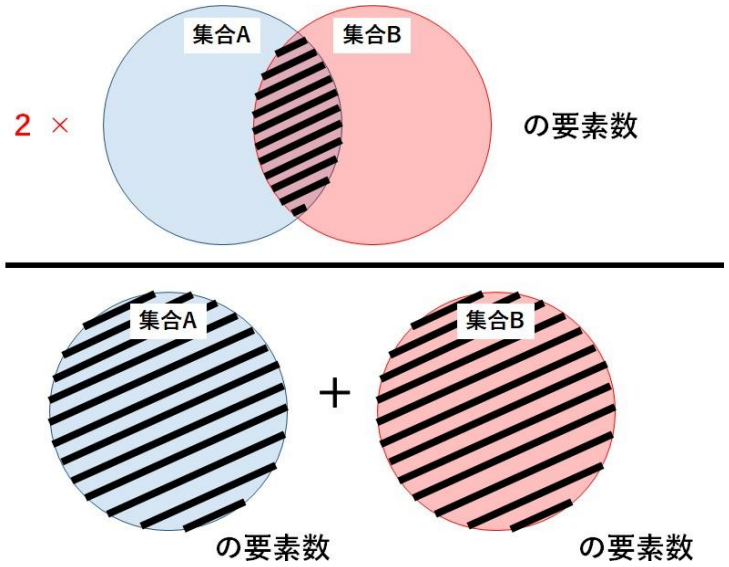
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

# Dice Similarity

**Sørensen–Dice** is similar to Jaccard but **strings with different lengths are not so strongly penalized**

$$Dice(s,t) = 2 \times \frac{|s \cap t|}{|s| + |t|}$$

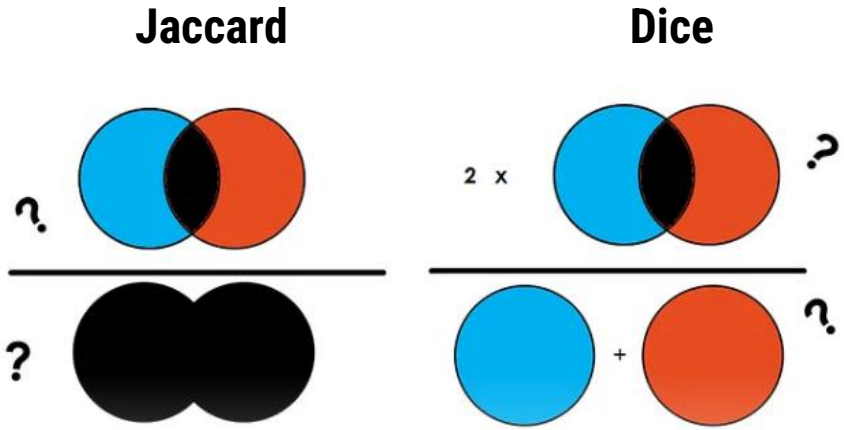
$$DSC(A,B) = \frac{2 \times \text{集合A と 集合B の要素数}}{\text{集合A の要素数} + \text{集合B の要素数}}$$



# Jaccard vs. Dice

**Doc 1 (d1)** – “The dog is on the table”

**Doc 2 (d2)** – “ The cat is on the table”




**Jaccard:**
d1 ∩ d2 = {the, is, on, table} = 4
d1 ∪ d2 = {The, dog, cat, is, on, table} = 6

**J(d1, d2) = 4/6 = 0.67**

**Dice:**
d1 ∩ d2 = {The, is, on, table} = 4
d1 = {The, dog, is, on, table} = 5
d2 = {The, cat, is, on, table} = 5

**DSC(d1, d2) = 2*4/(5+5) = 8/10 = 0.8**

NOVA IMS Information Management School

# 4. Classification with BoW

Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

# Classification with BoW

**Recall BoW**

Review 1 – “Great movie”
Review 2 – “Bad movie”

| | great | movie | bad |
|---|---|---|---|
| R1 | 1 | 1 | 0 |
| R2 | 0 | 1 | 1 |

**Task:**

Predict the sentiment of movie reviews using a neural network with sigmoid activation function.

