Politecnico di Bari

Dimensionality Reduction

Independent Component Analysis

we try to maximize the independence of the data. dimensionality reduction but looking at dependance of the data

Problem definition

Decomposing a mixed signal into independent sources

Example

Given: Mixed Signal

Goal: retrieve the original signals

Source1 News

Source2 Song

Common example

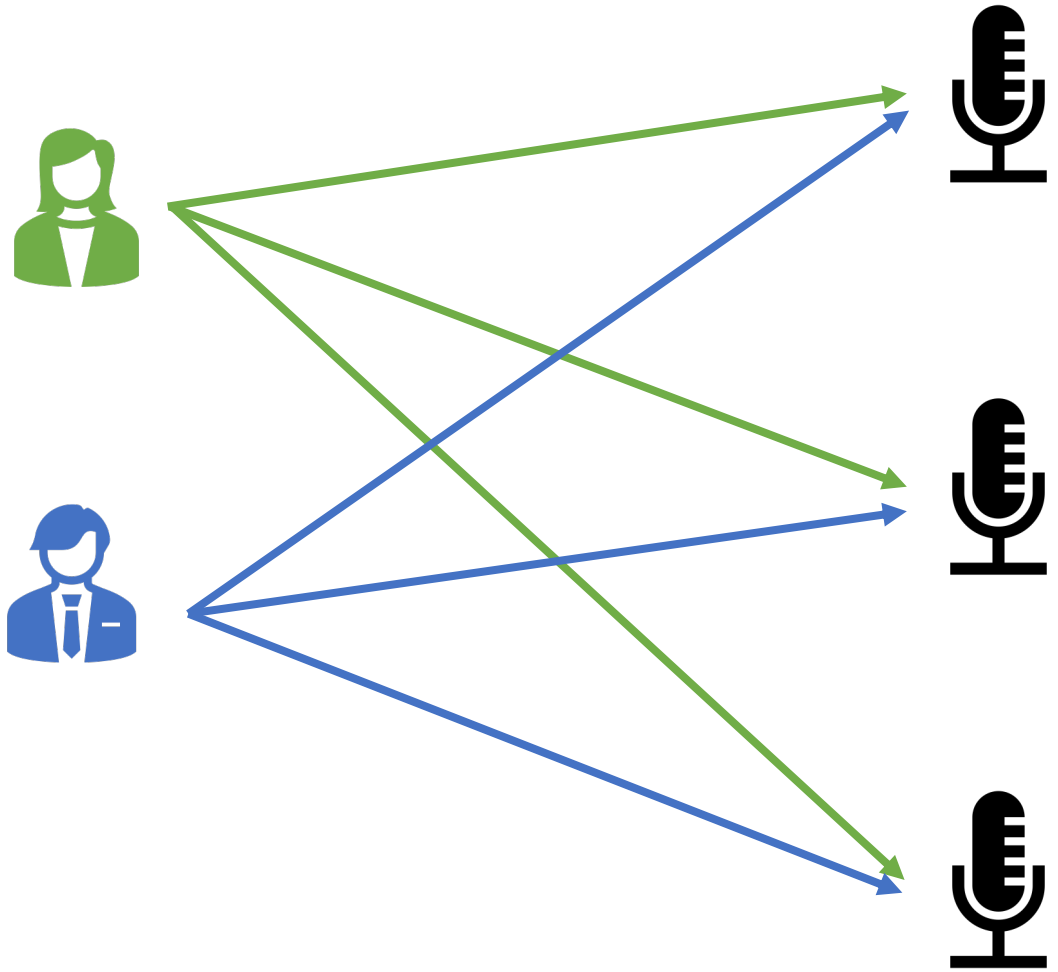
Cocktail party or Blind Source Separation (**BSS**) problem

Definitions: Statistical independence

In probability theory, saying that two events are **independent** means that the occurrence of one event makes it neither more nor less probable that the other occurs.

Examples of such independent random variables are the value of a dice thrown and of a coin tossed

Cocktail party problem



Problem

The microphones give us three recorded time signals.

We denote them with:

$x=(x_1(t), x_2(t), x_3(t))$.

x_1, **x_2**, and **x_3** are the amplitudes and **t** is the time index.

We denote the independent signals by

$s=(s_1(t),s_2(t));$

A - mixing matrix (3x2)

$$x_1(t) = a_{11}s_1 + a_{12}s_2$$

$$x_2(t) = a_{21}s_1 + a_{22}s_2$$

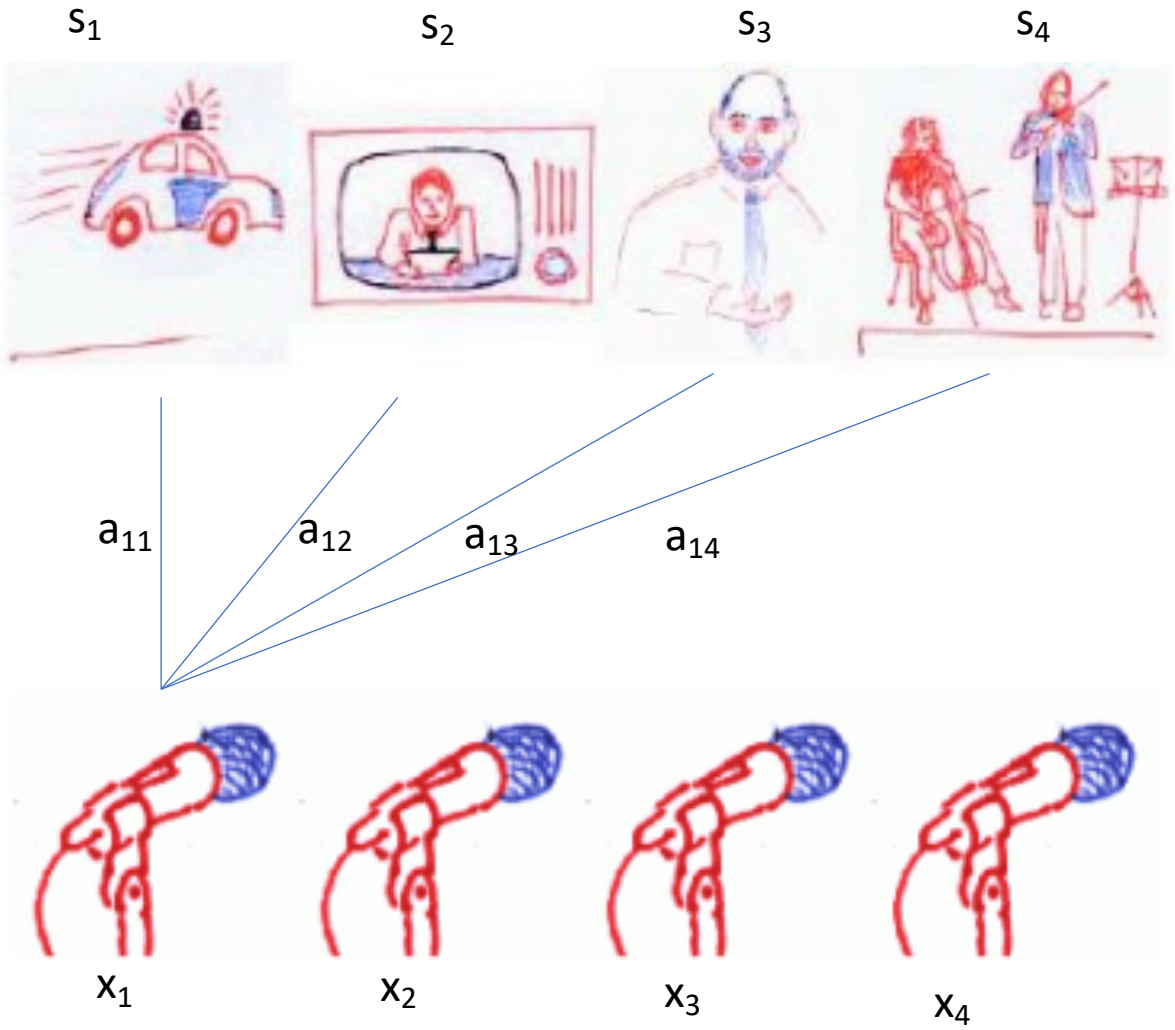
$$x_3(t) = a_{31}s_1 + a_{32}s_2$$

we assume the mixing is a linear combination of the two
we want to find aij and keeping independent as possible

a_{ij} are some parameters that depend on the distances of the microphones from the speakers.

It would be very good if we could estimate the two original speech signals *$s_i(t)$ using only the* recorded signals *$x_j(t)$. We need to estimate the a_{ij}.*, but it is enough to assume that *$s_1(t)$* and *$s_2(t)$, at each time instant t, are statistically independent. The main task is to transform the data s(x)=Ax to independent components, measured by function: $F(s_1,s_2)$*

ICA model



$$xi(t) = a_{i1} * s_1(t) + a_{i2} * s_2(t) + a_{i3} * s_3(t) + a_{i4} * s_4(t)$$

$i=1,\ldots,4.$

In vector-matrix notation, and dropping index t, this is

$$\mathbf{x} = \mathbf{A} * \mathbf{s}$$

A= Transformational matrix

Restrictions

In order to make ICA working some assumptions have to be made:

1. The independent components are assumed statistically *independent*
2. The independent components must have *non-Gaussian* distributions
3. For simplicity, we assume that the unknown mixing matrix is *squared*.

   The number of independent components is equal to the number of observed mixtures.

just for simplicity

Building bricks (recap)

Observed event: $x = As$ A:= mixing matrix

Observed streams $\{x^{(i)}; i = 1, \ldots, m\}$

Original sources: $s^{(i)} \in \mathbb{R}^k$

$s_j^{(i)}$

Unmixing Matrix: $W = A^{-1}$

$s^{(i)} = W x^{(i)}$

Unmixing element: w_i^T

$s_j^{(i)} = w_i^T x^{(i)}$

$$\mathbf{W} = \begin{bmatrix} \text{—} & (w^{(1)})^T & \text{—} \\ & \vdots & \\ \text{—} & (w^{(k)})^T & \text{—} \end{bmatrix}$$

ICA Ambiguities

Ambiguous permutation:

Let $\mathbf{P}$ be a n x n permutation matrix: $\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\mathbf{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

If z is a vector, then $\mathbf{P} \cdot z$ is another vector that contains a permutation of z coordinates. Given only the $x(i)$'s, there is no way to distinguish between $\mathbf{W}$ and $\mathbf{P} \cdot \mathbf{W}$ ($s = \mathbf{W} \cdot x$).

$$\mathbf{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix}$$

$$x = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

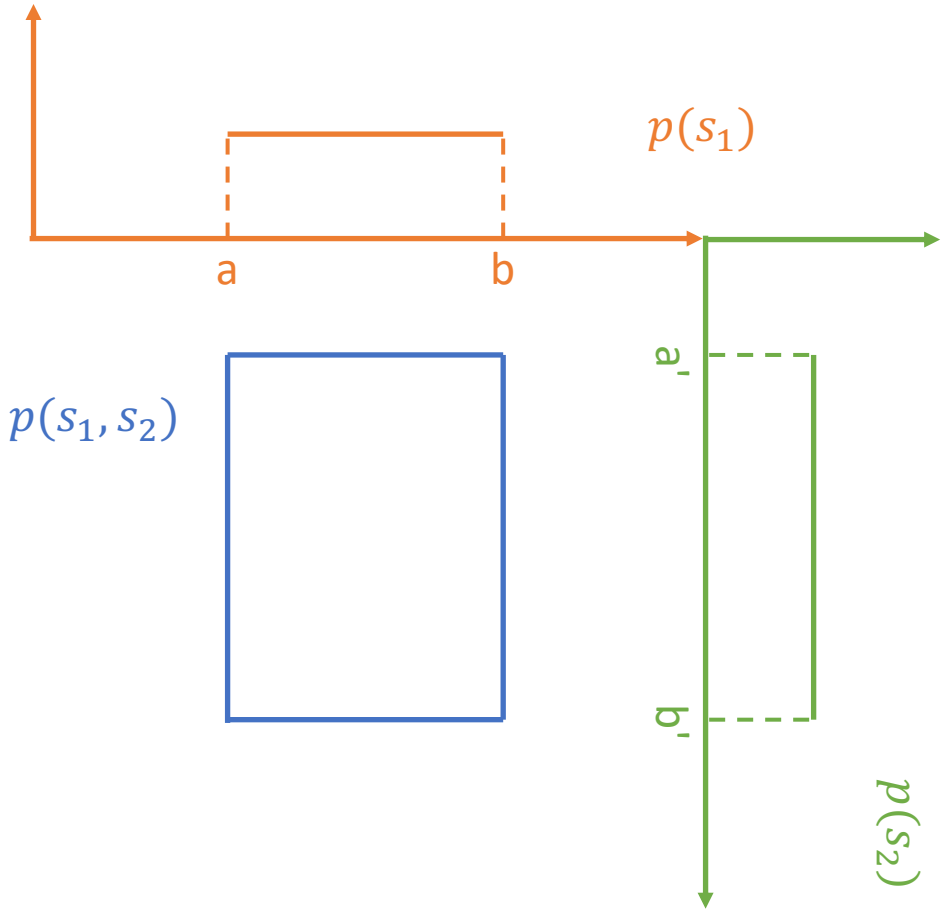
$$W \cdot x = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 19 \\ 24 \end{bmatrix}$$

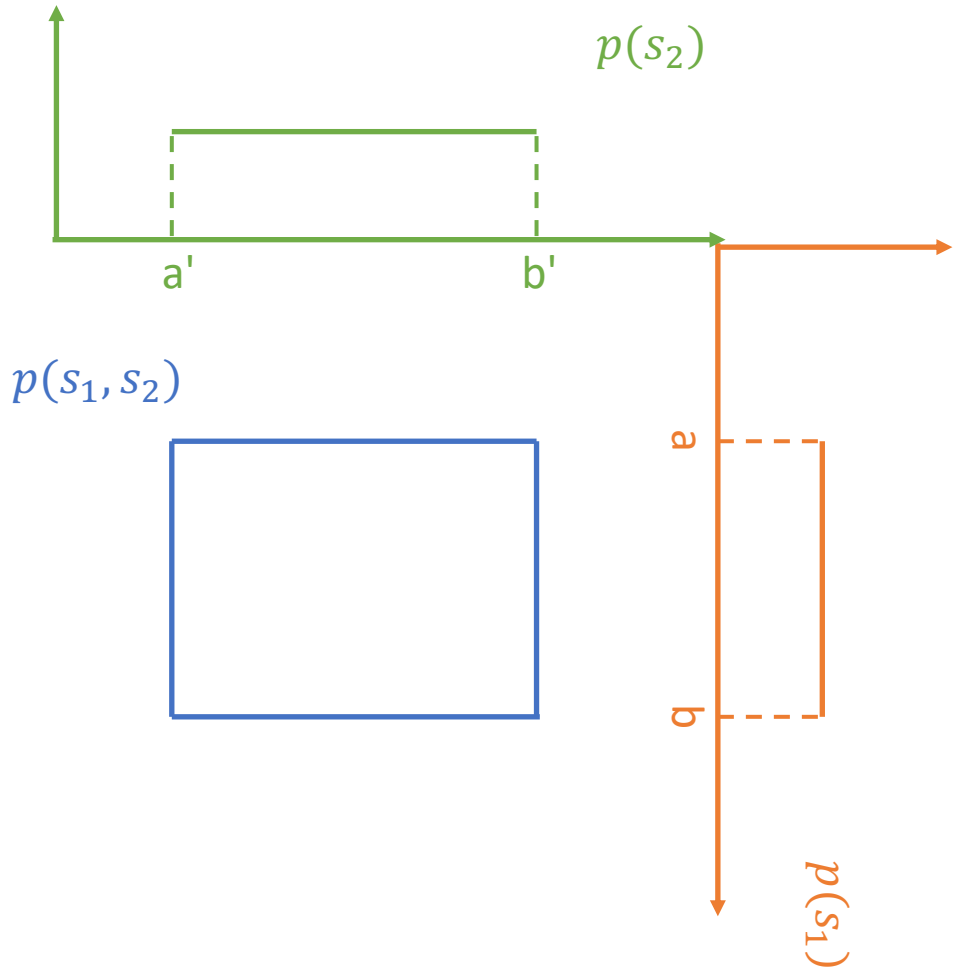
$$P \cdot W \cdot x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 3 & 5 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 24 \\ 19 \end{bmatrix}$$

we cant identify the explicit singer. we know there are two different singer but we could not say it's the first or the second.

we cant identify s1 and s2 because the joint probability even in permutation are the same

ICA Ambiguities - Permutations





ICA Ambiguities

Ambiguous scale:

We said that:

$$x^{(i)} = A \cdot s^{(i)}$$

There is no way to recover the correct scaling of the w_i's. For instance, if A was replaced with $2 \cdot A$, and every $s(i)$ were replaced with

$0.5 \cdot s(i)$, then we obtain

$$x^{(i)} = 2A \cdot (0.5)s^{(i)}$$

The observed x is still the same.

