MACHINE LEARNING

UNSUPERVISED LEARNING II

AGENDA

01 Principal Component Analysis

02 Independent Component Analysis

PRINCIPAL COMPONENT ANALYSIS

PRINCIPAL COMPONENT ANALYSIS

I N T R O D U C T I O N

The **algorithm** has a **similar motivation** as **factor analysis**, where it **tries** to **identify** the **subspace** in **which** the **data approximately** lies**.** However, **PCA** will **only require** an **eigenvector calculation** and **doesn´**t make **use** of the **EM algorithm**.

Given a **dataset** $\{\boldsymbol{x^{(i)}};\ \boldsymbol{i} = \mathbf{1},\ldots,\boldsymbol{m}\}$ where $\boldsymbol{x^{(i)}} \in \mathbb{R}^{\boldsymbol{n}}, (\boldsymbol{n} \ll \boldsymbol{m})$ we want to **reduce it** to a **low-dimensional dataset** of $\boldsymbol{k}$ **dimensions** $(\boldsymbol{k} < \boldsymbol{n})$.

The motivation is that **there may be two different attributes**, **unknown** to us that are **almost linearly dependent** (**redundancy - correlation**).

Thus, the **data** really **lies** approximately on an $\boldsymbol{n} - \mathbf{1}$ **dimensional subspace**.

PRINCIPAL COMPONENT ANALYSIS

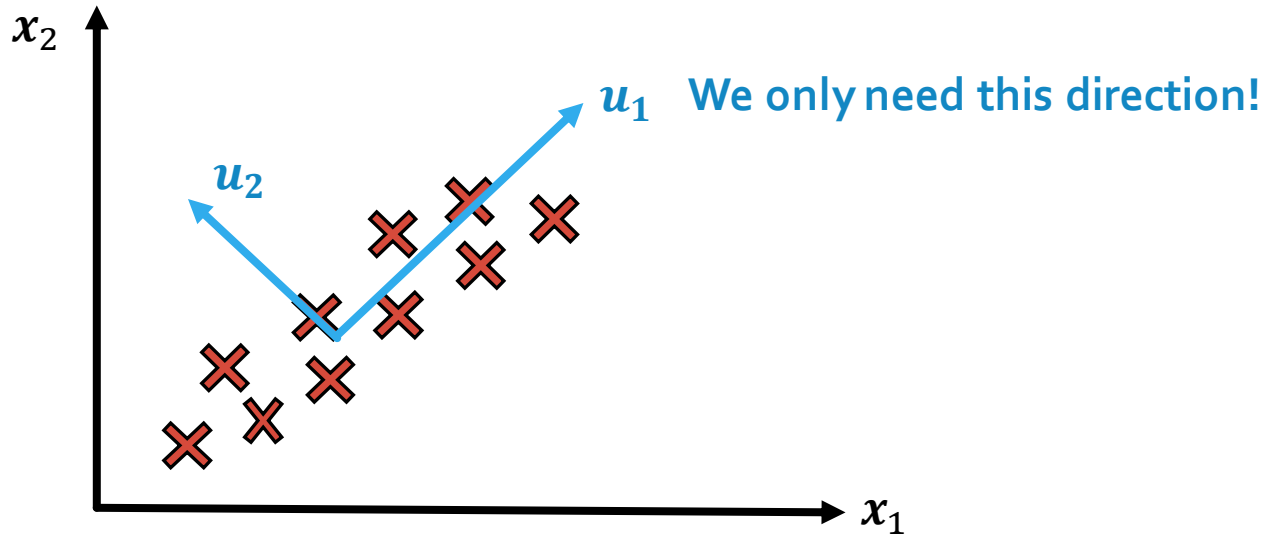
I N T R O D U C T I O N

EXAMPLE:

Consider a **dataset resulting** from a **survey** of **pilots** for **radio-controlled helicopters**:

- $x_1^{(i)}$: measure of the piloting skill of pilot i.
- $x_2^{(i)}$: captures how much he/she enjoys flying.

Only the **most committed students**, the **ones** that **truly enjoy flying**, **become good pilots**.



PRINCIPAL COMPONENT ANALYSIS

N O R M A L I Z A T I O N

Prior to **running** the **PCA algorithm**, we will **normalize** the **data** to have **mean 0** and **variance 1**:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$

Dividing by the **standard deviation** σ_j **rescales** each **coordinate** to **have unit variance**, which **ensures** that **different attributes** are all **treated on** the **same** "**scale**."

PRINCIPAL COMPONENT ANALYSIS

SETTING THE PROBLEM

The **main question** to be addressed is the following**:**

How do we compute the "major axis of variation" u, the direction on which the data approximately lies?

We can **set** this **problem** as **finding** the **unit vector u,** so that **when** the **data is projected** onto the **direction** of u**,** the **variance of** the **projected data** is **maximized**.