# Classification with BoW

**Recall BoW**

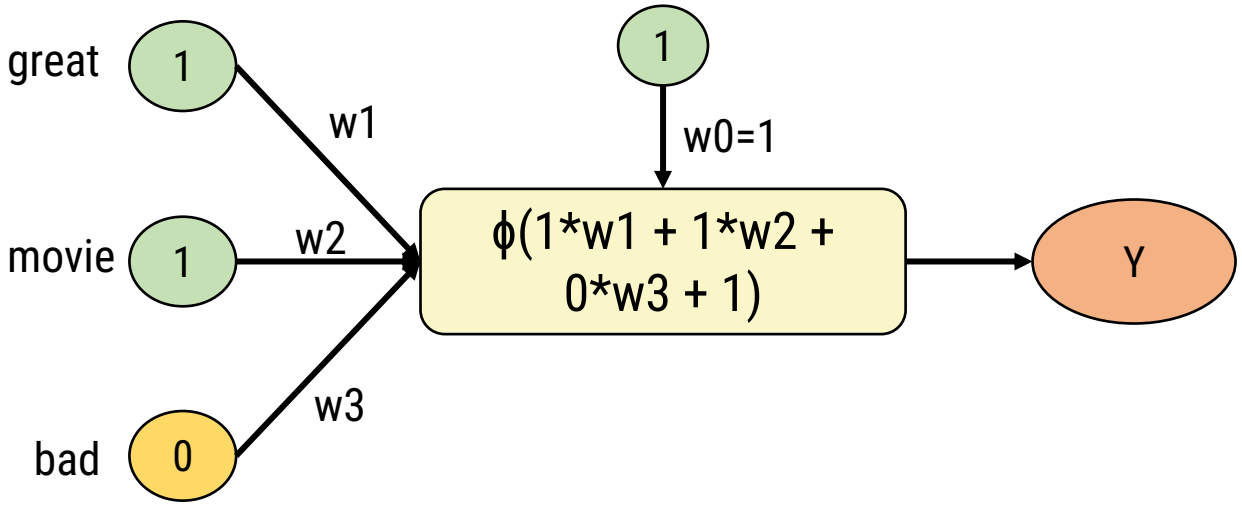
Review 1 – “Great movie”
Review 2 – “Bad movie”

| | great | movie | bad |
|---|---|---|---|
| R1 | 1 | 1 | 0 |
| R2 | 0 | 1 | 1 |

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Classification with BoW

**Recall BoW**

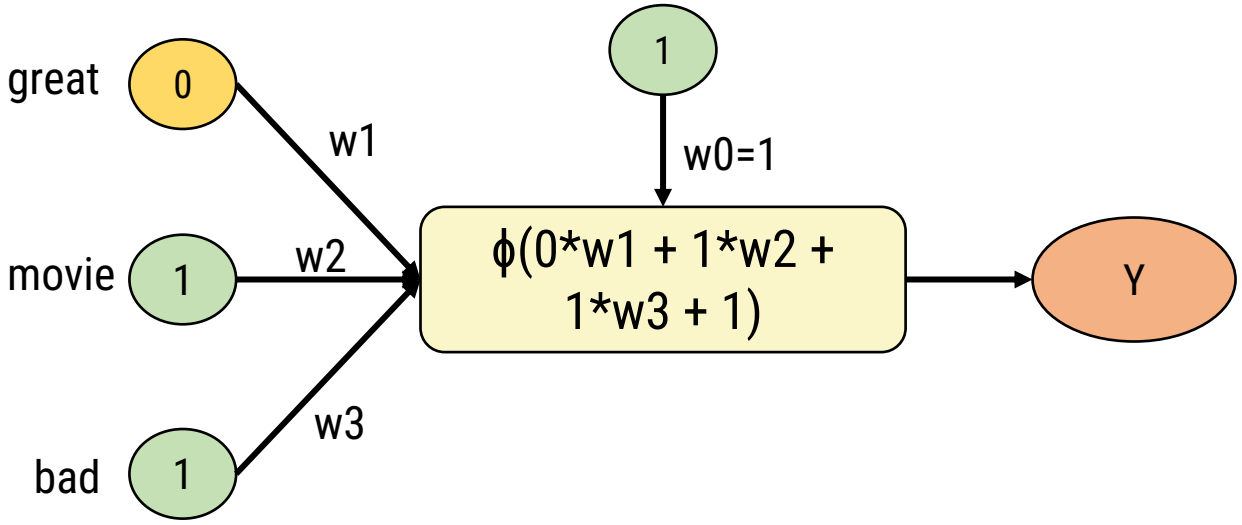
Review 1 – “Great movie”
Review 2 – “Bad movie”

| | great | movie | bad |
|---|---|---|---|
| R1 | 1 | 1 | 0 |
| R2 | 0 | 1 | 1 |