Gaussian Ambiguity

Why sources have to be non-Gaussian?

Let us suppose our sample shows a Gaussian distribution:

$$s \sim \mathcal{N}(0, I)$$

What we observe is $x = A \cdot s$

The distribution of x will also be Gaussian with zero mean and variance:

$$\mathbb{E}[xx^T] = \mathbb{E}[Ass^T A^T] = AA^T$$

Gaussian Ambiguity

Now let R be an orthogonal matrix such that:

$$RR^T = R^T R = I$$
$$A' = AR$$

If A' was used as mixing matrix we would observe x' $= A$' $\cdot s$.

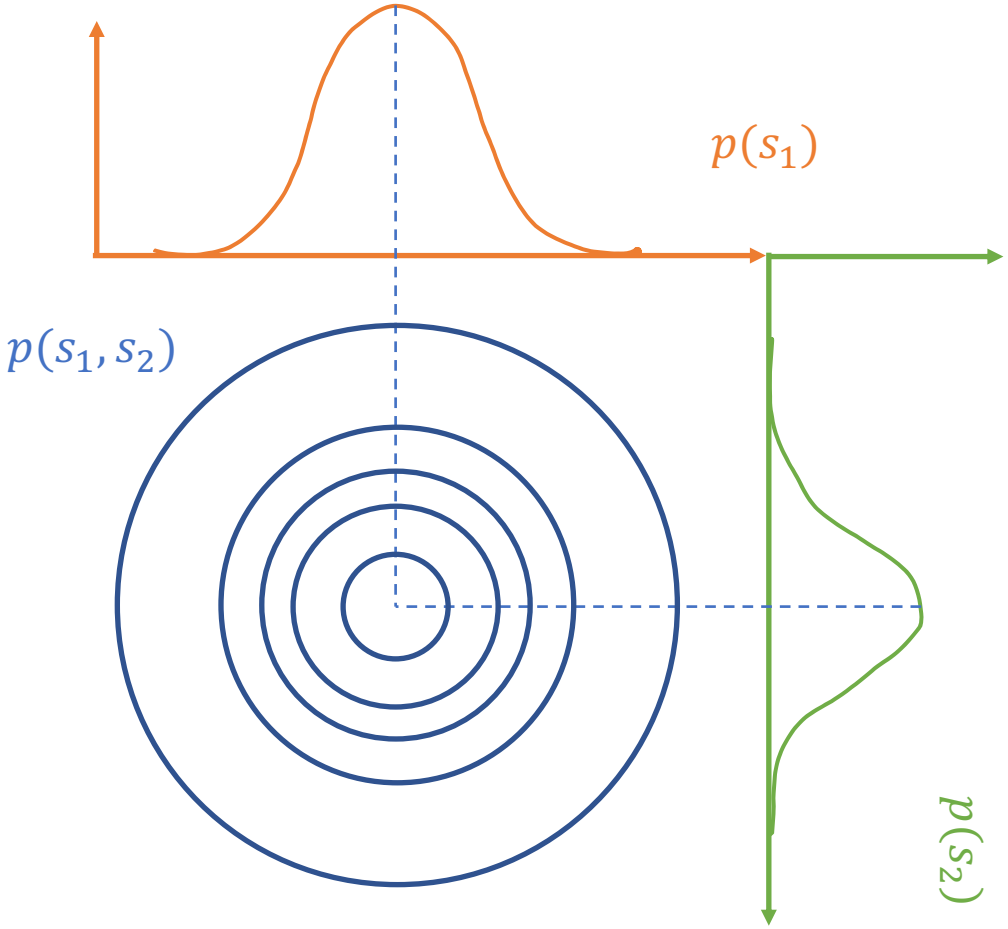
Also x' will show a Gaussian distribution with zero mean and variance:

$$\mathbb{E}[x'(x')^T] = \mathbb{E}[A'ss^T(A')^T] = \mathbb{E}[ARss^T(AR)^T] = ARR^T A^T = AA^T$$

Also in this case we will observe data in $\mathcal{N}(0, AA^T)$

If sources are Gaussian there is no way to know if they were mixed with A or A'.

ICA Ambiguities – Gaussian Distributions



Central Limit Theorem (practical remind)

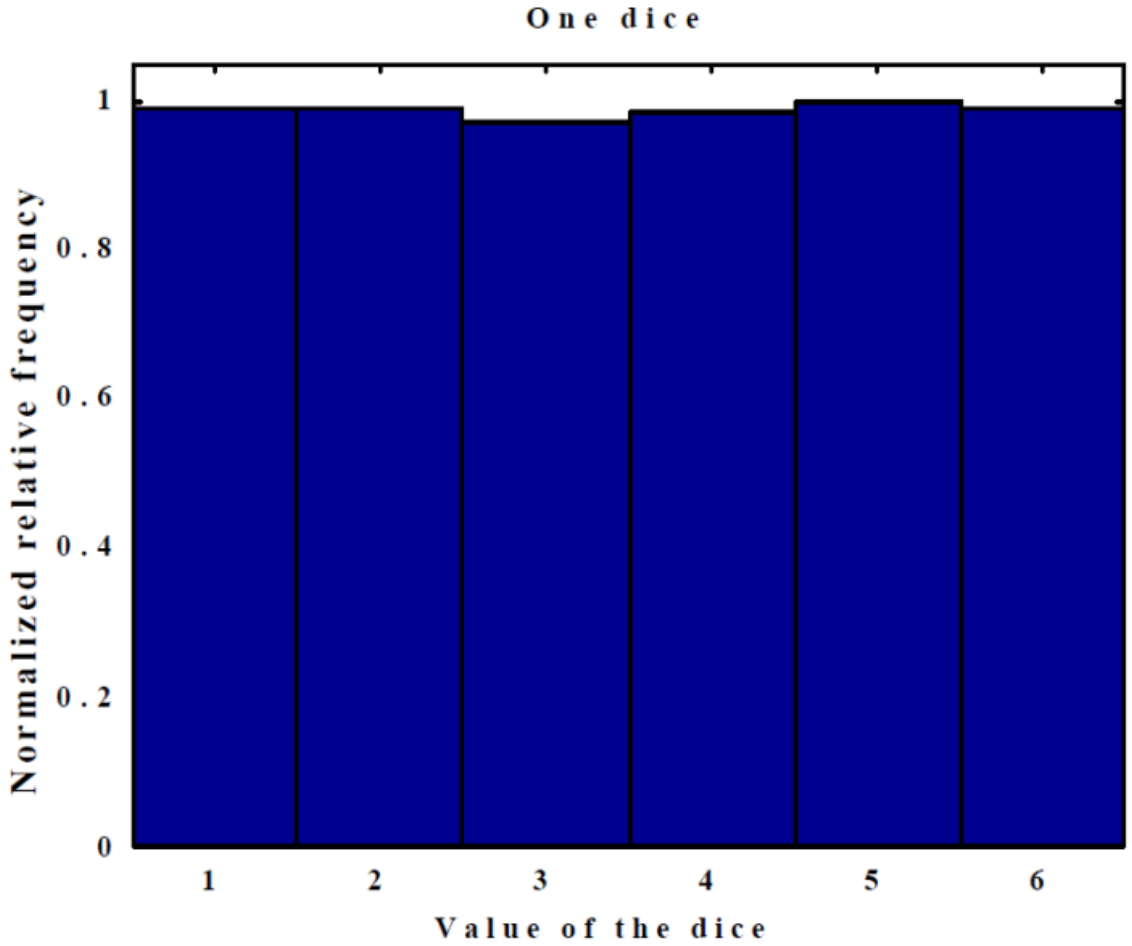
Central limit theorem states that linear combination of given random variables is more Gaussian as compared to the original variables themselves.

Example:

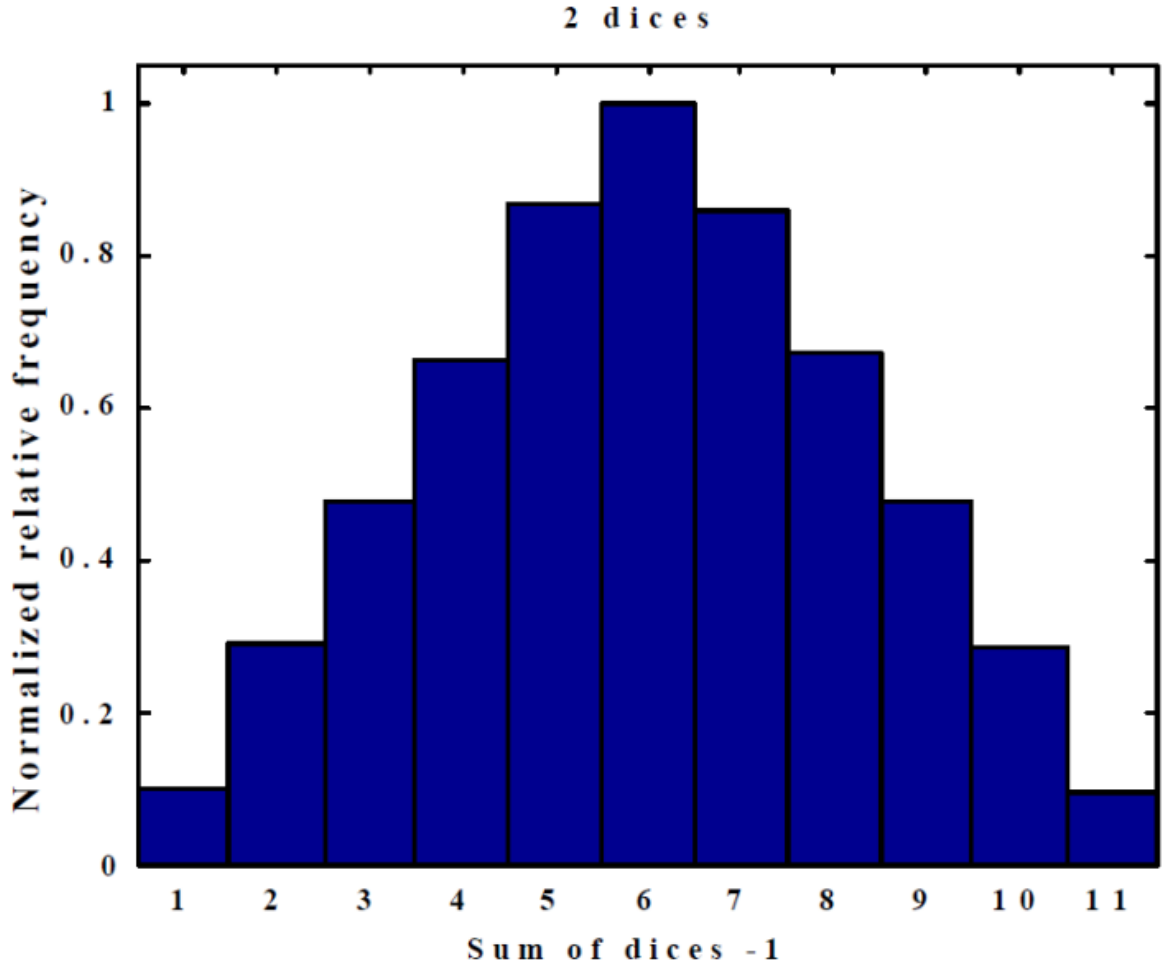
Repeated throwing of nonbiased dices, with the result given by the sum of the dice faces.

- For one dice the distribution is uniform;
- For two dices the sum is piecewise linearly distributed;
- For a set of *k* dices, the distribution is given by a k-order polynomial distribution;
- With the increase of *k*, the distribution of the sum tends towards a more Gaussian one.