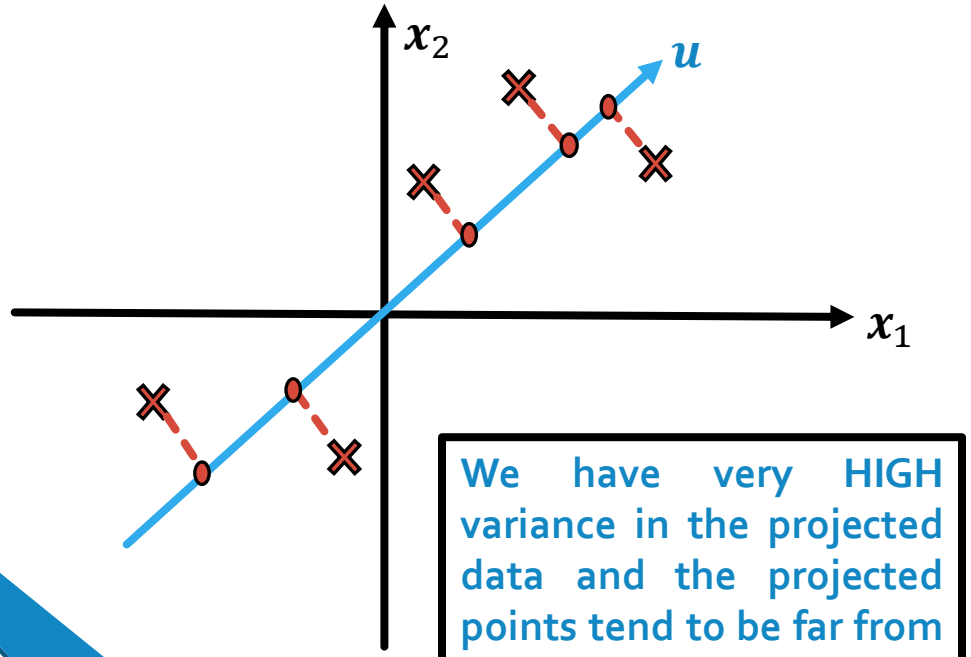
In other words, if **we** were to **approximate** the **data** as **lying** in the **direction/subspace** corresponding to u, **as much** as possible **of this variance** is **still retained**.

PRINCIPAL COMPONENT ANALYSIS

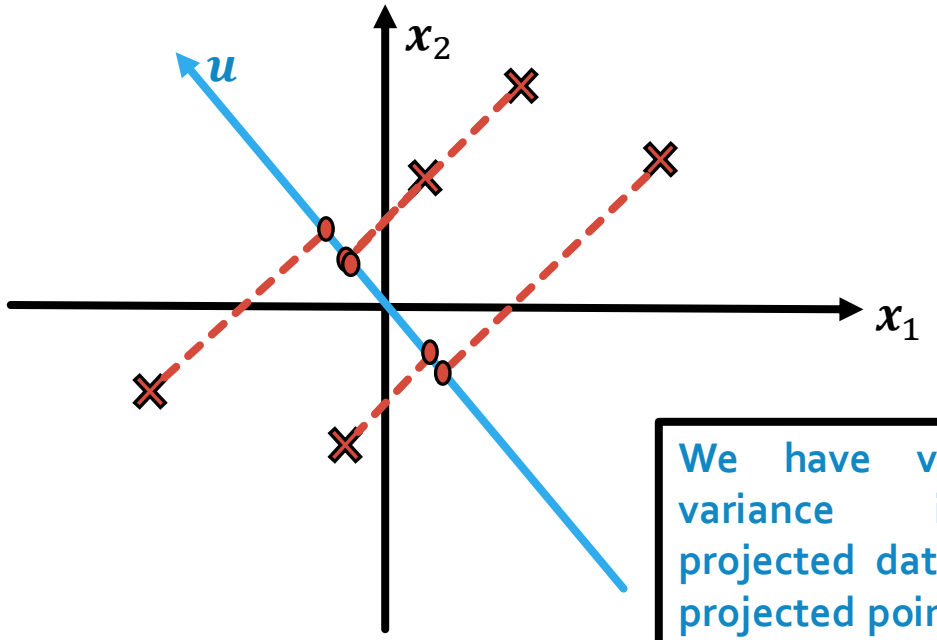
SETTING THE PROBLEM

EXAMPLE:

Consider the following **dataset** already **normalized:** (we would like to **automatically select** the **direction u corresponding** to the **left figure**).



We have very HIGH variance in the projected data and the projected points tend to be far from zero.

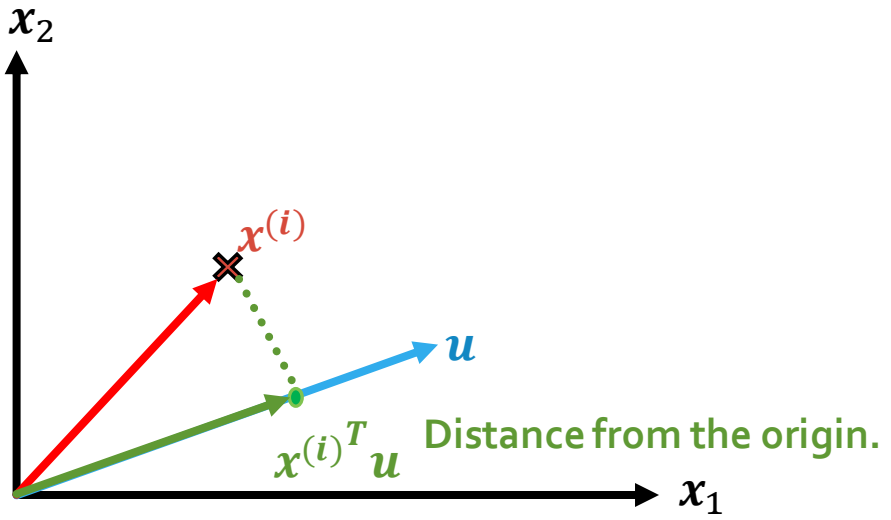


We have very LOW variance in the projected data and the projected points tend to be closer from zero.