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 2:**



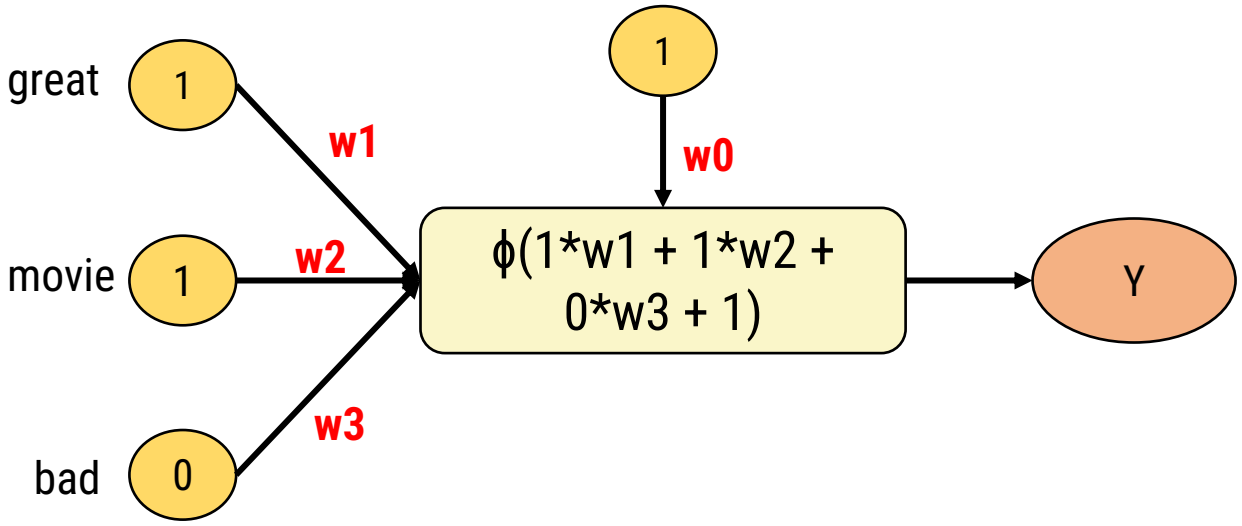
# Classification with BoW

**How to find the best weights?**

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Classification with BoW

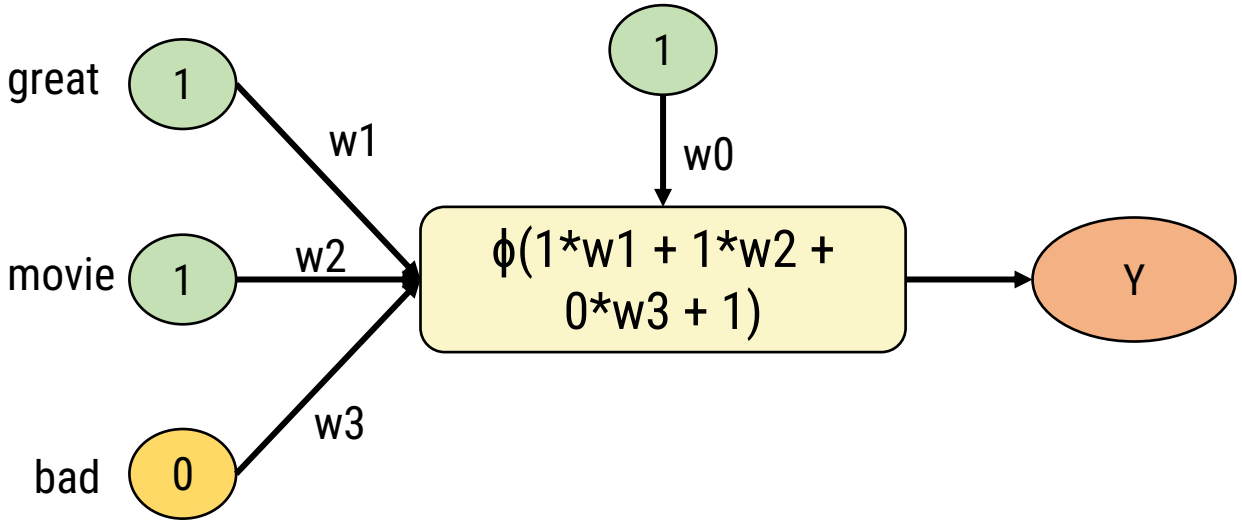
**How to find the best weights?**

1 – Start with random values
2 – Pass the input trough the network
3 – Calculate the error
4 – Backpropagate Error (Update the weights)