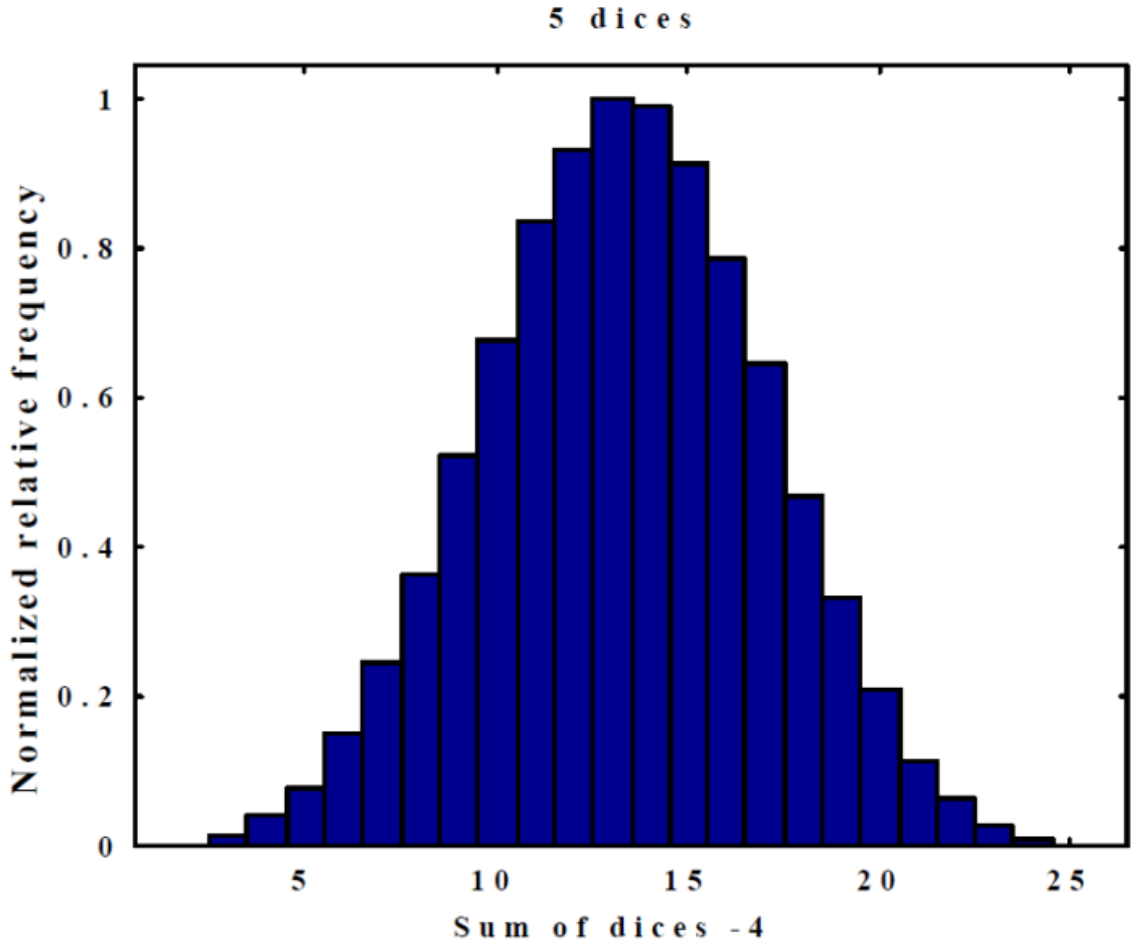
One dice sum distribution



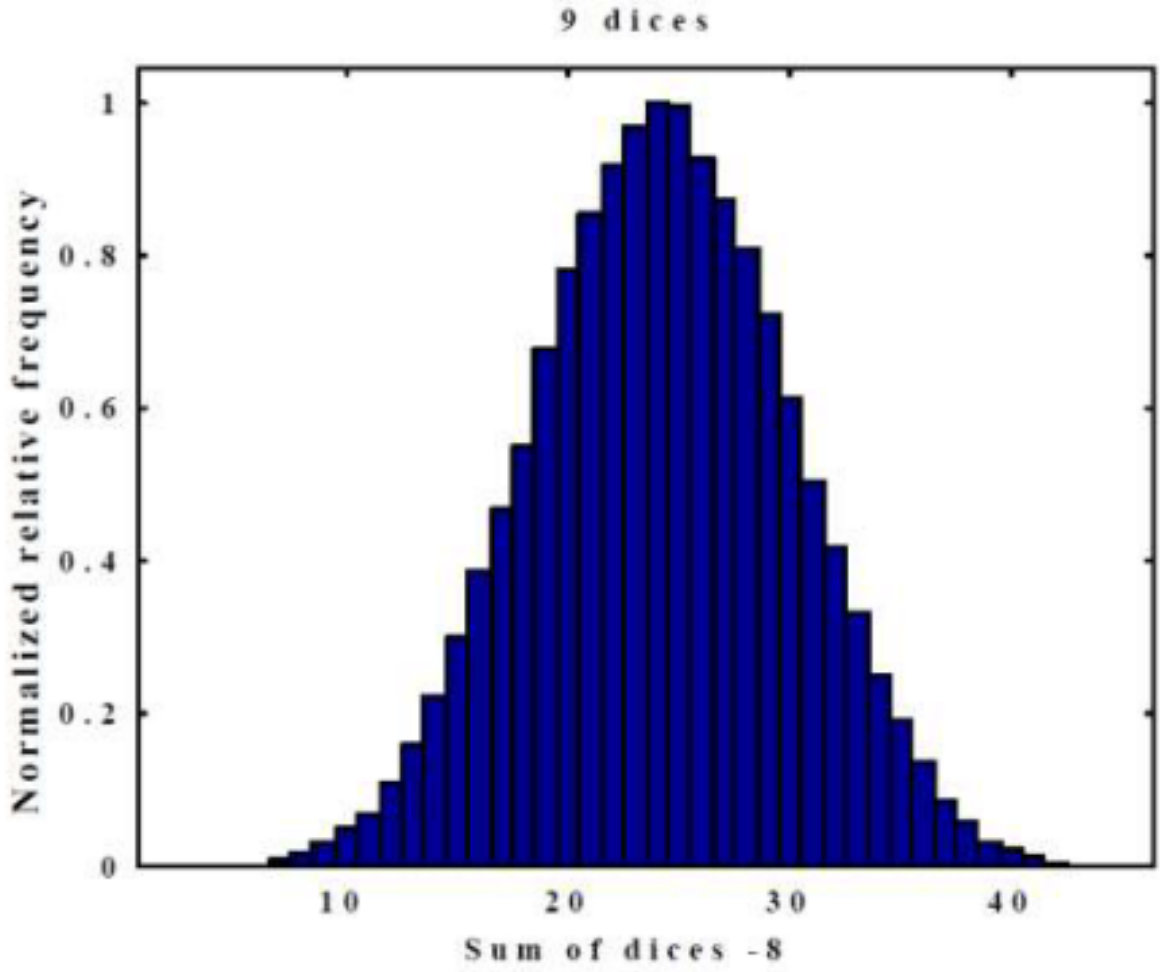
Two dices sum distribution



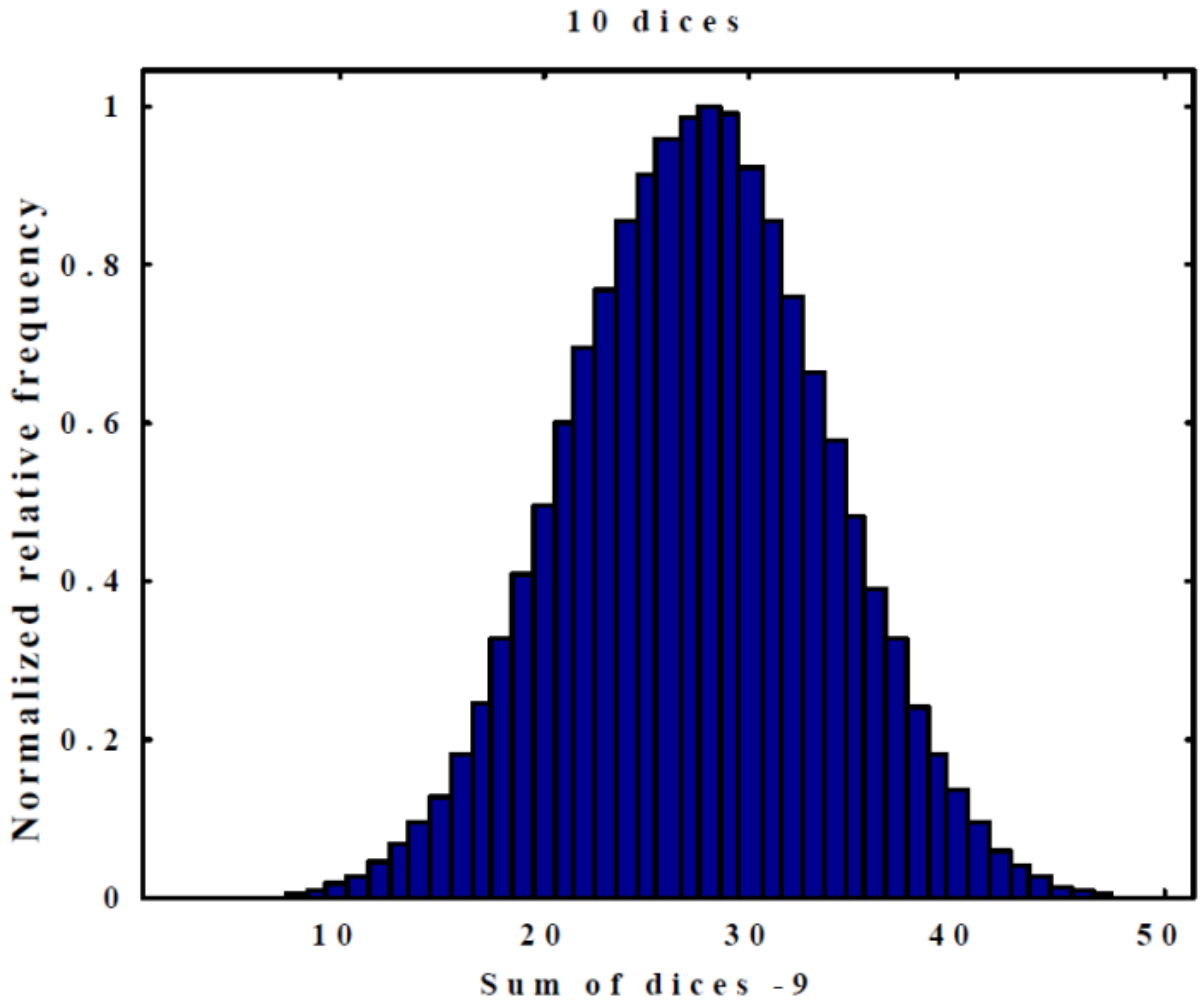
Three dices sum distribution



Nine dices sum distribution



Ten dices sum distribution



Why does ICA require signals to be non-Gaussian ?

Central limit theorem states that linear combination of given random variables is more Gaussian as compared to the original variables themselves.

So, if we have two independent Gaussian sources after mixing them, they become more Gaussian as central limit theorem state, which make it impossible to recover the original sources

Densities and linear transformations

Assuming that s could be substituted to $W \cdot x$ in density definition leads to **the following erroreous relation**: $p_x(x) = p_s(Wx)$

Let us see through an example the correct procedure to derive densities:

$$x = As \text{ with } s \in \mathbb{R}, x \in \mathbb{R}, A \in \mathbb{R}$$

$$s = Wx$$

$$s \sim Uniform[0,1] \Rightarrow p_s(s) = 1\{0 \leq s \leq 1\}$$

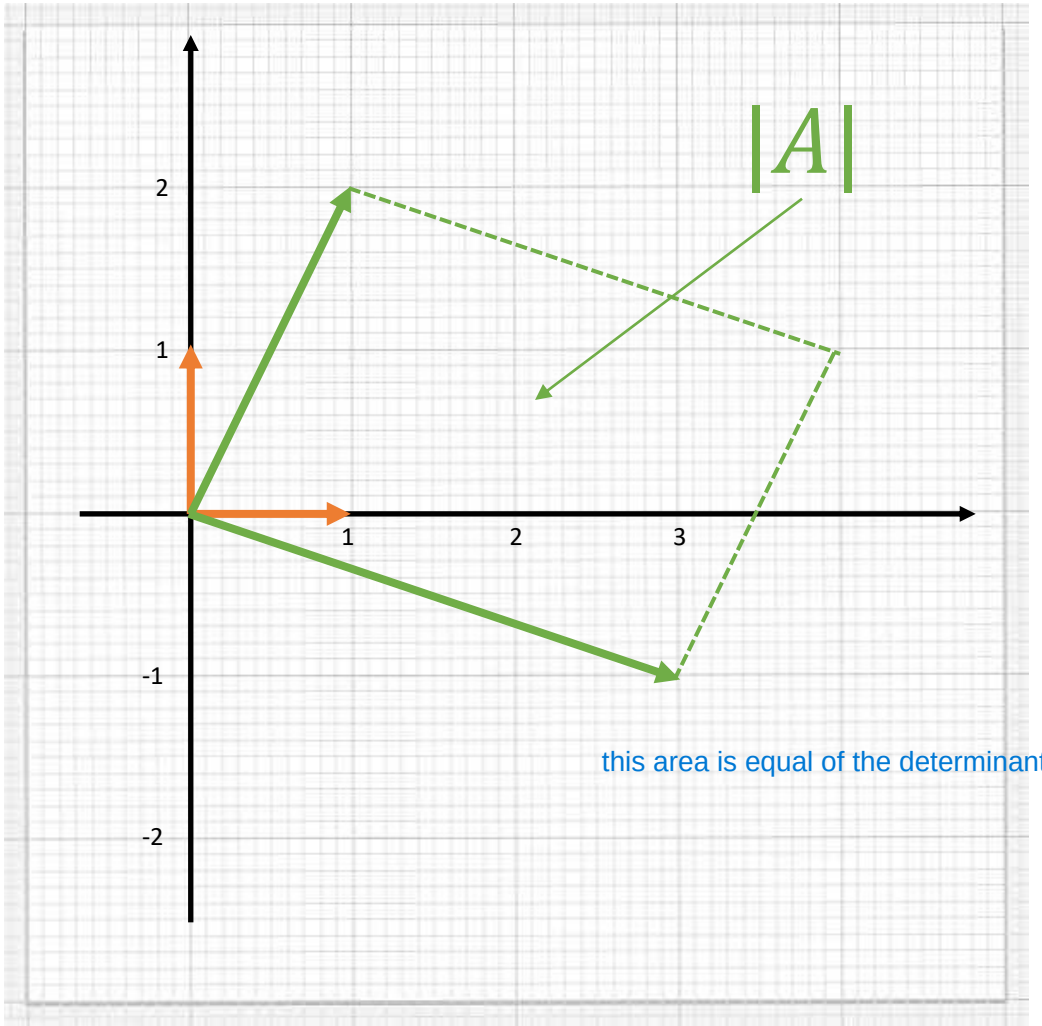
$$\text{If } A = 2; x = 2s \Rightarrow x \sim Uniform[0,2]$$

$$p_x(x) = (0.5)1\{0 \leq x \leq 2\}$$

$$p_x(x) = p_s(Wx)|W|$$

ps(Wx)|A^-1|=ps(Wx)|A|

$$p_x(x) = p_s(W \cdot x) \cdot |W|$$



this area is equal of the determinant of A |A|

$$A = \begin{bmatrix} 1 & 3 \\ 2 & -2 \end{bmatrix} \quad s_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad s_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$x_1 = A \cdot s_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad x_2 = A \cdot s_2 = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

$$p_x(x) = \frac{p_s(W \cdot x)}{|A|} = p_s(W \cdot x) \cdot |W|$$

ICA Algorithm

We can define the joint distribution as product of the marginal distributions:

$$p(s) = \prod_{i=1}^{n} p_s(s_i)$$

Now we can take advantage of the former derivation:

$$p(x) = \prod_{i=1}^{n} p_s(w_i^T x) \cdot |W|$$

ICA Algorithm

Now we will maximize $p(x)$. What does it mean?

We want to find the value for the unknown parameter that maximize the probability of obtaining the observed values.

But this is the definition of likelihood, hence we can compute the log likelihood of $p(x)$:

$$l(W) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \log p_s(w_j^T x^{(i)}) + \log|W| \right)$$

Usually a high-Kurtosis model for p_s is assumed

$$p_s(s) = (\ 1\ - \tanh(s)^2) = (1 - \tanh(w_j^T x^{(i)})^2)$$

Sigmoid-based $p_s(s)$

- $p_s(s)$ is such that $P(s) = \int_{s_0}^{s_1} p_s(s)ds \in [0, \dots, 1]$
- We know that $sigmoid(s) \in [0, \dots, 1]$
- Idea: $P(s) = sigmoid(s) = g(s)$ then $p_s(s) = g'(s)$

$$l(W) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \log g'(w_j^T x^{(i)}) + \log|W| \right)$$

ICA Algorithm

not so important

Now we know from linear algebra that the gradient of a determinant (in case W is invertible) is equal to:

$$\nabla_W = |W|(W^{-1})^T$$

Now we can derive a stochastic gradient learning rule:

$$W := W + \alpha \left(\begin{bmatrix} 1 - 2g(w_1^T x^{(i)}) \\ 1 - 2g(w_2^T x^{(i)}) \\ \vdots \\ 1 - 2g(w_n^T x^{(i)}) \end{bmatrix} x^{(i)} + (W^T)^{-1} \right)$$

Politecnico di Bari

Dimensionality Reduction

Kernel PCA

idea behind -> pca reduce the dimension to have a better cluster, but is linear, if we use pca to work with non linearly separable data is not effective.
in the case of non linearly we can use the kernel trick.