PRINCIPAL COMPONENT ANALYSIS

THE ALGORITHM

To **formalize** the **algorithm**, note that **given unit vector** $\boldsymbol{u}$ and a **point** $\boldsymbol{x}$**,** the **length of** the **projection** of $\boldsymbol{x}$ onto $\boldsymbol{u}$ is **given** by $\boldsymbol{x^T u}$.



To **maximize** the **variance of** the **projections**, we would like to **choose** a **unit-length** $\boldsymbol{u}$ to **maximize**:

$$\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)^T}u\right)^2 = \frac{1}{m}\sum_{i=1}^{m} u^T x^{(i)} x^{(i)^T} u$$

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

Therefore, we want to **maximize** the **following expression** with **respect** to unit **vector** $\boldsymbol{u}$, where we have the **constraint** $\|\boldsymbol{u}\|_2 = \mathbf{1}$:

$$\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\boldsymbol{x}^{(\boldsymbol{i})^{\boldsymbol{T}}}\boldsymbol{u} = \boldsymbol{u}^{\boldsymbol{T}}\left(\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{x}^{(\boldsymbol{i})}\boldsymbol{x}^{(\boldsymbol{i})^{\boldsymbol{T}}}\right)\boldsymbol{u}$$

$$\max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2=\mathbf{1}} \boldsymbol{u}^{\boldsymbol{T}}\left(\frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{x}^{(\boldsymbol{i})}\boldsymbol{x}^{(\boldsymbol{i})^{\boldsymbol{T}}}\right)\boldsymbol{u}$$

The **covariance matrix** will be **substituted** as follows:

$$\max_{\boldsymbol{u}:\|\boldsymbol{u}\|_2=\mathbf{1}} \boldsymbol{u}^{\boldsymbol{T}}(\boldsymbol{\Sigma})\boldsymbol{u}$$

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

We **construct** the **Lagrangian constraint**:

$$\|\boldsymbol{u}\|_2 = \mathbf{1}$$

$$\|\boldsymbol{u}\|_2 - \mathbf{1} = \mathbf{0}$$

Substituting in the **Lagrangian,** where $\boldsymbol{\lambda}$ is the **Lagrange multiplier**:

$$\boldsymbol{L(u, \lambda) = u^T(\Sigma)u - \lambda(\|u\|_2 - 1)}$$

$$\boldsymbol{L(u, \lambda) = u^T(\Sigma)u - \lambda\left(u^T u - 1\right)}$$

Taking the **gradient** with **respect** to $\boldsymbol{u}$ and **applying** the **property** $\boldsymbol{\nabla_u\, u^T Au = 2Au}$ **(**symmetric covariance**):**

$$\boldsymbol{\nabla_u L(u, \lambda) = 2\Sigma u - 2\lambda u}$$

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

Setting the **gradient** to **0**:

$$2\Sigma u - 2\lambda u = 0$$

$$\Sigma u = \lambda u$$

This is the **problem** of **finding** the **eigen vectors** and **eigen values** of Σ

$$\lambda u - \Sigma u = 0$$

$$(\lambda I - \Sigma)u = 0$$

$$det(\lambda I - \Sigma)u = 0$$

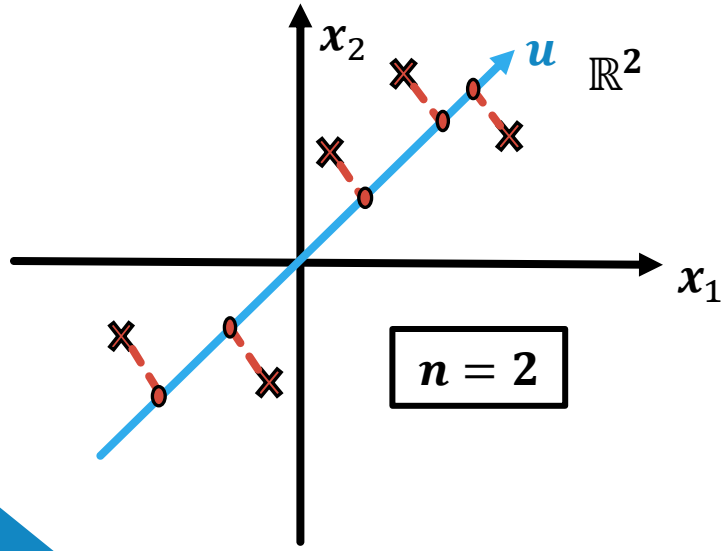
PRINCIPAL COMPONENT ANALYSIS

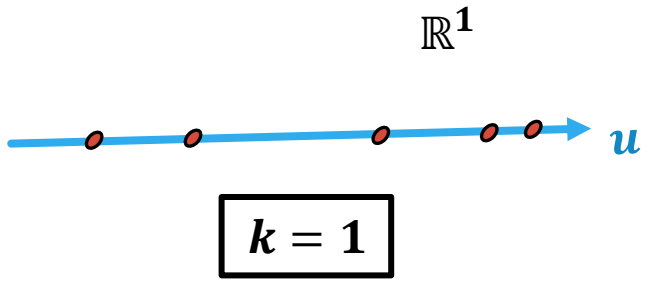
THE ALGORITHM

If we want to **find** a **1-dimensional subspace** to **approximate** the **data**, we should **choose** $\boldsymbol{u}$ **to be** the **principal eigenvector** of the **covariance matrix** $\boldsymbol{\Sigma}$.