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Classification with BoW

**How to find the best weights?**

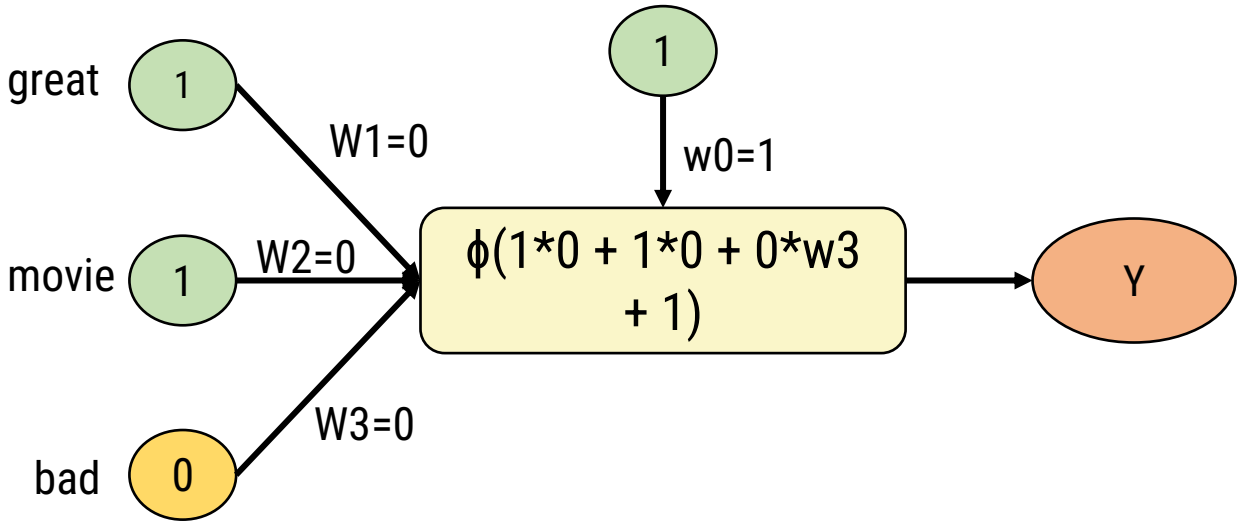
**1 – Start with random values**

w0= 1
w1= 0
w2= 0
w3= 0

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Classification with BoW

**How to find the best weights?**

1 – Start with random values

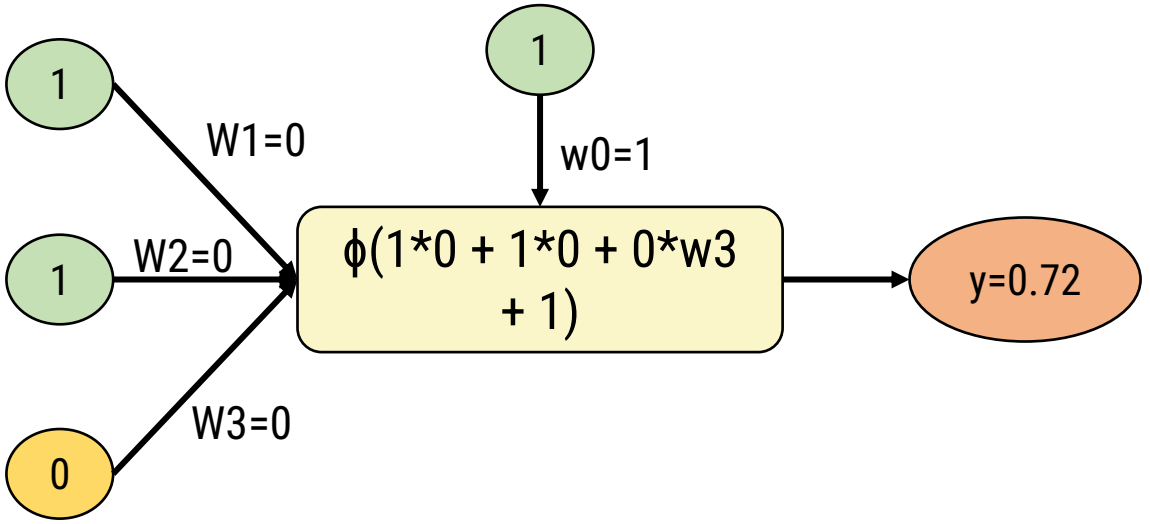
**2 – Pass the input trough the network**

$$\phi(R1) = \frac{1}{1+e^{-(1*0+1*0+0*0+1)}} = \frac{1}{1+0.37} = 0.72$$

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Classification with BoW

**How to find the best weights?**

1 – Start with random values
2 – Pass the input trough the network
**3 – Calculate the error**