PCA in a nutshell

PCA finds principal axes by working on the covariance matrix:

$$Cov = \frac{1}{m} \sum_{i=1}^{m} x_i x_i^T$$

Cov is positive definite, and can be diagonalized with non-negative eigenvalues.

$$\lambda v = Cov\ v$$

projecting datapoint in a new space hoping they gets more separable

PCA in terms of dot products

$$Cov\ v = \frac{1}{m}\sum_{i=1}^{m} x_i x_i^T v = \lambda v$$

From the former we can compute v:

$$v = \frac{1}{m\lambda}\sum_{i=1}^{m} x_i x_i^T v$$

x^T v=v^T xi

$$= \frac{1}{m\lambda}\sum_{i=1}^{m} (x_i \cdot v) x_i$$

All the solutions v with $\lambda \neq 0$ can be expressed as $v = \sum_{i=1}^{m} \alpha_i x_i$

Kernel PCA

If we map our data in another space $x \in \mathbb{R}^n \rightarrow \Phi(x) \in \mathbb{R}^d$ we can write the covariance matrix as:

$$Cov = \frac{1}{m} \sum_{i=1}^{m} \Phi(x_i)\Phi(x_i)^T$$

When we consider its eigenvectors we have:

$$Cov\ v = \frac{1}{m} \sum_{i=1}^{m} \Phi(x_i)\Phi(x_i)^T v = \lambda v$$

$$v = \frac{1}{\lambda m} \sum_{i=1}^{m} \Phi(x_i)\Phi(x_i)^T v = \frac{1}{\lambda m} \sum_{i=1}^{m} (\Phi(x_i) \cdot v)\Phi(x_i)^T$$

Kernel PCA

All the solutions v_l with $\lambda \neq 0$ can be expressed as:

$$v_l = \sum_{i=1}^{m} \alpha_{li} \cdot \Phi(x_i)$$

This means that *finding the eigenvectors is equivalent to finding the alpha coefficients*.

If we substitue the result in the second last equation of the previous slide:

$$Cov \cdot v_l = \frac{1}{m} \cdot \sum_{i=1}^{m} \Phi(x_i) \cdot \Phi(x_i)^T \cdot \left(\sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_j) \right) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_j)$$

Kernel PCA

$$Cov \cdot v_l = \frac{1}{m} \cdot \sum_{i=1}^{m} \Phi(x_i) \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_i)^T \cdot \Phi(x_j) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_j)$$

Kernel function: $k(x_i, x_j) = \Phi(x_i)^T \cdot \Phi(x_j) \in \mathbb{R}$

$$Cov \cdot v_l = \frac{1}{m} \cdot \sum_{i=1}^{m} \Phi(x_i) \cdot \left(\sum_{j=1}^{m} \alpha_{lj} \cdot k(x_i, x_j) \right) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_j)$$

Kernel PCA

$$Cov \cdot v_l = \frac{1}{m} \cdot \sum_{i=1}^{m} \Phi(x_i) \cdot \left(\sum_{j=1}^{m} \alpha_{lj} \cdot k(x_i, x_j) \right) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_j)$$

We can multiply by a data point $\Phi(x_k)^T$ that let us express the equation as composed by the same kernel value

$$Cov \cdot v_l = \frac{1}{m} \cdot \sum_{i=1}^{m} \Phi(x_k)^T \cdot \Phi(x_i) \cdot \left(\sum_{j=1}^{m} \alpha_{lj} \cdot k(x_i, x_j) \right) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \Phi(x_k)^T \cdot \Phi(x_j)$$

$$Cov \cdot v_l = \frac{1}{m} \cdot \sum_{i=1}^{m} k(x_k, x_i) \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot \mathrm{k}(x_i, x_j) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot k(x_k, x_j)$$

Kernel PCA

$$\frac{1}{m} \cdot \sum_{i=1}^{m} k(x_k, x_i) \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot k(x_i, x_j) = \lambda_l \cdot \sum_{j=1}^{m} \alpha_{lj} \cdot k(x_k, x_j)$$

Can be rewritten in matrix notation as

$$K^2 \cdot a_l = m \cdot \lambda_l \cdot K \cdot a_l$$

with $K \in \mathbb{R}^{d \times d}$. Since K is invertible we have

$$K \cdot a_l = m \cdot \lambda_l \cdot a_l$$

Kernel PCA

Now the equation can be normalized

$$K\alpha = \lambda\alpha$$

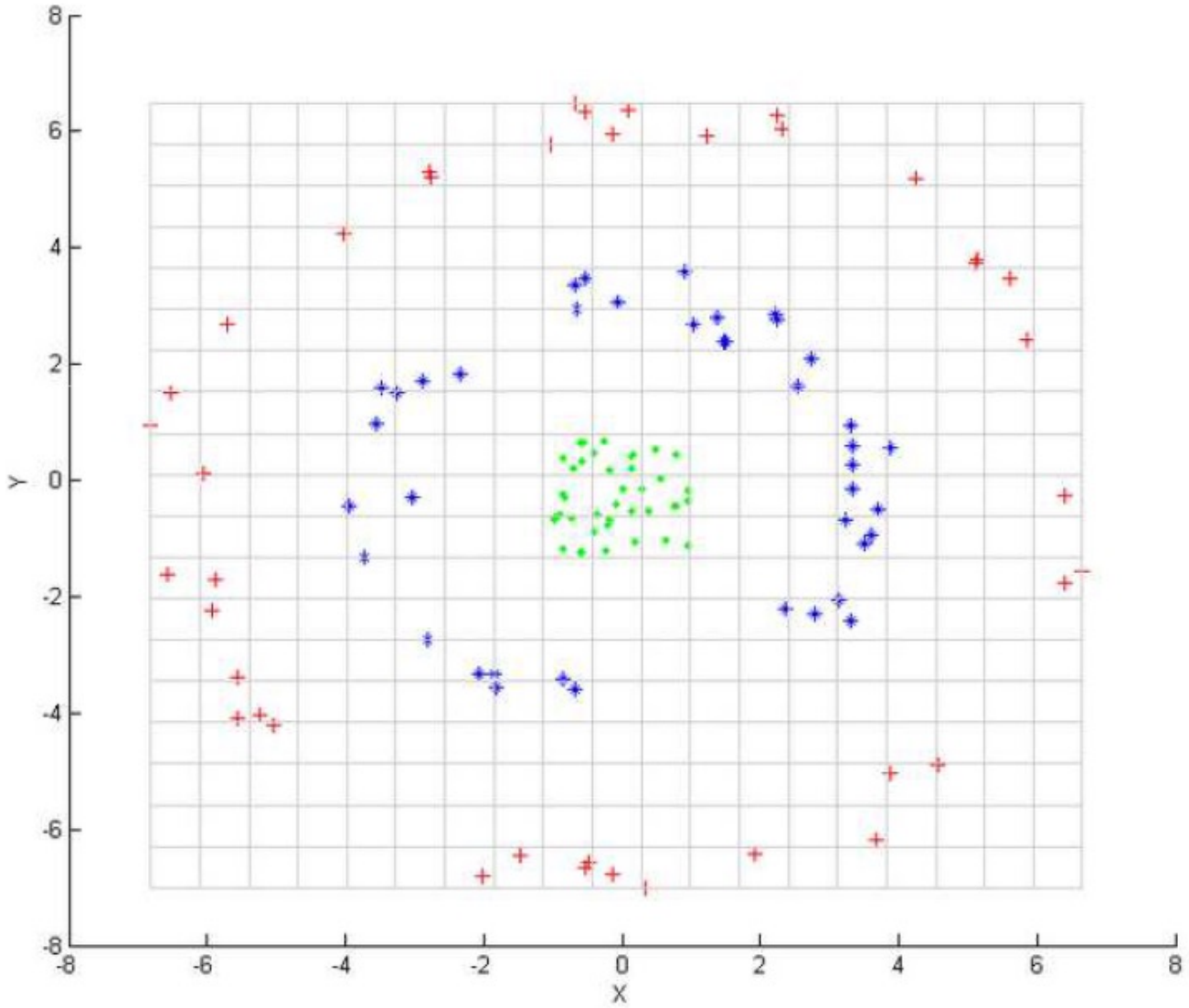
And we can obtain the same formulation as before defining:

- The kernel matrix $K\ m\ x\ m$ in which the $ij\text{-}th$ element is $K(x_i, x_j)$
- The vector $\alpha\ m\ x\ 1$ whose $j\text{-}th$ element is α_j

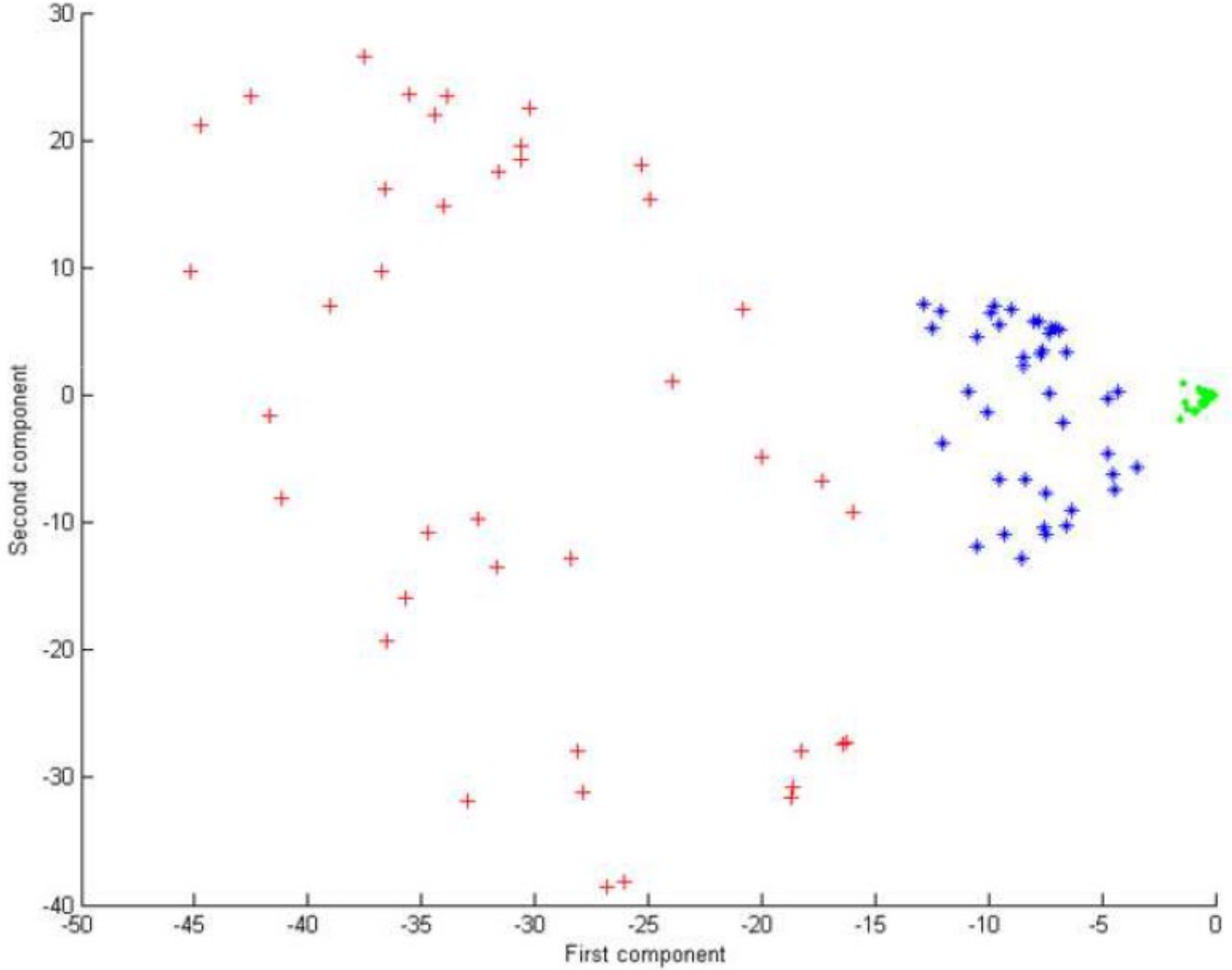
Now we can compute the principal components in the new space by:

$$\Phi(x)^T v^k = \sum_{i=1}^{m} \alpha_i^k K(x_i, x)$$

Kernel PCA – Example 1



Kernel PCA – Example 2



Kernel PCA – Denoising images

Original data

Data corrupted with Gaussian noise

Result after linear PCA

Result after kernel PCA, Gaussian kernel

Politecnico di Bari

Dimensionality Reduction

Embeddings

Vector Space Models

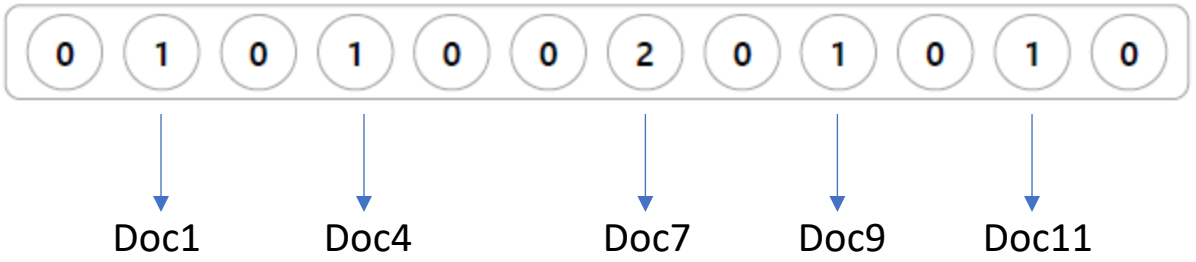
Represent an item (e.g., word) as a vector of numbers.

banana

Vector Space Models

Represent an item (e.g., word) as a vector of numbers.

banana

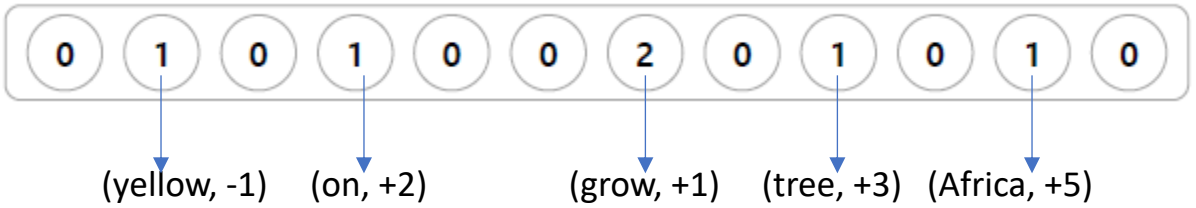


The vector can correspond to documents in which the word occurs.