More **generally**, if we wish to **project** our **data into** a $\boldsymbol{k}$**-dimensional subspace** ($\boldsymbol{k} < \boldsymbol{n}$), we should **choose** $\boldsymbol{u_1, \ldots, u_k}$ to be the **top** $\boldsymbol{k}$ **eigenvectors** of the **covariance matrix** $\boldsymbol{\Sigma}$**.**

The $\boldsymbol{u_i's}$ will now **form** a **new**, **orthogonal basis for** the **data**.





PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

If we want to **represent** our **original input vector** $x^{(i)}$ **in** this **new basis**, we **compute** the **following vector** $x'^{(i)}$:

$$x'^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k$$

Therefore, our **original vector** $x^{(i)}$ **that was** in an d**-dimensional** space **is now represented** in a **lower** k**-dimensional space**.

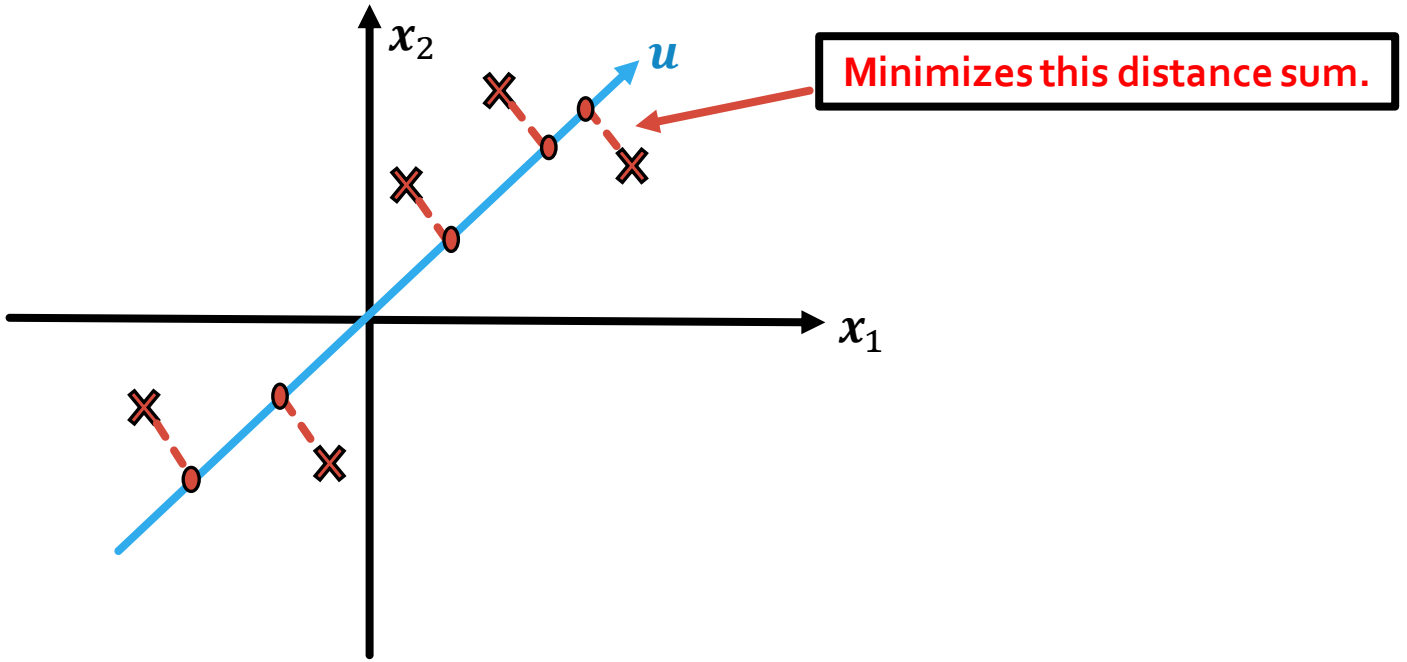
This is the **reason why** the **PCA algorithm** is **also called** the **dimensionality reduction algorithm**.

The vectors $u_1, \ldots, u_k$ are **called** the **first k principal components** of the **data.**

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

You can also **think about PCA** as an **algorithm** that **tries** to **choose** a **subspace** that **minimizes** the **squared mean error sum** of the **differences between** the **projected data and** the **original data**.



PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

SUMMARY:

1. **Normalize** the **data** to **0 mean** and **unit variance**.
2. **Compute** the **covariance matrix**:

$$\boldsymbol{\Sigma} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)T}$$

3. **Find** the $\boldsymbol{k}$ **eigen vectors** of $\boldsymbol{\Sigma}$.

PRINCIPAL COMPONENT ANALYSIS

SINGULAR VALUE DECOMPOSITION

One of the **main problems** resides in **computing** the **covariance matrix Σ,** which **can be very large** in **many cases (high dimensional data).**

We are going to **use Singular Value Decomposition** (**SVD**) from **linear algebra** which stated that **any matrix** $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{m\,x\,n}}$ can be **decomposed** as **follows**:

$$\boldsymbol{A = UDV^T}$$

Where $\boldsymbol{U} \in \mathbb{R}^{\boldsymbol{m\,x\,n}}$, $\boldsymbol{D} \in \mathbb{R}^{\boldsymbol{n\,x\,n}}$ and $\boldsymbol{V^T} \in \mathbb{R}^{\boldsymbol{n\,x\,n}}$. Matrix $\boldsymbol{D}$ is **diagonal:**