## Loss Function

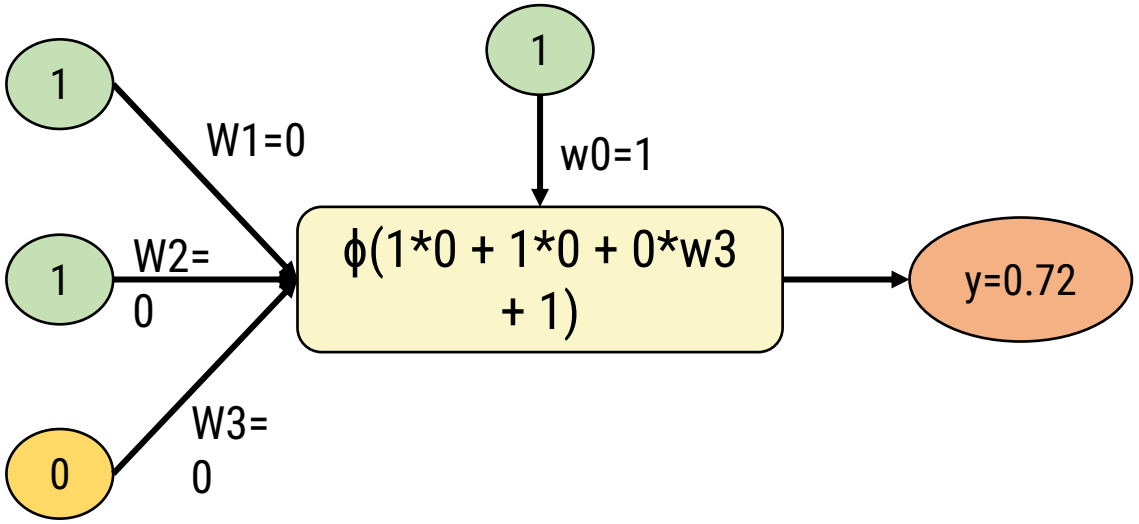
$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

Predicted Actual

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Classification with BoW

**Loss Functions**

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\right)$$

Regression:

**Mean Squared Error Loss**

$$\text{MSE} = \overbrace{\frac{1}{n}\sum_{i=1}^{n}}^{\text{Mean}} \left(\overbrace{Y_i - \hat{Y}_i}^{\text{Error}}\right)^{\overbrace{2}^{\text{Squared}}}$$

Classification:

**Cross-Entropy Loss**

$$J(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}\right) + \left(1 - \underbrace{y^{(i)}}_{\text{Actual}}\right) \log\left(1 - \underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}\right)$$

http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf
https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a
https://www.youtube.com/watch?v=Pwgpl9mKars

# Classification with BoW

**How to find the best weights?**

1 – Start with random values
2 – Pass the input trough the network
**3 – Calculate the error**

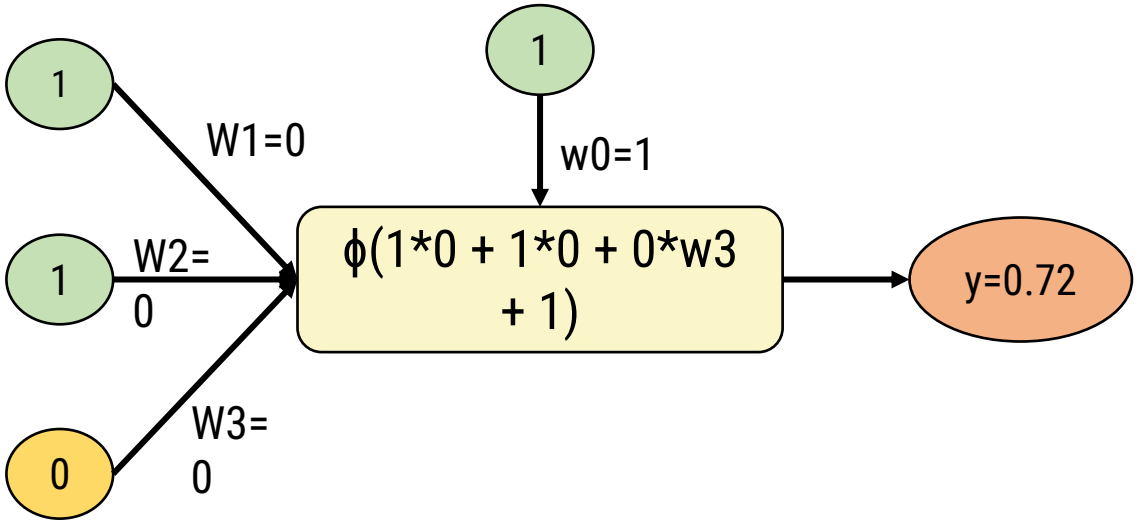
Simple Example:

$$\varepsilon = \boldsymbol{y} - \hat{\boldsymbol{y}} = 1 - 0.72 = 0.28$$

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**

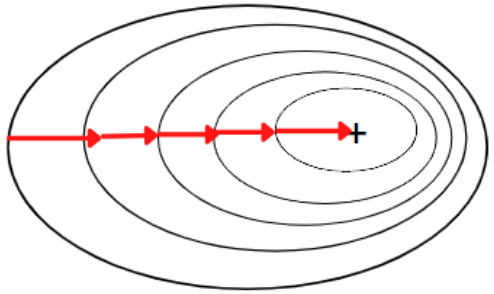


# Classification with BoW

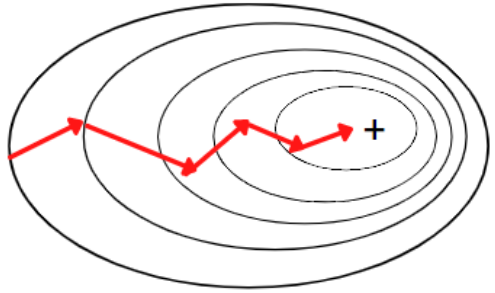
## Training methods

**Batch Gradient Descent**

Many training samples used to update weights



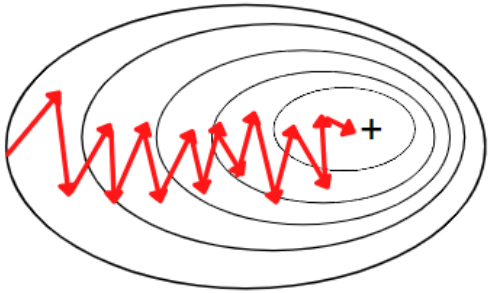
**Mini-Batch Gradient Descent**



Some sample inputs used to update weights

**Stochastic Gradient Descent**

Single sample used to update weights



https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

# Classification with BoW

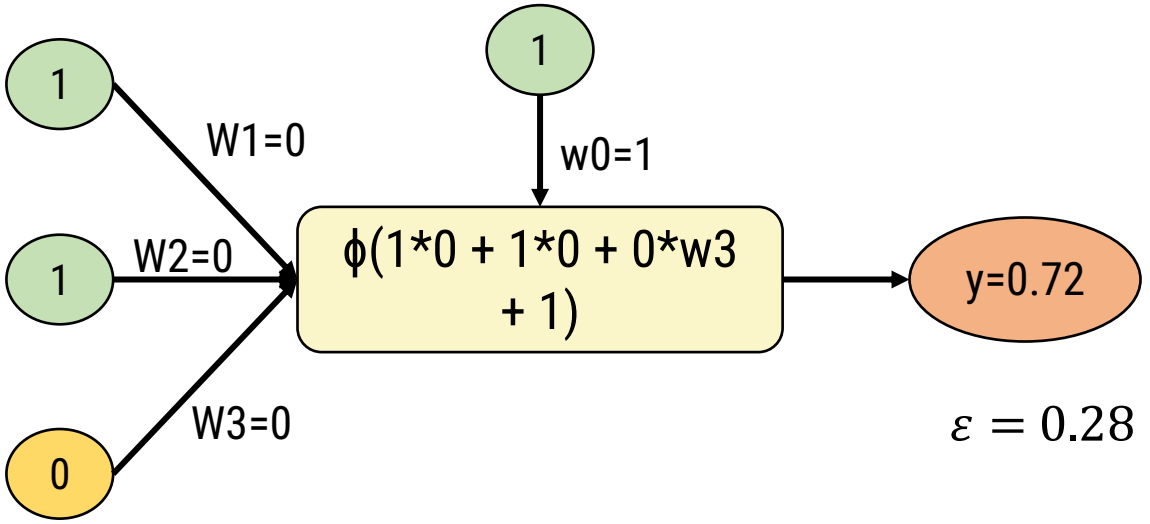
**How to find the best weights?**

1 – Start with random values
2 – Pass the input trough the network
3 – Calculate the error
**4 – Backpropagate Error (Update the weights)**

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**




$\varepsilon = 0.28$

# Classification with BoW

**How to find the best weights?**

**4 – Backpropagate Error (Update the weights)**

Simple derivative:

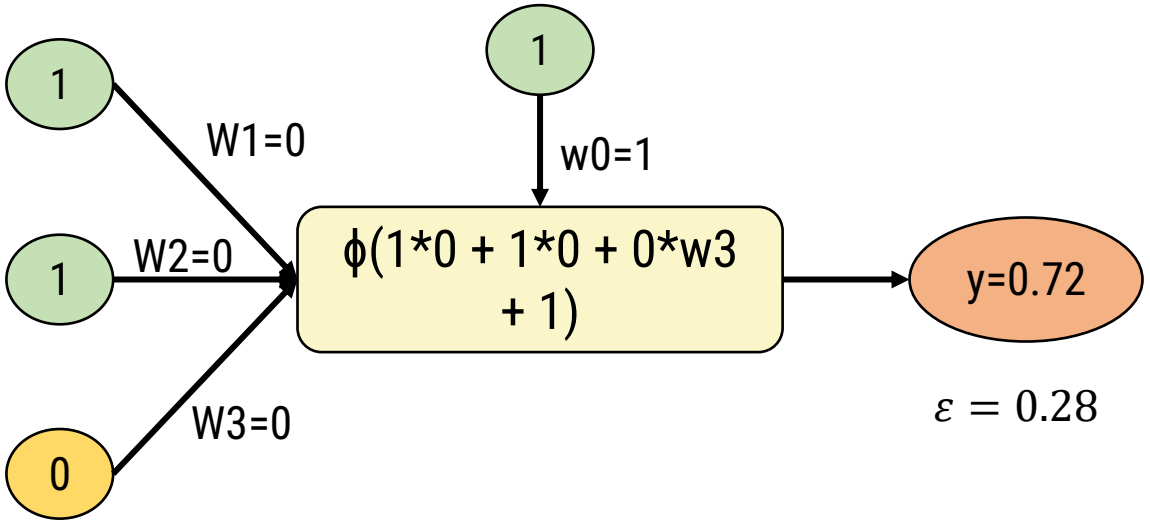
$$\frac{dLoss}{dw} = (\hat{y} - y)x$$

$$\text{new w} = \text{w} - \alpha(\hat{y} - y)x$$

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**




$\varepsilon = 0.28$

# Neural Network in Text

**How to find the best weights?**

**4 – Backpropagate Error (Update the weights)**

With learning rate = 0.05

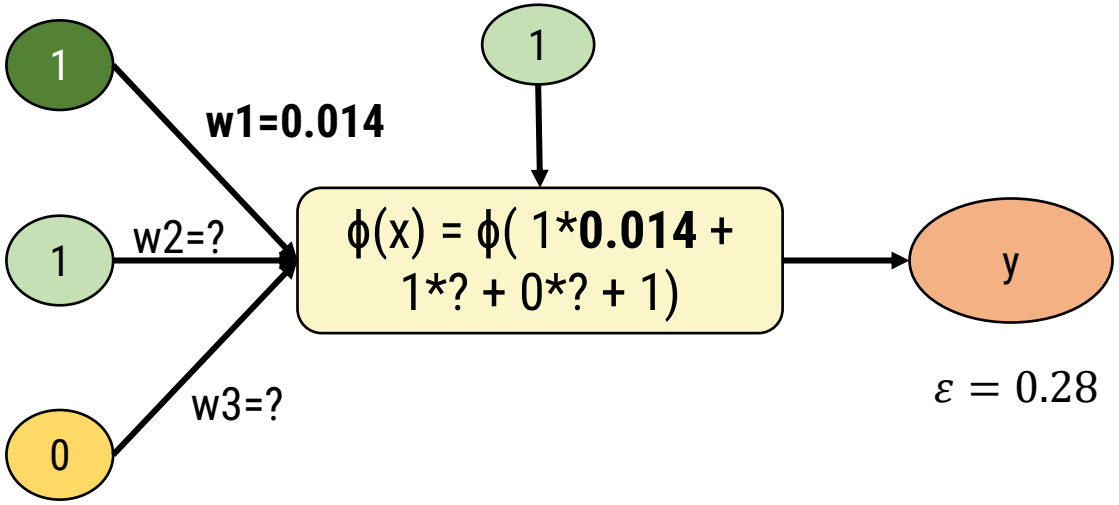
$$\text{new } w = w - \alpha(\hat{y} - y)x$$