Vector Space Models

Represent an item (e.g., word) as a vector of numbers.

banana



The vector can correspond to neighboring word context with respect to the following two documents.

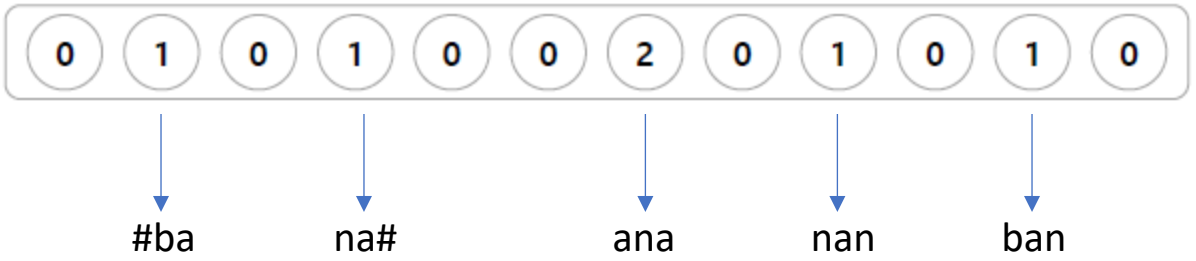
“Bananas grow best in soil that is rich and fertile”

“yellow bananas grow on trees in Africa”

Vector Space Models

Represent an item (e.g., word) as a vector of numbers.

banana



The vector can correspond to character trigrams in the word.

Notions of Relatedness

Comparing two vectors (e.g., using cosine similarity) estimates how similar the two words are. However, the notion of relatedness depends on what vector representation you have chosen for the words.

seattle similar to **denver**?

Because they are both cities.

seattle similar to **seahawks**?

Because “Seattle Seahawks”.

Important note: In previous slides I showed raw counts. They should either be normalized (e.g., using pointwise-mutual information) or (matrix) factorized.

Let’s consider the following example…

We have four documents,

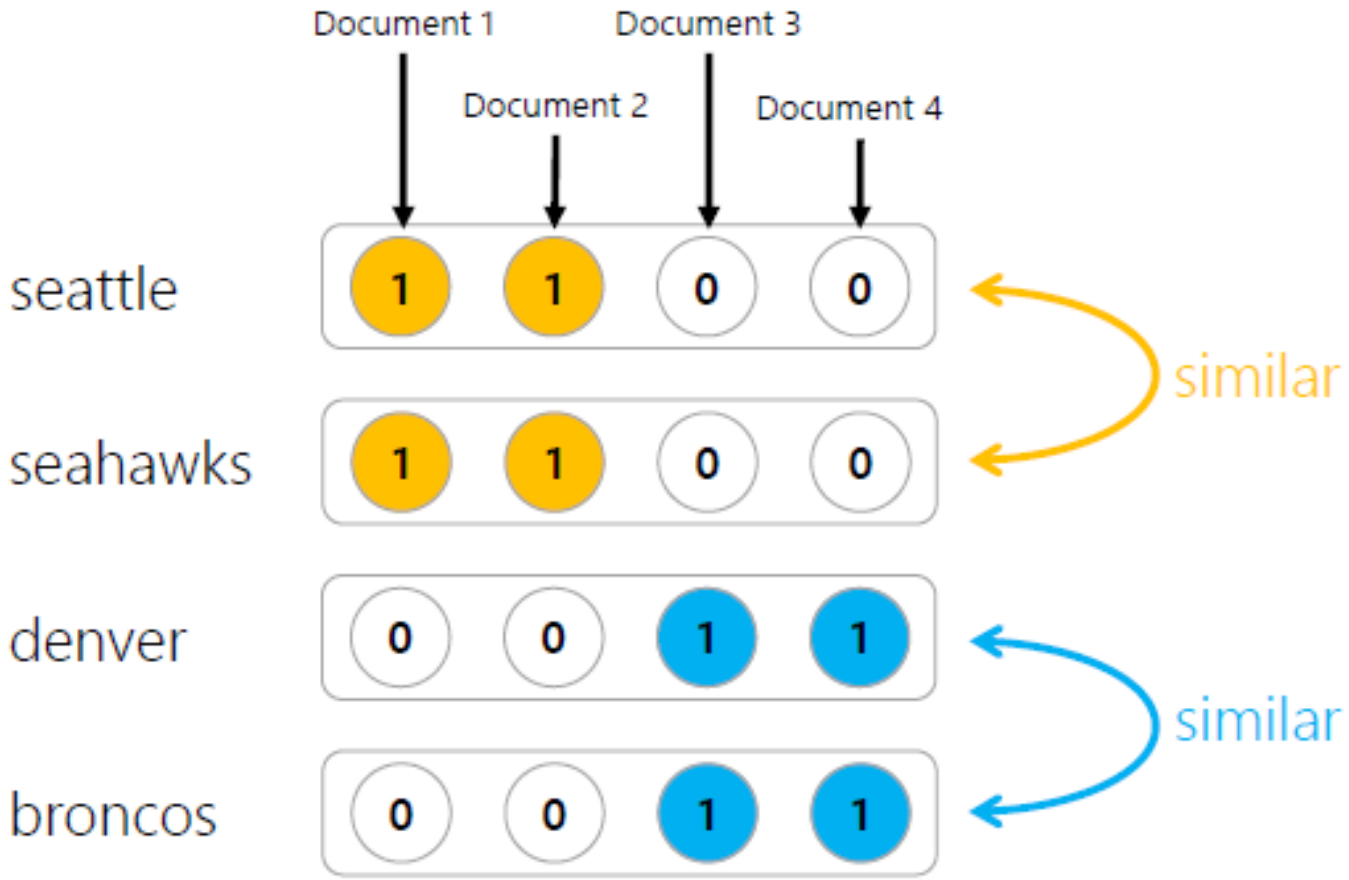
Document 1 : “seattle seahawks jerseys”

Document 2 : “seattle seahawks highlights”

Document 3 : “denver broncos jerseys”

Document 4 : “denver broncos highlights”

If we use document occurrence vectors…

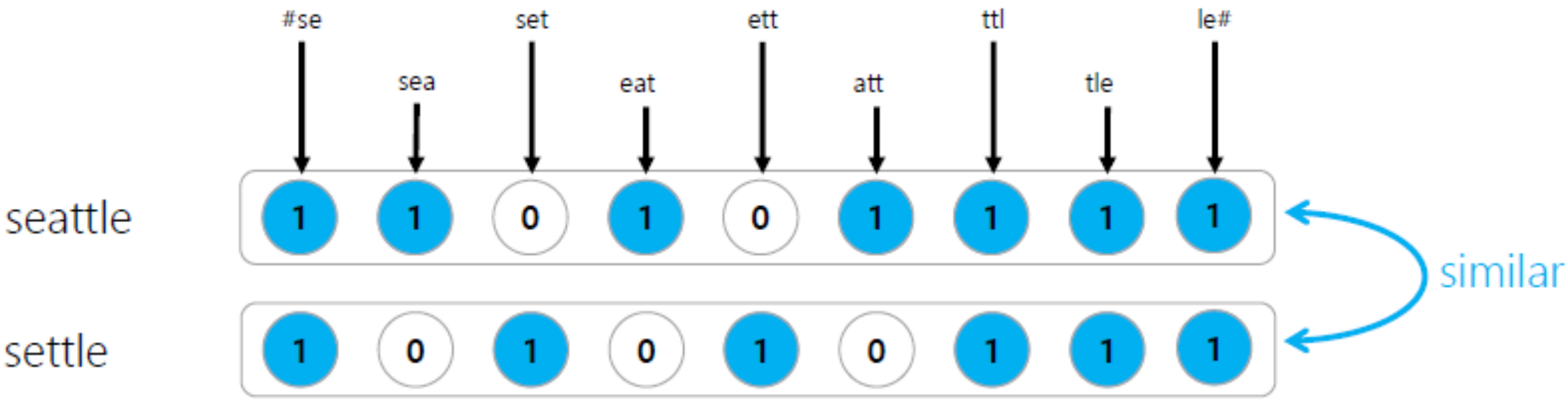


We refer to this notion of relatedness as **Topical** similarity.

If we use word context vectors…

We refer to this notion of relatedness as **Typical** (by-type) similarity

If we use character trigram vectors…



This notion of relatedness is similar to string edit-distance.

Word Analogy Task

man is to *woman* as *king* is to ____ ?

good is to *best* as *smart* is to ____ ?

china is to *beijing* as *russia* is to ____ ?

It turns out that the word-context based vector model we just learnt is good for such analogy tasks

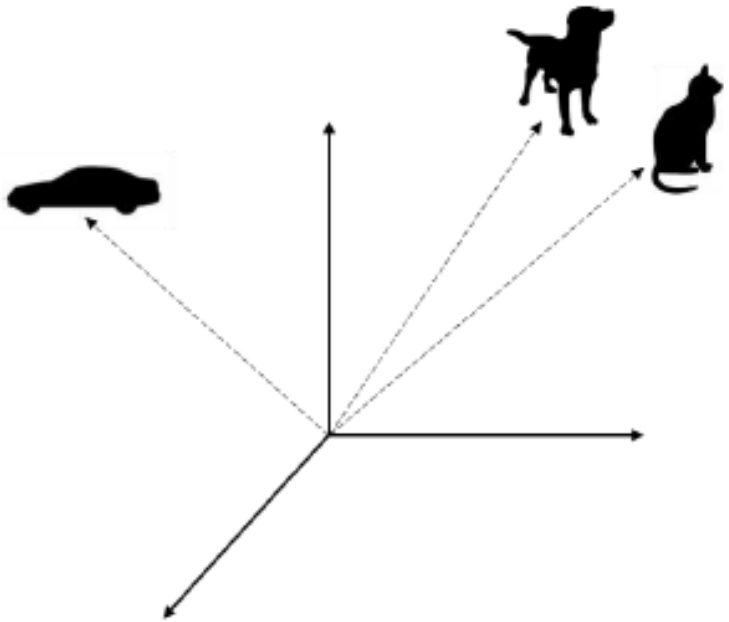
[king] – [man] + [woman] ≈ [queen]

Embeddings

The vectors we have been discussing so far are very high- dimensional (thousands, or even millions) and sparse. But there are techniques to learn lower-dimensional dense vectors for words using the same intuitions.

These dense vectors

are called embeddings.



Learning Dense Embeddings

Matrix Factorization

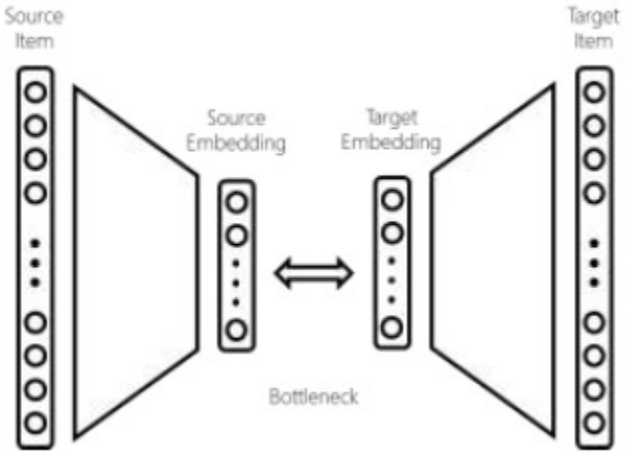
Factorize word-context matrix.

	$Context_1$	$Context_1$	....	Context $_k$
$Word_1$				
$Word_2$				
⋮				
$Word_n$				

E.g., LDA (Word-Document),
GloVe (Word-NeighboringWord)

Neural Networks

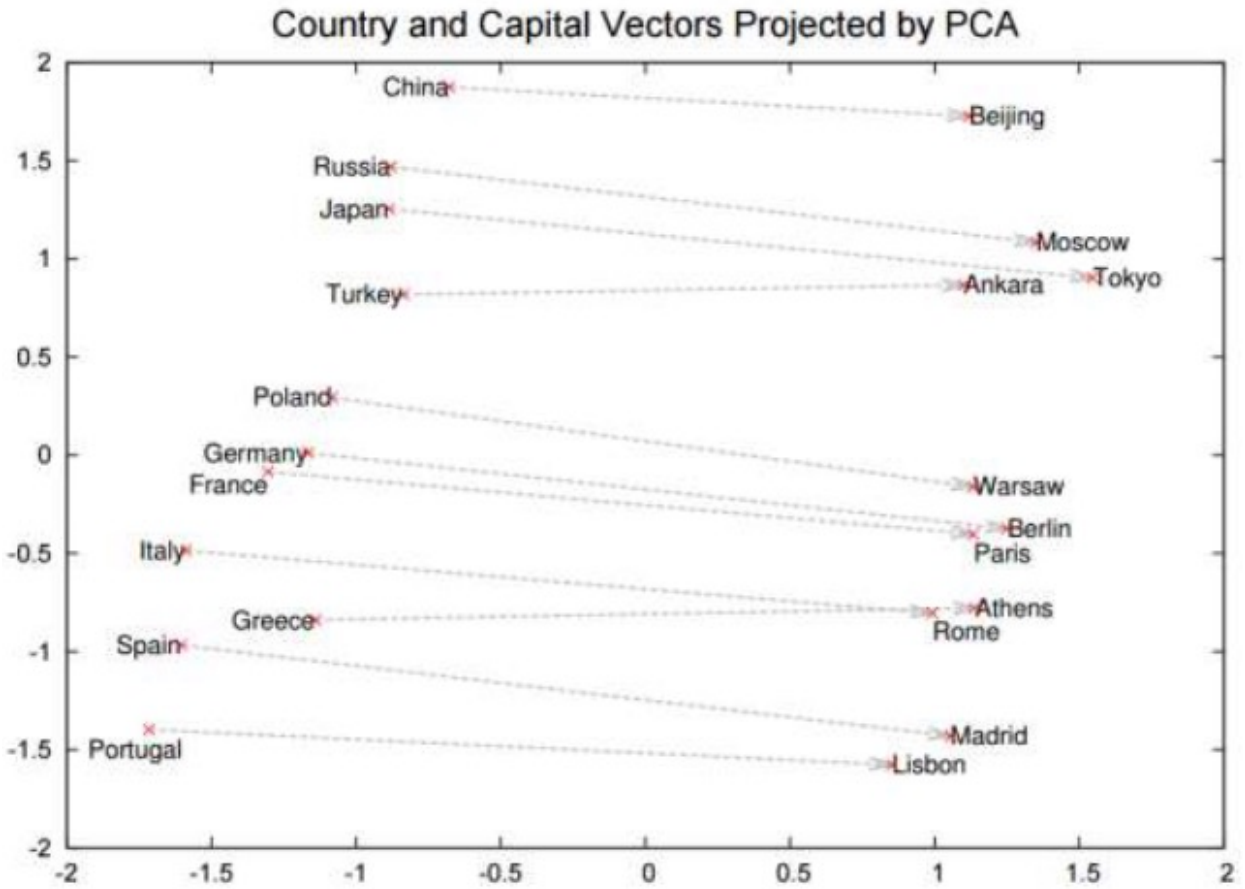
A neural network with a bottleneck, word and context as input and output respectively.