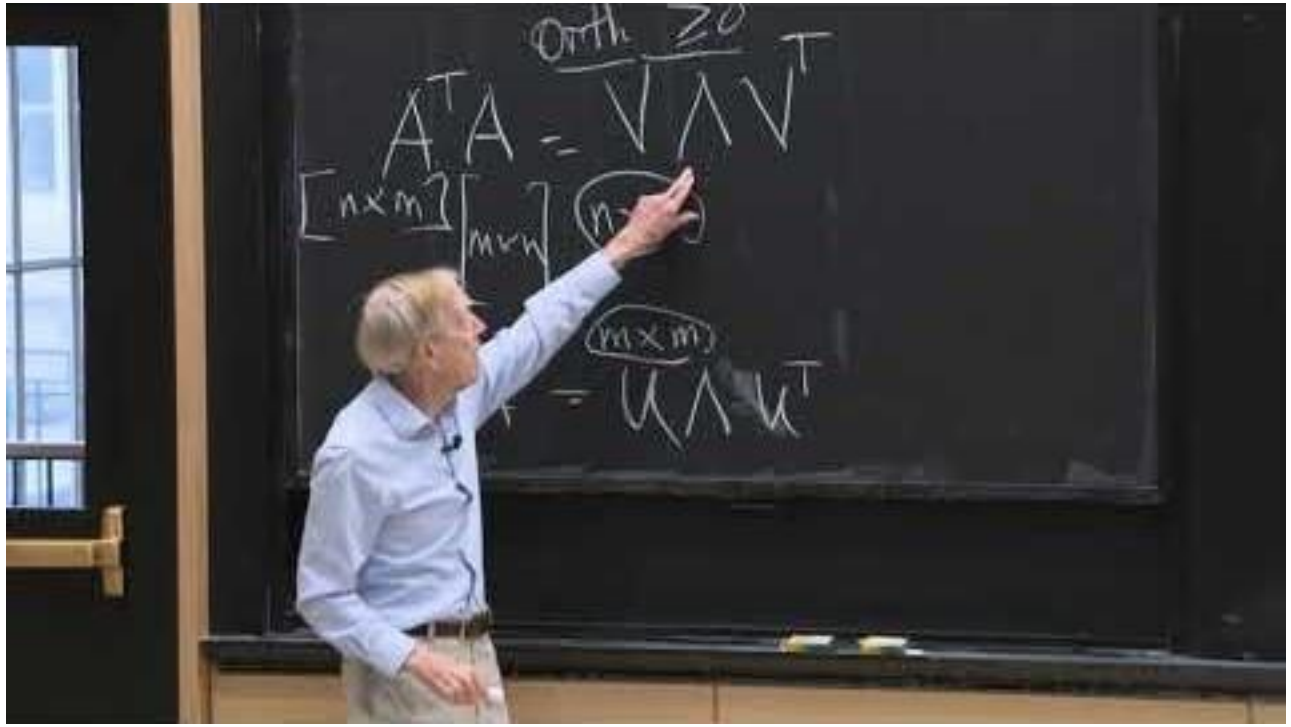
$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{\sigma_1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\sigma_2} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\vdots} & \boldsymbol{\cdots} & \boldsymbol{\ddots} & \boldsymbol{\vdots} \\ \boldsymbol{0} & \boldsymbol{\cdots} & \boldsymbol{0} & \boldsymbol{\sigma_n} \end{bmatrix}$$

Where $\boldsymbol{\sigma_i}$ are called the **singular values** of **matrix** $\boldsymbol{A}$.

PRINCIPAL COMPONENT ANALYSIS

SINGULAR VALUE DECOMPOSITION

For a **detailed description** on **Singular Value Decomposition** check **this video** from **MIT**:



https://www.youtube.com/watch?v=rYz83XPxiZo

PRINCIPAL COMPONENT ANALYSIS

SINGULAR VALUE DECOMPOSITION

In this **decomposition** we have the **following:**

- $\boldsymbol{U's}$ **columns:** eigen vectors of $\boldsymbol{AA^T}$**.**
- $\boldsymbol{V's}$ **columns:** eigen vectors of $\boldsymbol{A^TA}$.

You can **compute** it in **Python** using **“`numpy.linalg.svd(A)`”.**

This **mathematical technique** can be **used** in **PCA** to **compute eigen vectors** much **more efficiently**.

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

Let us start by **representing** the **covariance matrix $\boldsymbol{\Sigma}$** as follows:

$$\boldsymbol{\Sigma} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} x^{(i)^T} = X^T X$$

Where we have that $\boldsymbol{X}$ is:

$$X = \begin{bmatrix} -x^{(1)} - \\ -x^{(2)} - \\ \vdots \\ -x^{(m)} - \end{bmatrix}$$

Therefore:

$$X^T X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} -x^{(1)} - \\ -x^{(2)} - \\ \vdots \\ -x^{(m)} - \end{bmatrix}$$

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

To **get** the $\boldsymbol{k}$ **eigen vectors** of $\boldsymbol{X}$ we **apply SVD** to it:

$$\boldsymbol{X = UDV^T}$$

The **top** $\boldsymbol{k}$ **columns** of $\boldsymbol{V}$ will be the **top** $\boldsymbol{k}$ **eigen vectors** of $\boldsymbol{X^T X = \Sigma}$.

Thus, now we have a **method** to **compute eigen vectors** with **highly dimensional data**.