$$new\ w1 = 0 - 0.05 * -0.28 * 1 = \mathbf{0.014}$$

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



**How to find the best weights?**

**4 – Backpropagate Error (Update the weights)**

With learning rate = 0.05

$$\text{new w} = \text{w} - \alpha(\hat{y} - y)x$$

$$new\ \text{w1} = 0 - 0.05 * -0.28 * 1 = 0.014$$

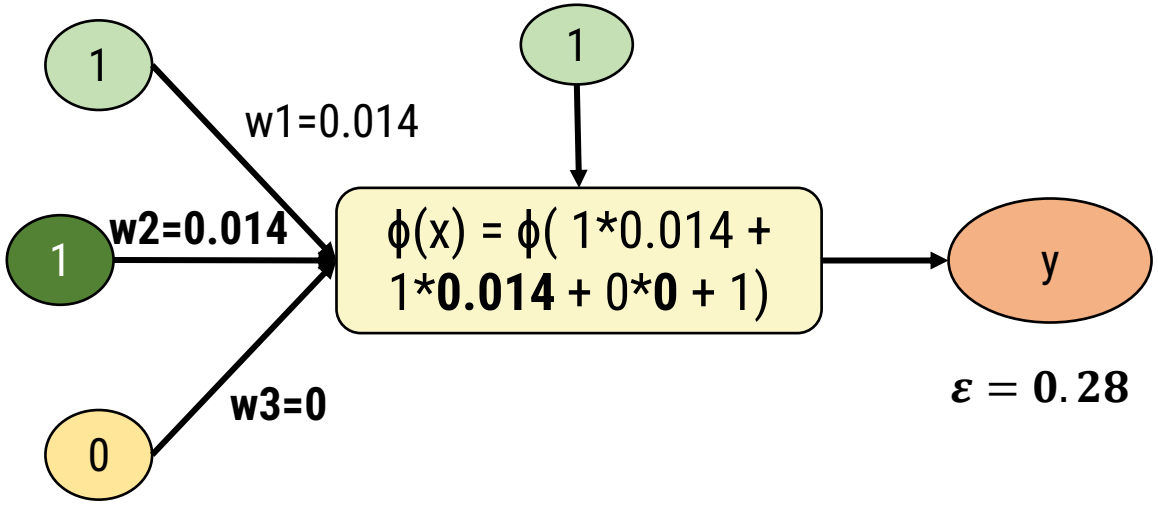
$$new\ \text{w2} = 0 - 0.05 * -0.28 * 1 = 0.014$$

$$new\ \text{w3} = 0 - 0.05 * -0.28 * 0 = 0$$

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**



# Neural Network in Text

**Run the model again:**

w1= 0.014
w2= 0.014
w3= 0.014

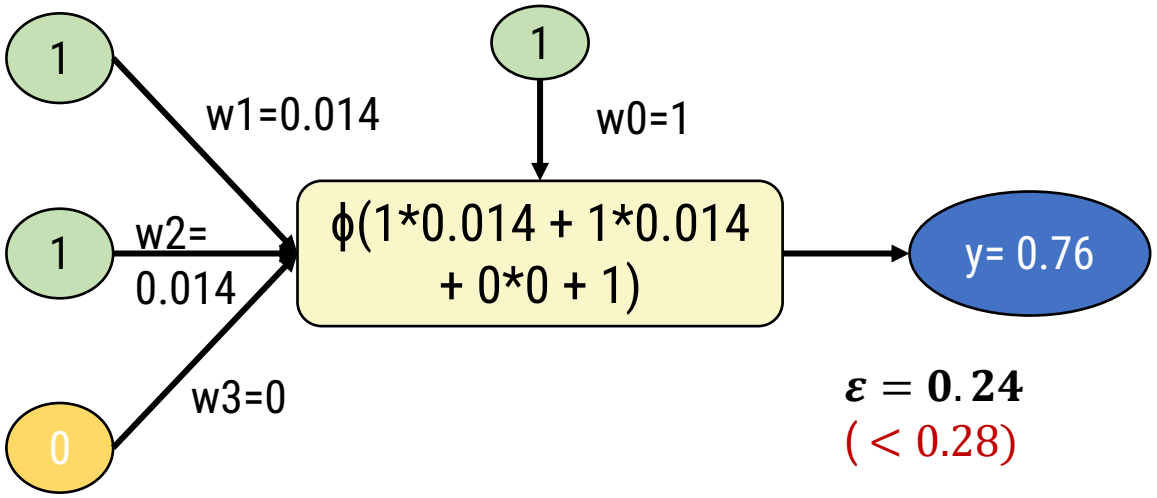
$$\phi(Review\ 1) = \frac{1}{1+e^{-(1*0.014+1*0.014+0*0+1)}}$$

$$= \frac{1}{1+0.36} = \mathbf{0.76}$$

**Task:**

Predict the sentiment of movie reviews using a neural network.

**Review 1:**




$\boldsymbol{\varepsilon = 0.24}$
$(< 0.28)$

**Error is now smaller!**

# Bibliography


- Dan Jurafsky and James H. Martin. **Speech and Language Processing** 3rd ed. https://web.stanford.edu/~jurafsky/slp3/
- Alammar, J., & Grootendorst, M. (2024). **Hands-On large language models: Language Understanding and Generation**. Sebastopol : O'Reilly Media. NOVA IMS Access: https://search.library.novaims.unl.pt/cgi-bin/koha/opac-detail.pl?biblionumber=97234

NOVA IMS Information Management School

# Thank you!

**Instituto Superior de Estatística e Gestão de Informação**
Universidade Nova de Lisboa