E.g., Word2vec (Word-NeighboringWord)

How do word analogies work?

Visually, the vector
{china -> beijing}
turns out to be almost parallel
to the vector
{russia -> moscow}.



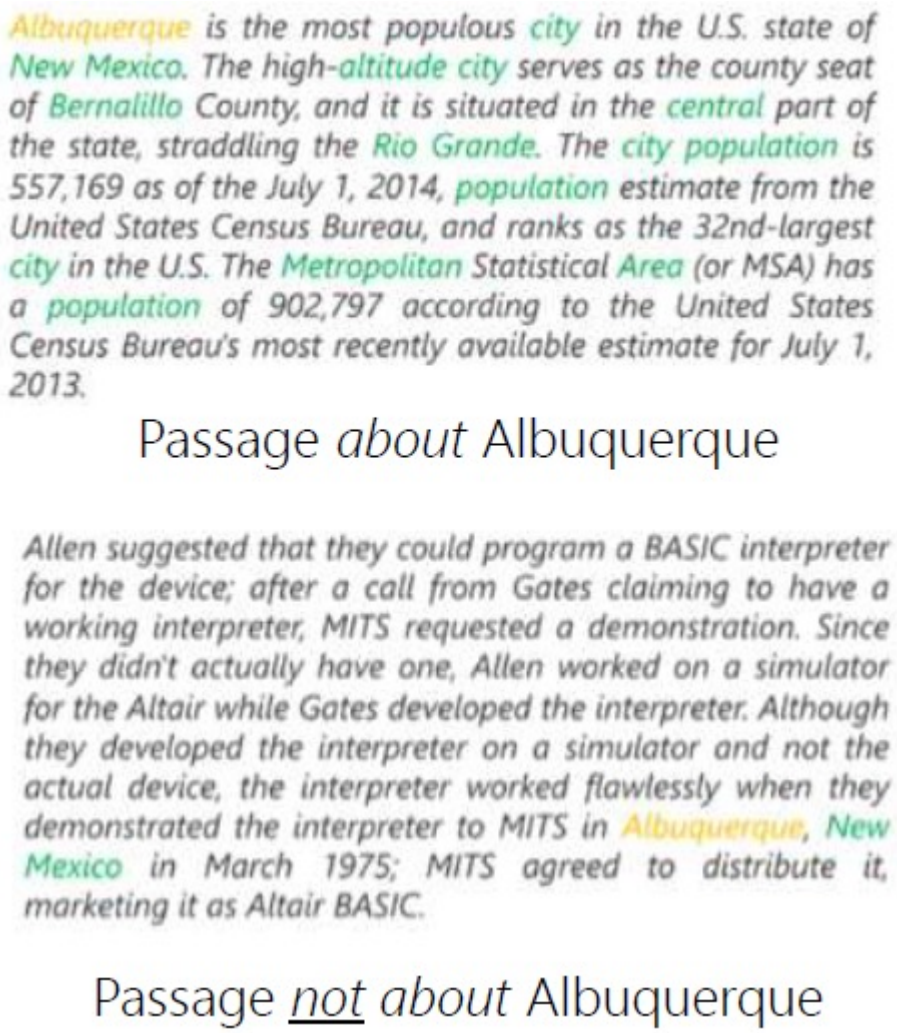
Word embeddings for Document Ranking

Traditional IR uses Term matching,

→ # of times the doc says *Albuquerque*

We can use word embeddings to compare all-pairs of query-document terms,

→ # of terms in the doc that relate to *Albuquerque*



Politecnico di Bari

Dimensionality Reduction

Embeddings – Word2vec

Word2vec

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words/Context of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

The context represents the meaning of a word

Word2vec

- The context represents the meaning of a word

«*Rome is the capital of the country Italy*»

«*London is the capital of the country UK*»

Word2vec

- The context represents the meaning of a word

«*Rome is the capital of the country Italy*»

«*London is the capital of the country UK*»

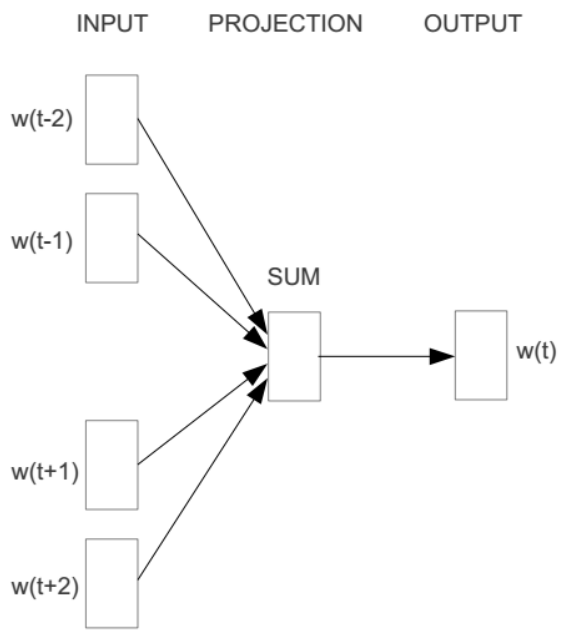
Rome is similar to *London* as they share the same context *capital*

Italy is similar to *UK* as they share the same context *country*

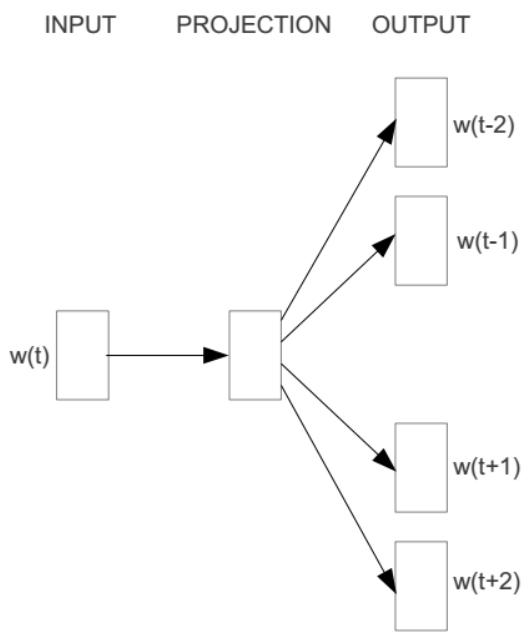
Represent the meaning of word

2 basic (quasi) Neural Network models:

- Continuous Bag of Word (CBOW): uses a window of words to predict the middle word
- Skip-gram (SG): uses a word to predict the surrounding ones in window.



CBOW

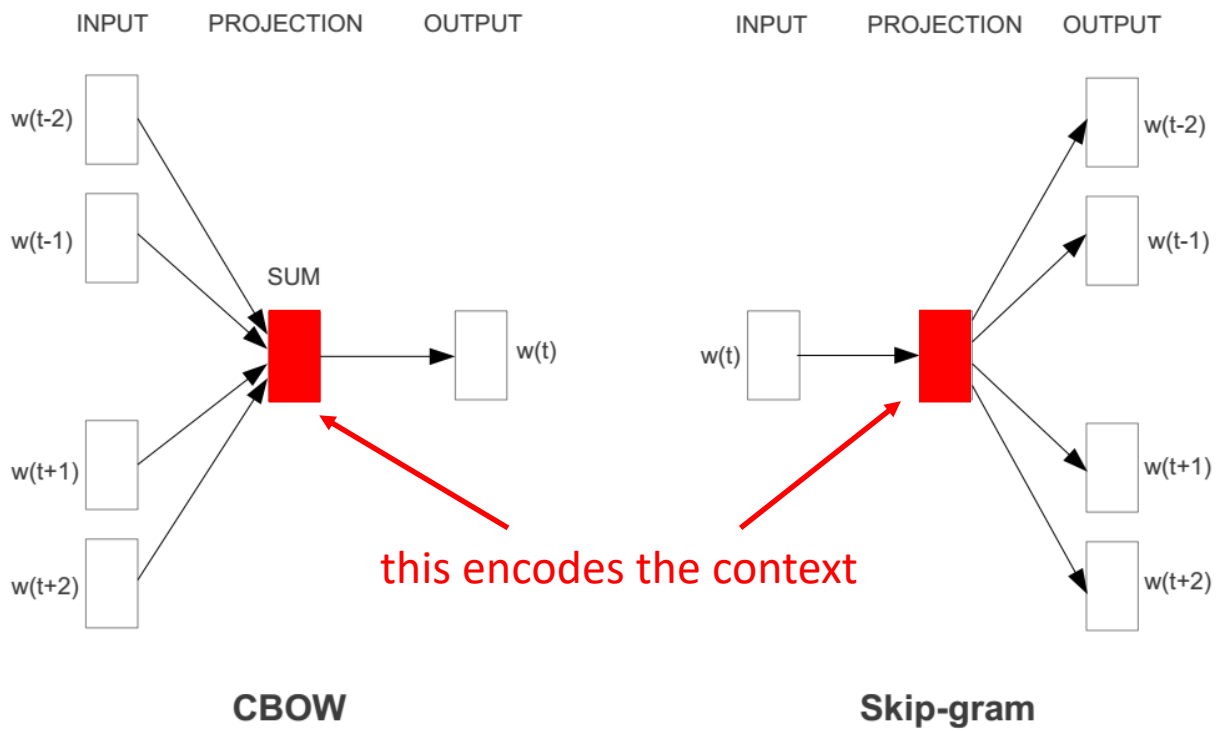


Skip-gram

Represent the meaning of word

2 basic (quasi) Neural Network models:

- Continuous Bag of Word (CBOW): uses a window of words to predict the middle word
- Skip-gram (SG): uses a word to predict the surrounding ones in window.



Word2vec – Continuous Bag of Word

“The cat sat on the floor”

“I always sat on my favorite chair”

“Just sat on the sofa”

Input	Output
the	cat
cat	sat
sat	on
the	floor
I	always
always	sat
on	my
my	favorite
…	…

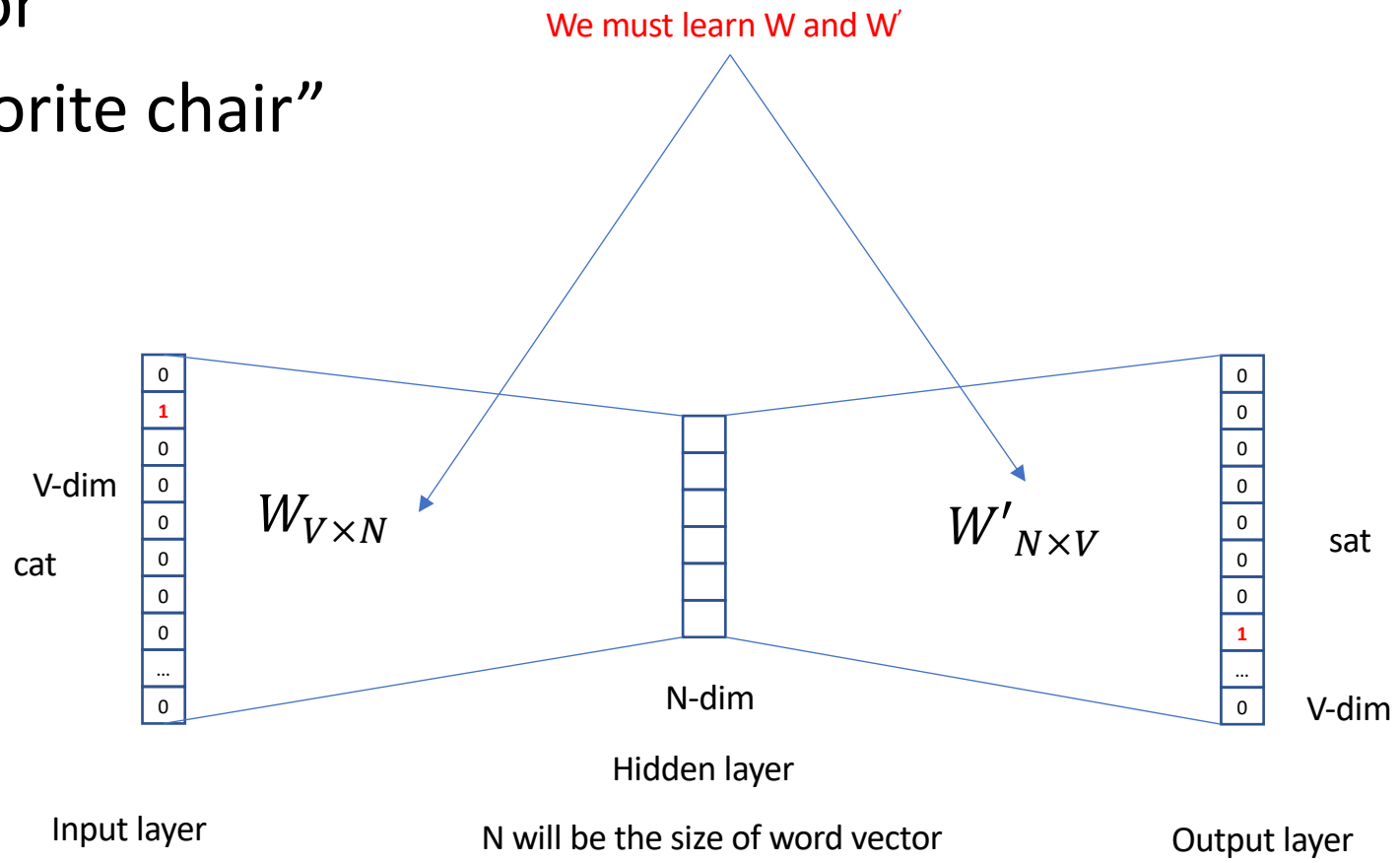
the	cat	sat	on	floor	I	always	my	favorite	…
1	0	0	0	0	0	0	0	0	…
0	**1**	0	0	0	0	0	0	0	…
0	0	**1**	0	0	0	0	0	0	…
0	0	0	**1**	0	0	0	0	0	…
0	0	0	0	**1**	0	0	0	0	…
0	0	0	0	0	**1**	0	0	0	…
0	0	0	0	0	0	**1**	0	0	…
0	0	0	0	0	0	0	**1**	0	…
0	0	0	0	0	0	0	0	**1**	…

Word2vec – Continuous Bag of Word

"The cat sat on the floor"

"I always sat on my favorite chair"

"Just sat on the sofa"