PRINCIPAL COMPONENT ANALYSIS

T H E A L G O R I T H M

Some of the **applications** of the **algorithm include**:

- **Compression** of **data**.
- **Visualizing high dimensional data** to see clusters (**anomaly detection**).
- **Pre-processing** the **data before** it is **fed** to a **supervised learning algorithm** allowing **less computational expense** and **reducing** the **complexity** of the **learnt hypothesis** (**reduce overfitting**).
- **Noise reduction** (only **consider** the **most important features** and **remove slight variations** that **do not contribute**).
- **Matching** (**distance calculations**).

INDEPENDENT COMPONENT ANALYSIS

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

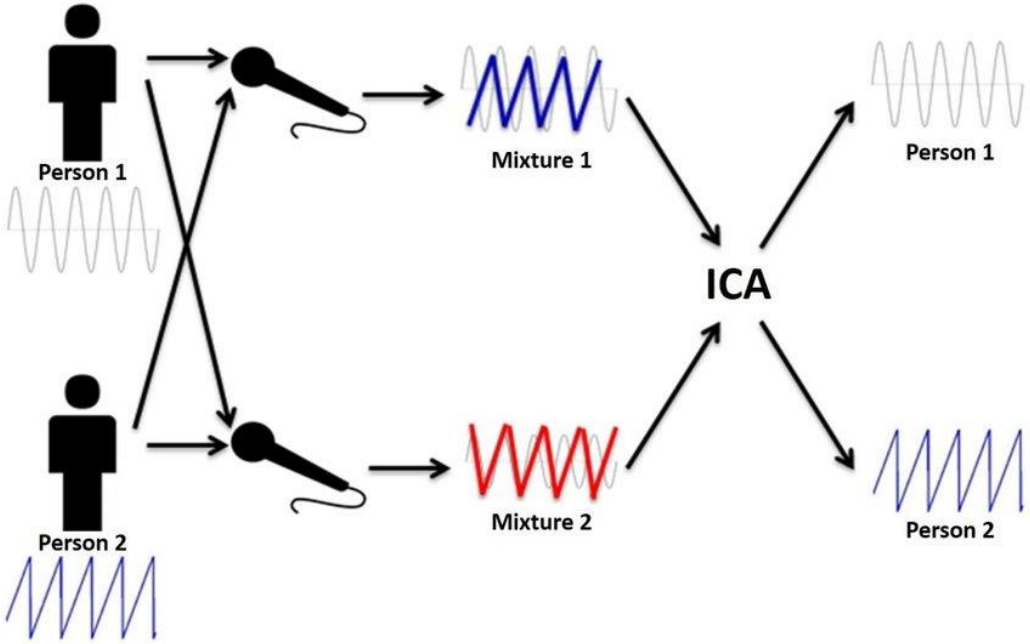
INTRODUCTION

The **new objective** will reside in **finding** the **independent components** that **capture** the **greatest variation** of the **data**.

The **motivating example** is the **cocktail party problem**: let us define n different **microphones** and n **speakers** placed in a room.

Because **each microphone** is a **different distance** from each **speaker**, it **captures** a **different combination** of the **speakers' voices**.

Using the microphone recordings, **can we separate** out the **original** n **speakers' speech signals**?



INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

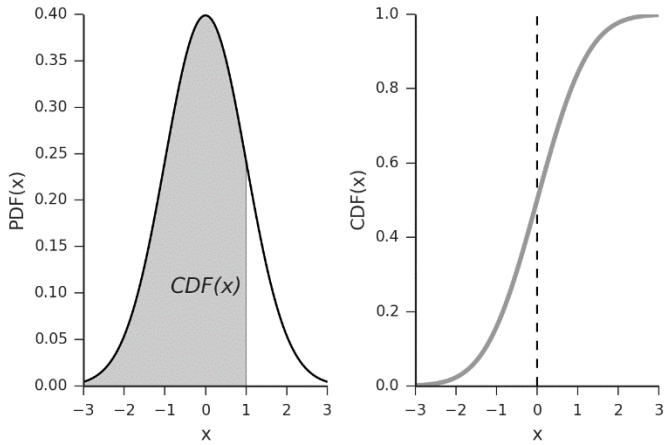
INTRODUCTION

First, we are going to remember **cumulative distribution functions** (**CDFs**). Let us suppose that you have a **random variable** $\boldsymbol{s}$ and has a **probability density function** $\boldsymbol{p_s(s)}$**.**

Therefore its **cumulative distribution function** will be:

$$\boldsymbol{F(s) = P(S \leq s) = \int_{-\infty}^{s} p_s(t)dt}$$

Example Gaussian:



INDEPENDENT COMPONENT ANALYSIS

THE ALGORITHM

FORMALIZATION

Let us **assume** that some **data** $s \in \mathbb{R}^n$ is **generated** via n **independent sources**. What we **observe** is**:**

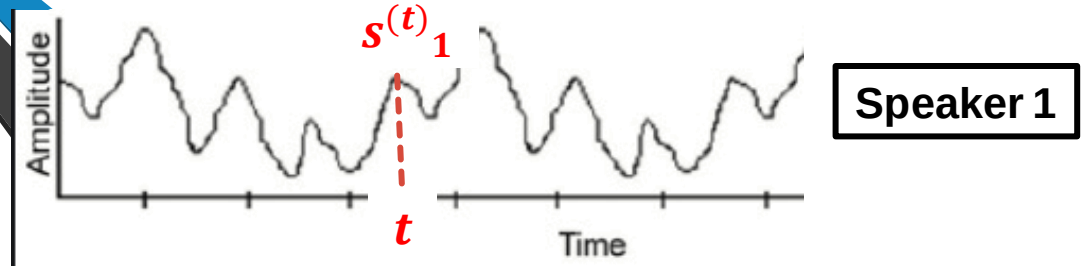
$$x = As$$

where A is an **unknown square matrix** called the **mixing matrix** (linear combination of n sources).

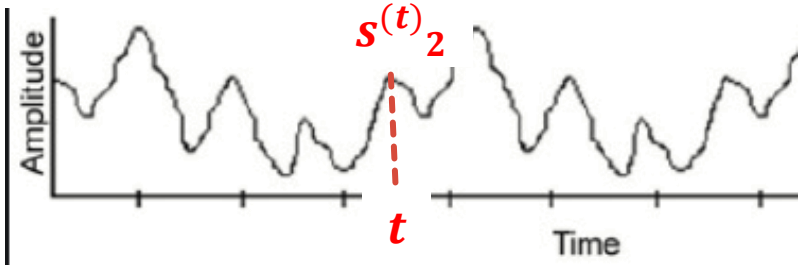
Making **repeated observations** gives us a **dataset:** $\{x^{(i)}; i = 1, \dots, m\}$ and the **objective** is to **recover** the **sources** $s^{(i)}$ that had **generated** our **data** $(x^{(i)} = As^{(i)})$.

In the **cocktail party problem** we have:

- $s^{(i)} \in \mathbb{R}^n$, where each $s^{(i)}_j$ is the **sound** that **speaker** j was **producing** at **time** i.
- $x^{(i)} \in \mathbb{R}^n$, where each $x^{(i)}_j$ is the **acoustic reading** that **microphone** j **recorded** at **time** i.



Speaker 1



Speaker 2

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

FORMALIZATION

Let us **define** $\boldsymbol{W = A^{-1}}$ as the **unmixing matrix.** Therefore, the **objective** is to **find** $\boldsymbol{W}$, such that the **sources** $\boldsymbol{s^{(i)}}$ can be **recovered from** the **recordings** $\boldsymbol{x^{(i)}}$ by computing:

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

where the **entries** of $\boldsymbol{W}$ are **computed** as **follows**, where $\boldsymbol{w_i} \in \mathbb{R}^{\boldsymbol{n}}$:

$$\boldsymbol{W} = \begin{bmatrix} -\boldsymbol{w_1^T} - \\ \vdots \\ -\boldsymbol{w_n^T} - \end{bmatrix}$$

INDEPENDENT COMPONENT ANALYSIS

THE ALGORITHM

AMBIGUITIES: PERMUTATION

The **following question arises**:

To what degree can $W = A^{-1}$ be recovered?

Given only the **recordings** $x^{(i)}$ and **no prior information** of both, the **sources** $s^{(i)}$ and the **mixing matrix** A**,** many **inherent ambiguities arise** that makes **impossible** to **recover some information** of W from A**.**

Concretely, let us **define** P as $n \; x \; n$ **permutation matrix** (each **row** and **column** has **exactly one "1").**

If z is a **vector**, then Pz is **another vector** that **contains** a **permuted version** of z**'**s **coordinates**.

$$Pz = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

AMBIGUITIES: PERMUTATIONS

Given only the **recordings** $\boldsymbol{x^{(i)}}$, there is **no way** to **distinguish** between $\boldsymbol{W}$ and $\boldsymbol{PW}$ (**permutated version** of **unmixing matrix**) because you **obtain** the **same sources** but in **different order**.

The **permutation** of the **original sources** is **ambiguous**, but this **doesn't matter** for **most applications**.

The only drawback is that you will not know which source corresponds to each speaker.

INDEPENDENT COMPONENT ANALYSIS

THE ALGORITHM

AMBIGUITIES: SCALING

There is **no way** to **recover** the **correct scaling** of the $\boldsymbol{w_i}$**'s.**

If a **single column** of $\boldsymbol{A}$ were **scaled** by a **factor** of $\boldsymbol{\alpha}$, and the **corresponding source** were **scaled** by a **factor** of $\boldsymbol{1/\alpha}$, then there is again **no way** to **determine** that **this** had **happened given only** the $\boldsymbol{x^{(i)'}s}$.

Example:

$$\boldsymbol{x^{(i)} = As^{(i)}}$$

$$\boldsymbol{x^{(i)} = (2A)\left(0.5s^{(i)}\right)}$$

Again, for the **applications** that we are **concerned** with, this **ambiguity does not matter** (**scaling** a **speaker's speech signal** $\boldsymbol{s_j^{(i)}}$ by some **positive factor** $\boldsymbol{\alpha}$**, affects only** the **volume** of that **speaker's speech**).

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

AMBIGUITIES:

Both, **permutations** and **scaling**, will be the **only sources** of **ambiguity** so long as the **sources** s_i are **non-Gaussian**.

Example:

We have $n = 2$, and $s \sim N(0, I)$. Here, I is the **2x2 identity matrix**. Note that the contours of the **density** are **rotationally symmetric**.

If we **observe** some $x = As$, the **distribution** of x will be **Gaussian**, $x \sim N(0, AA^T)$.

$$E_{s \sim N(0,I)}[x] = E[As] = AE[s] = 0$$

$$Cov[x] = E_{s \sim N(0,I)}[xx^T] = E[Ass^T A^T] = AE[ss^T]A^T = A\,Cov[s]A^T = A(I)A^T$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

AMBIGUITIES:

Now, let $\boldsymbol{R}$ be an **arbitrary orthogonal** (rotation/reflection) **matrix**, so that $\boldsymbol{RR^T = R^TR = I}$, and let $\boldsymbol{A' = AR}$.