Word2vec – Continuous Bag of Word

$$W_{V\times N}^{T} \times x_{cat} = v_{cat}$$

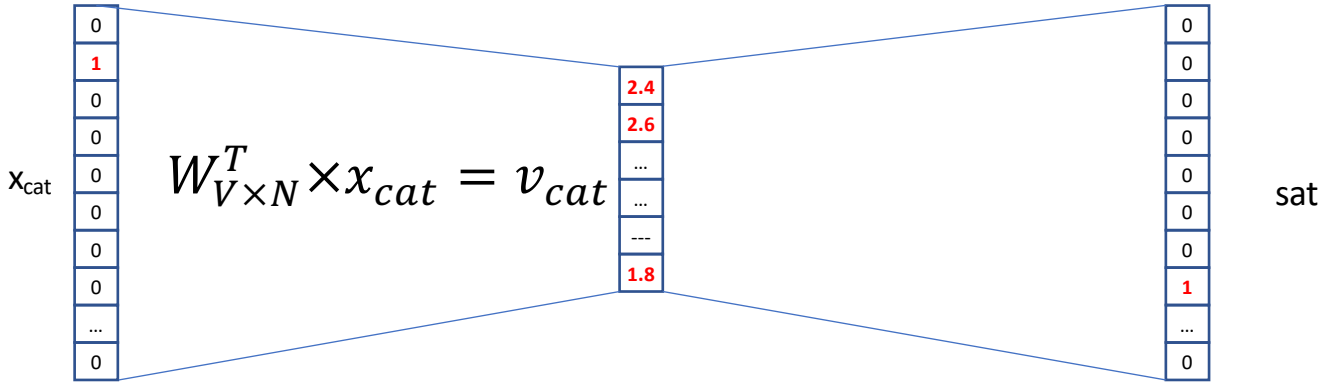
0.1	**2.4**	1.6	1.8	0.5	0.9	…	…	…	3.2
0.5	**2.6**	1.4	2.9	1.5	3.6	…	…	…	6.1
…	**…**	…	…	…	…	…	…	…	…
…	**…**	…	…	…	…	…	…	…	…
0.6	**1.8**	2.7	1.9	2.4	2.0	…	…	…	1.2

×

0
1
0
0
0
0
0
0
…
0

=

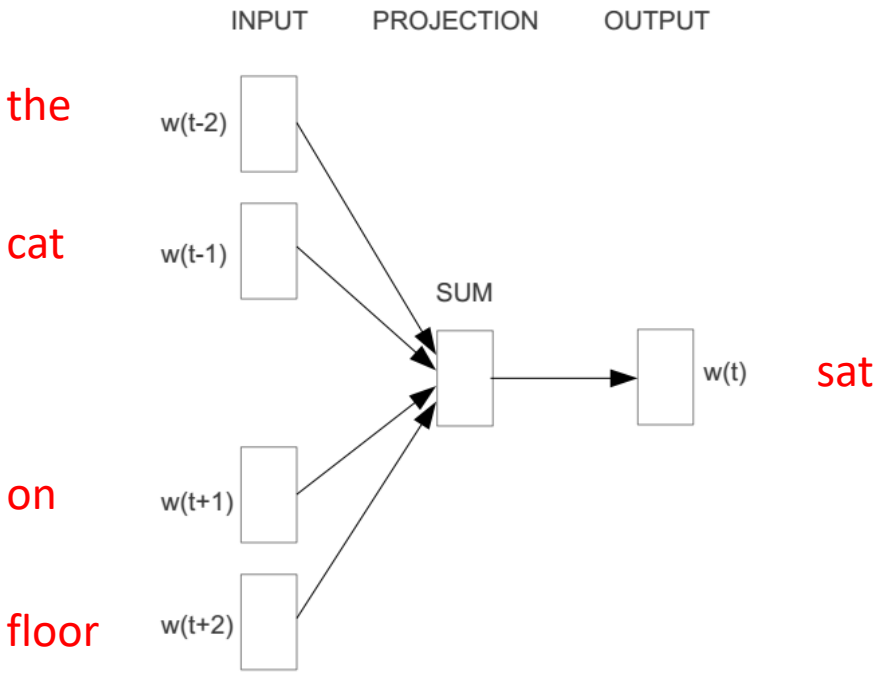
2.4
2.6
…
…
1.8



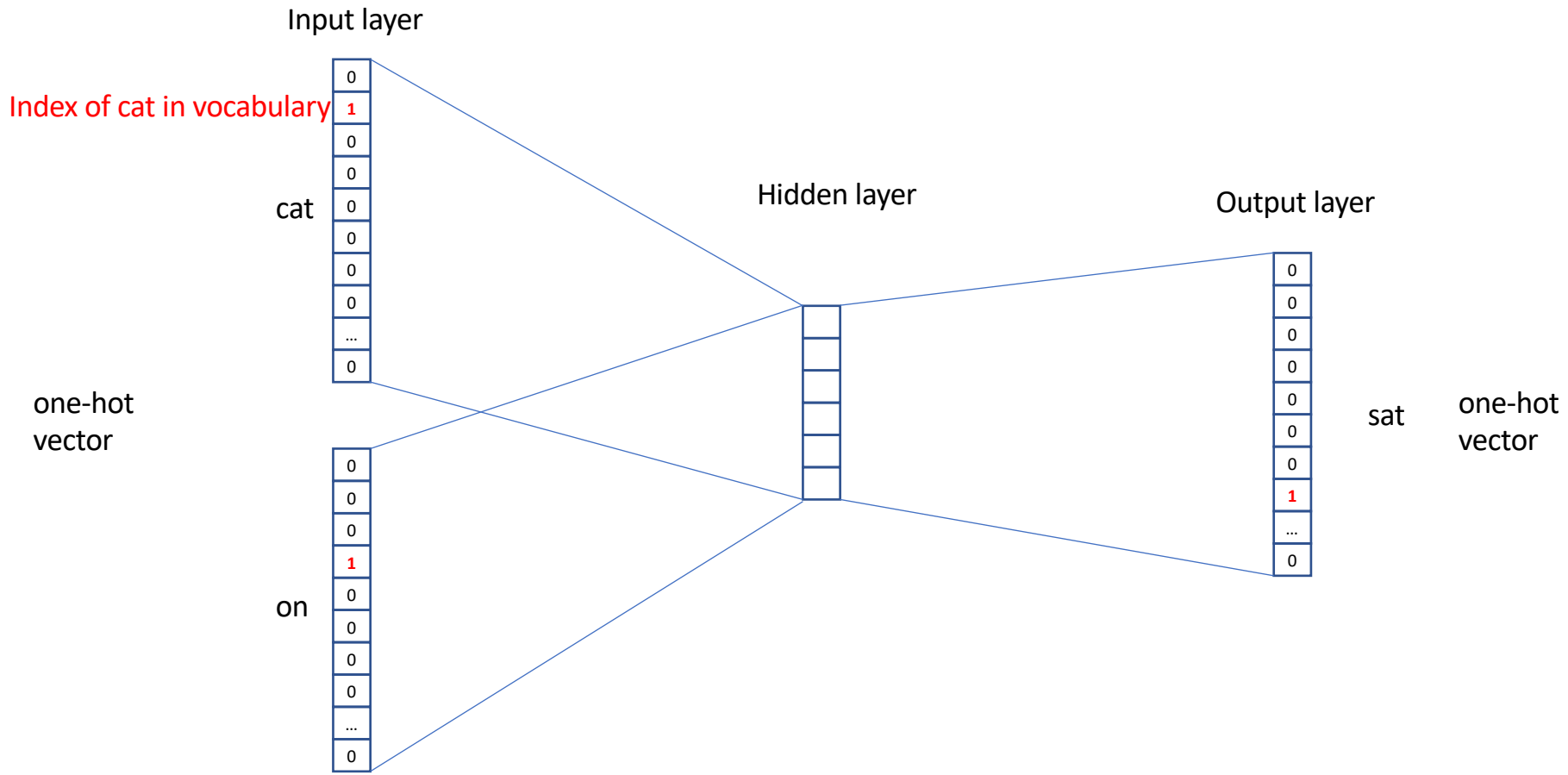
Word2vec – Continuous Bag of Word

E.g. “The cat sat on floor”

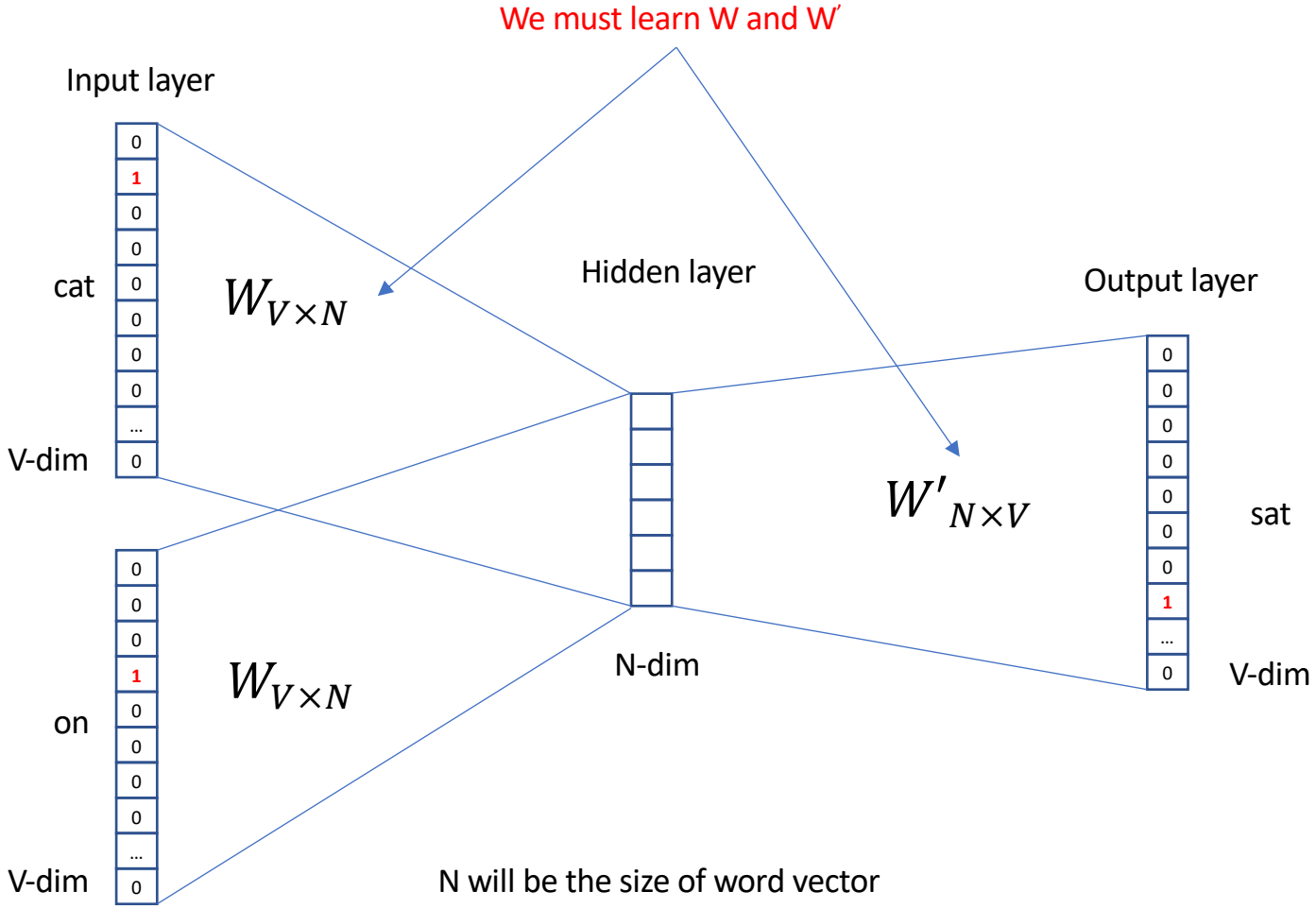
- Window size = 2



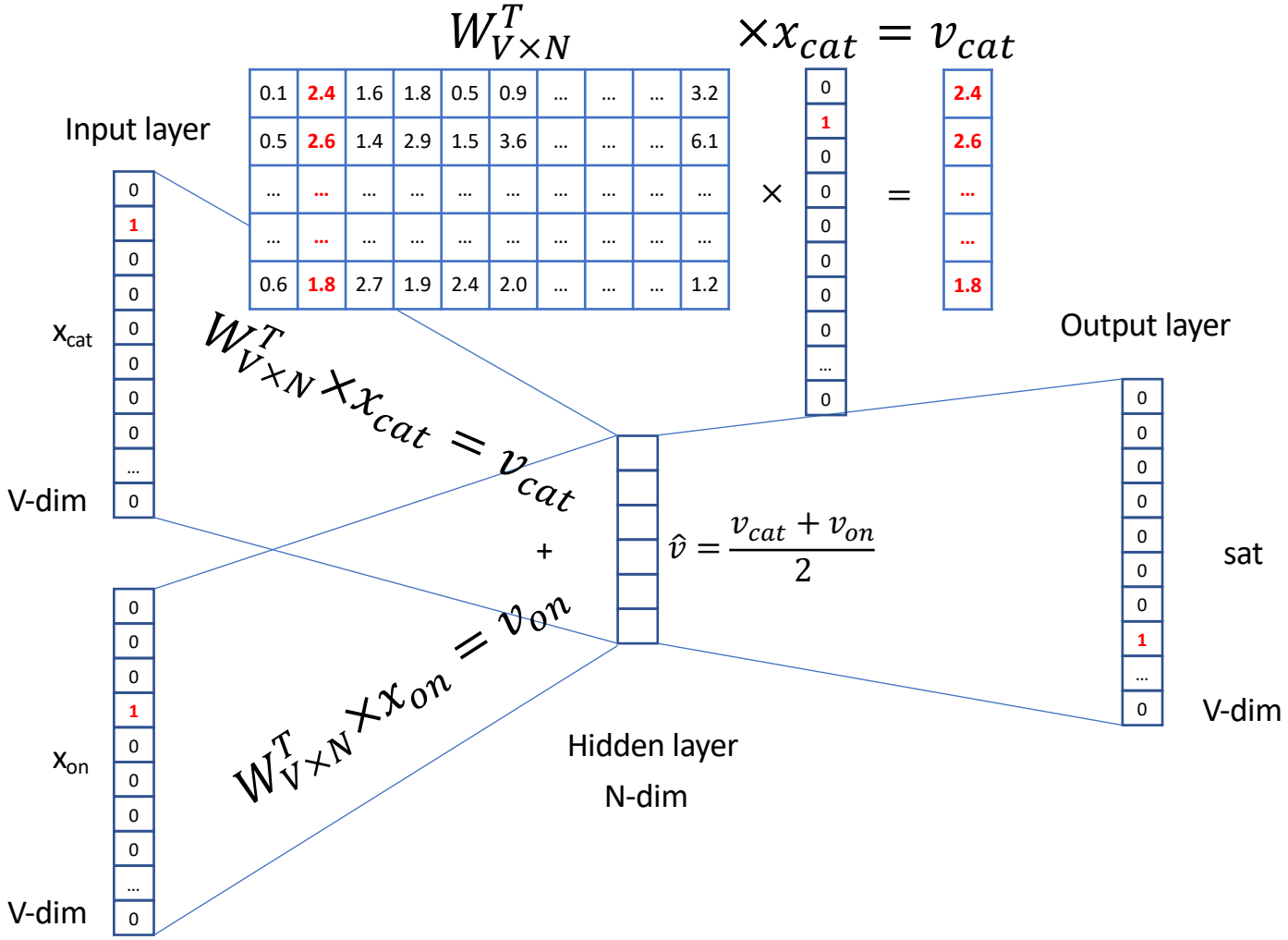
Word2vec – Continuous Bag of Word



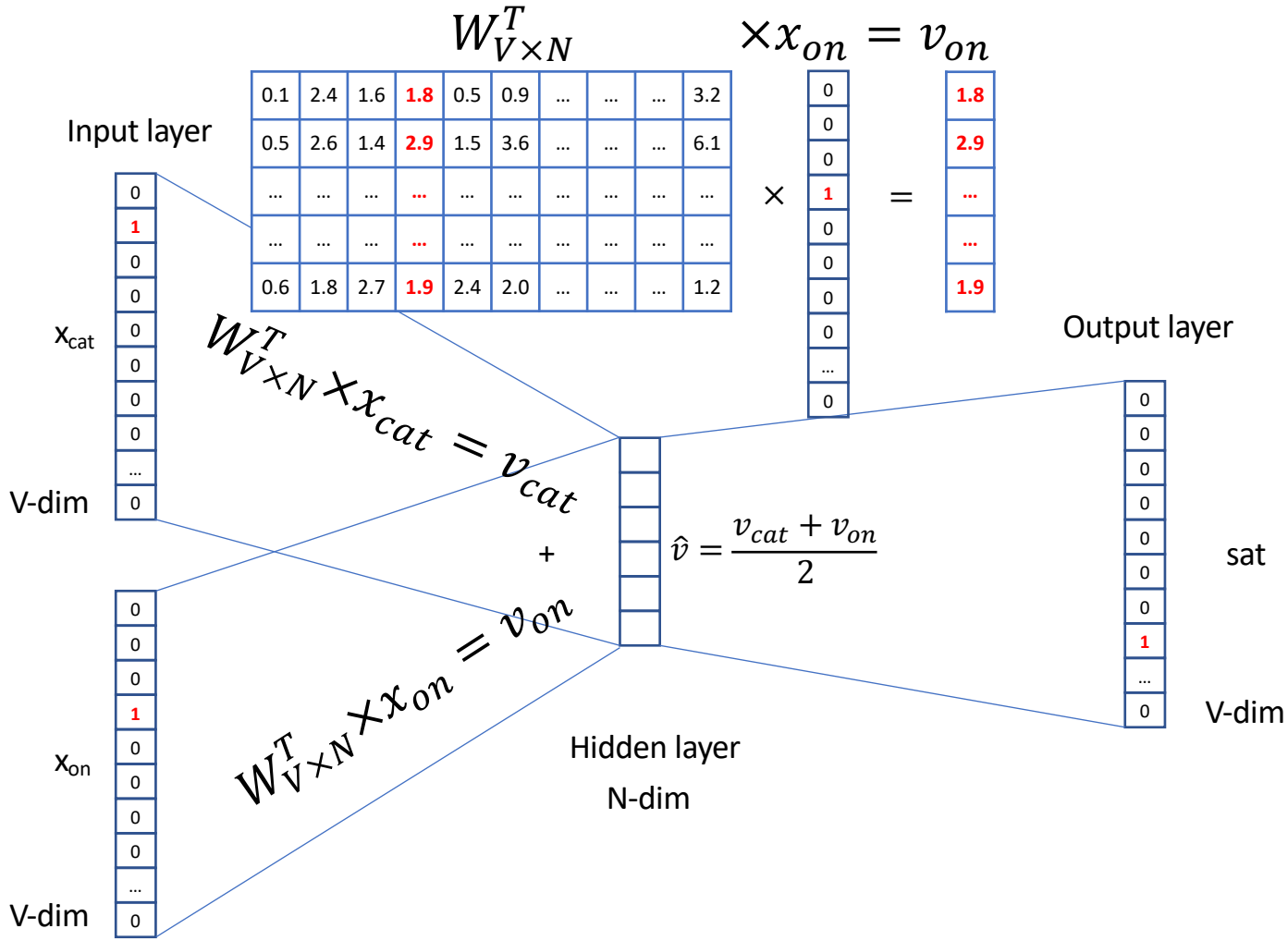
Word2vec – Continuous Bag of Word



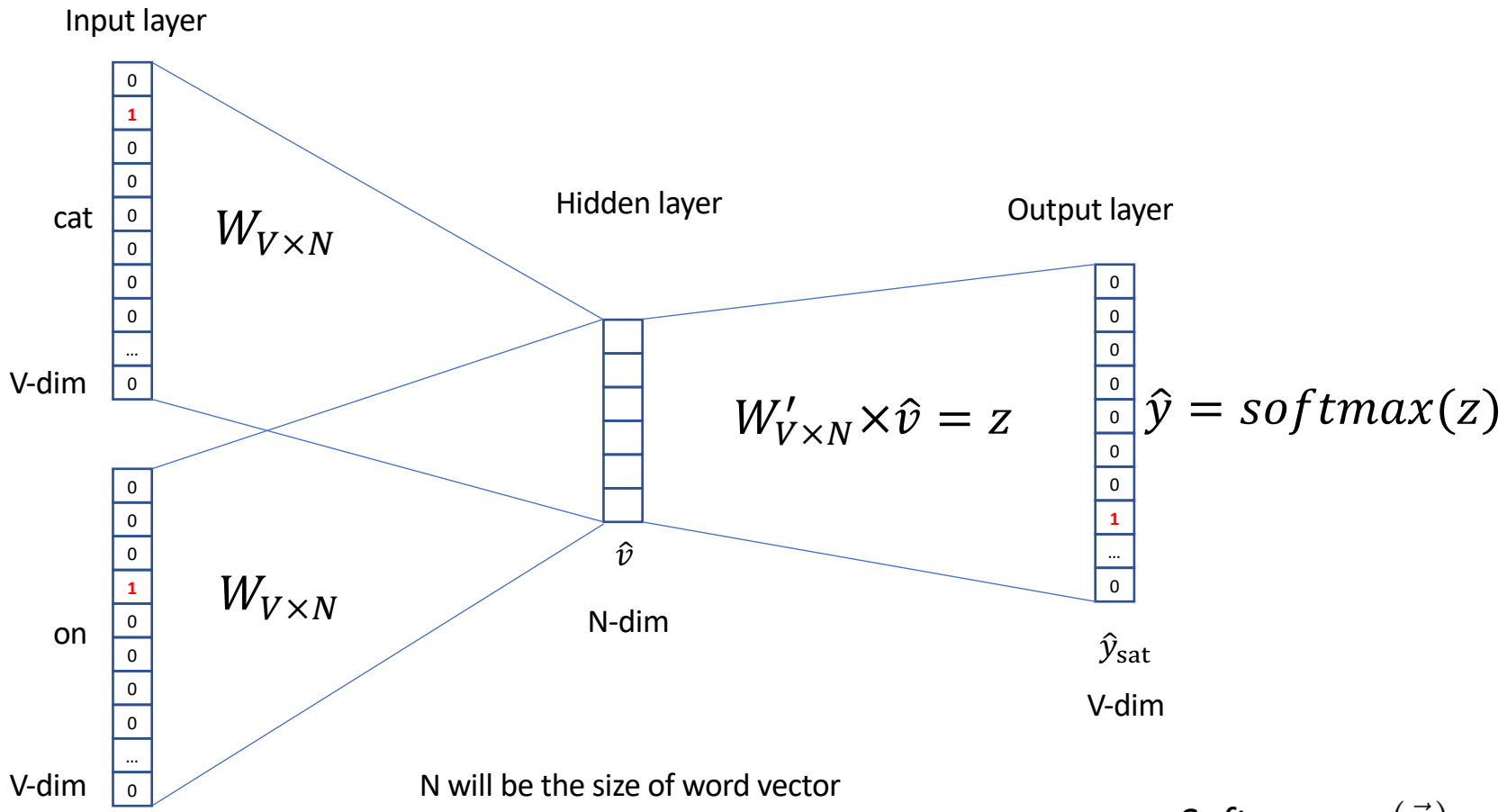
Word2vec – Continuous Bag of Word



Word2vec – Continuous Bag of Word

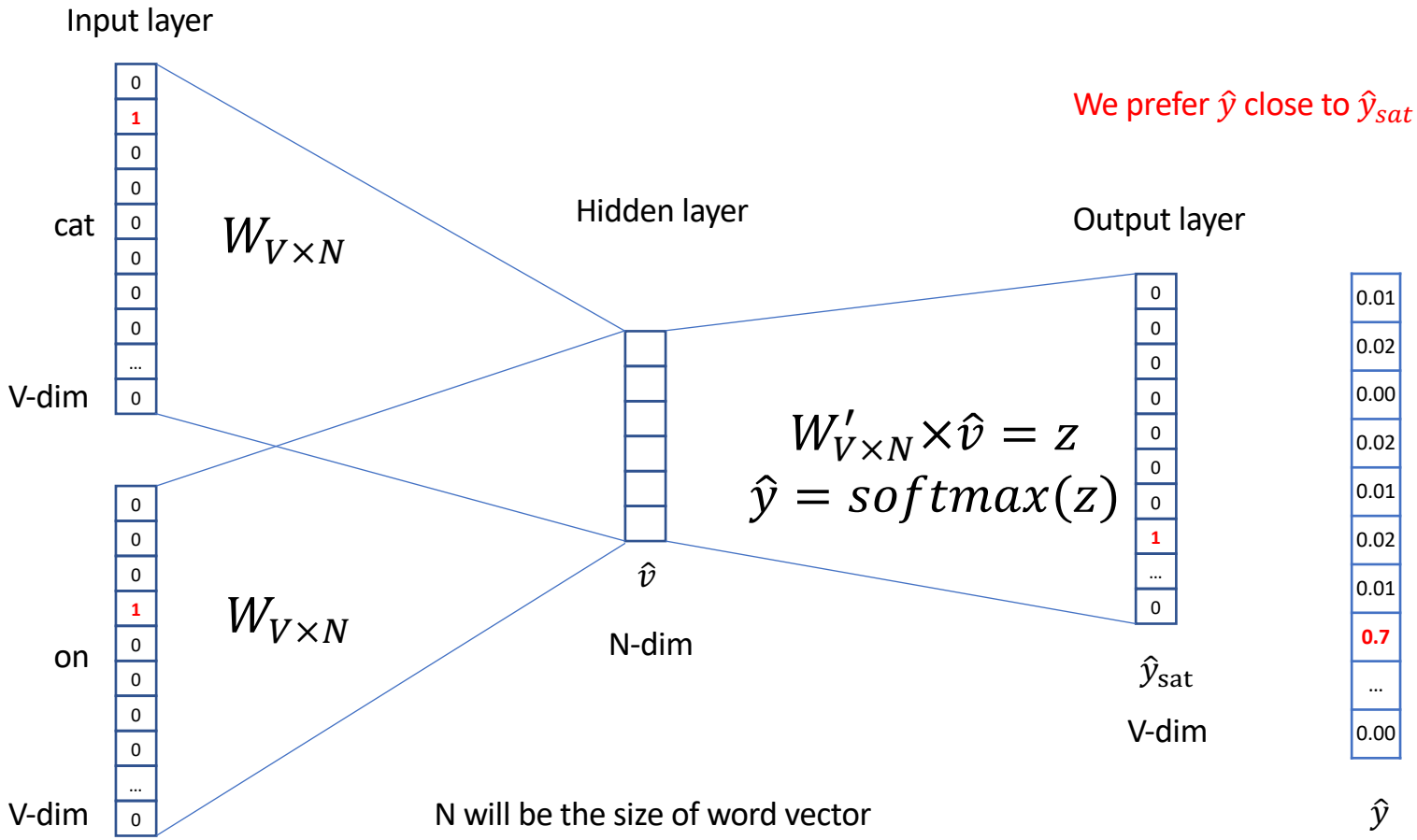


Word2vec – Continuous Bag of Word

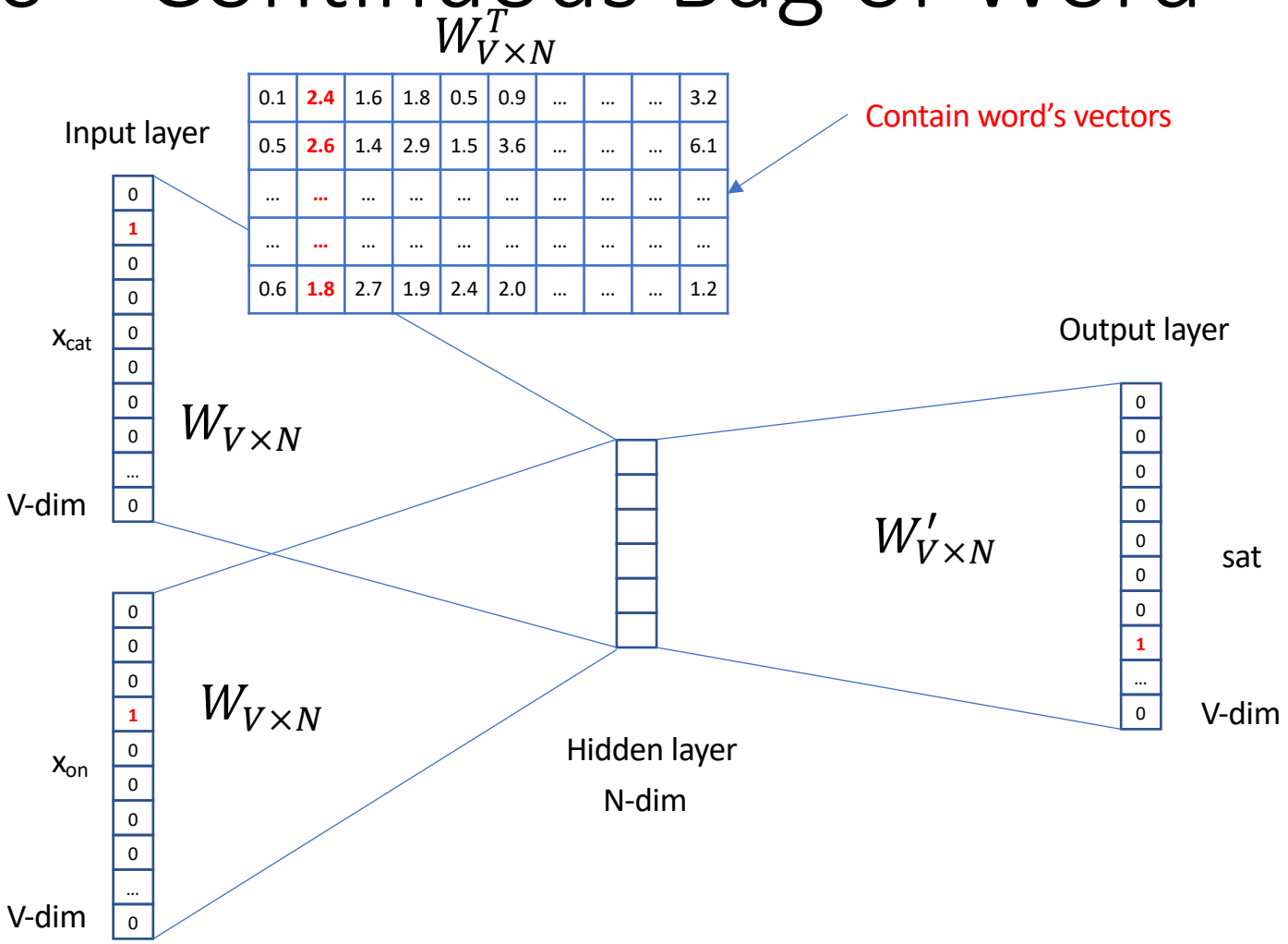


Softmax $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Word2vec – Continuous Bag of Word



Word2vec – Continuous Bag of Word



We can consider either W or W' as the word's representation. Or even take the average.