If the **data** had been **mixed according** to $\boldsymbol{A'}$ instead of $\boldsymbol{A}$, we would have instead observed $\boldsymbol{x' = A's}$.

The **distribution** of $\boldsymbol{x'}$ is also **Gaussian**, $\boldsymbol{x' \sim N(0, AA^T)}$, since

$$\boldsymbol{E_{s\sim N(0,I)}[x'(x')^T] = E[A'ss^T(A')^T]}$$

$$\boldsymbol{E_{s\sim N(0,I)}[x'(x')^T] = E[ARss^T(AR)^T]}$$

$$\boldsymbol{E_{s\sim N(0,I)}[x'(x')^T] = AR(I)R^TA^T}$$

$$\boldsymbol{E_{s\sim N(0,I)}[x'(x')^T] = ARR^TA^T}$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

AMBIGUITIES:

Therefore, **whether** the **mixing matrix** is $\boldsymbol{A}$ or $\boldsymbol{A'}$, we would **observe data** from a $\boldsymbol{N(0, AA^T)}$ **distribution**.

Thus, there is **no way** to **tell** if the **sources** were **mixed** using $\boldsymbol{A}$ and $\boldsymbol{A'}$. There is an **arbitrary rotational component** in the **mixing matrix** that **cannot** be **determined from** the **data.**

This was defined by the **assumption** that **sources** had a **multivariate standard normal distribution**, which is **rotationally symmetric**.

So long as the data is NOT Gaussian, it is possible, given enough data, to recover the $\boldsymbol{d}$ independent sources.

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

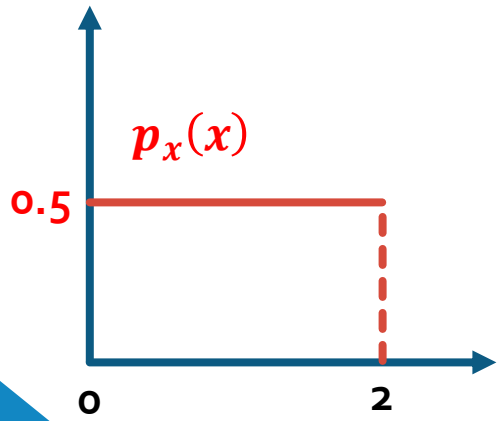
LINEAR TRANSFORMATIONS:

Suppose a **random variable** s is **drawn** according to some **density** $p_s(s)$, where $s \in \mathbb{R}$**.** Thus, our **random variable** x will be **defined** as follows **(**$A \in \mathbb{R}$**):**

$$x = As$$

What will be the **density** of x, $p_x(x)$?

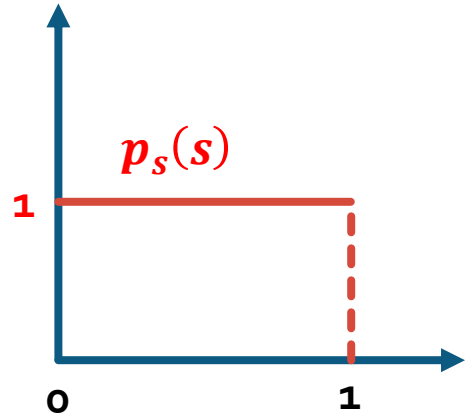
If we assume that $p_s(s) =$ ***Uniform[0, 1]*** and $A = 2$**,** then x is **distributed uniformly** in the **interval** $[0, 2]$**.** More **concretely,** we have**:**



$$p_s(s) = 1\{0 \leq s \leq 1\}$$

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

$$p_x(x) = \left|A^{-1}\right| * p_s(Wx)$$



INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

LINEAR TRANSFORMATIONS:

More generally, if $\boldsymbol{s}$ is a **vector-valued distribution** with **density** $\boldsymbol{p_s}$, and $\boldsymbol{x = As}$ for a **square**, **invertible** matrix $\boldsymbol{A}$, then the **density** of $\boldsymbol{x}$ is given by **($\boldsymbol{W = A^{-1}}$)**.

$$\boldsymbol{p_x(x) = p_s(Wx) \cdot |W|}$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

We suppose that the **distribution** of **each source** s_j is **given** by a **density** p_s, and that the **joint distribution** of the **sources** s is **given by:**

$$p(s) = \prod_{j=1}^{n} p_s(s_j)$$

Therefore, we **assume** that the **sources** are **independent**.

Substituting the **derivation** of **linear transformations** $p_x(x) = p_s(Wx) \cdot |W|$, we **obtain** the **distribution** of x (microphones)**:**

$$p(x) = \prod_{j=1}^{n} p_s(w^T{}_j x) \cdot |W|$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

Given a **real-valued random variable** z, its **cumulative distribution function** (**cdf**) F is defined by:

$$F(z_0) = P(z \leq z_0) = \int_{-\infty}^{z_0} p_z(z)dz$$

and the **density** is the **derivative** of the **cdf**: $p_z(z) = F'(z)$.

Thus, to **specify** a **density** for the s_i's, we need to **define** a **cdf** for it. **Recalling** that a **cdf** has to be a **monotonic function** that **increases** from **zero** to **one**.

A **good choice** (we **cannot pick** a **Gaussian**) is the **sigmoid function**:

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

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

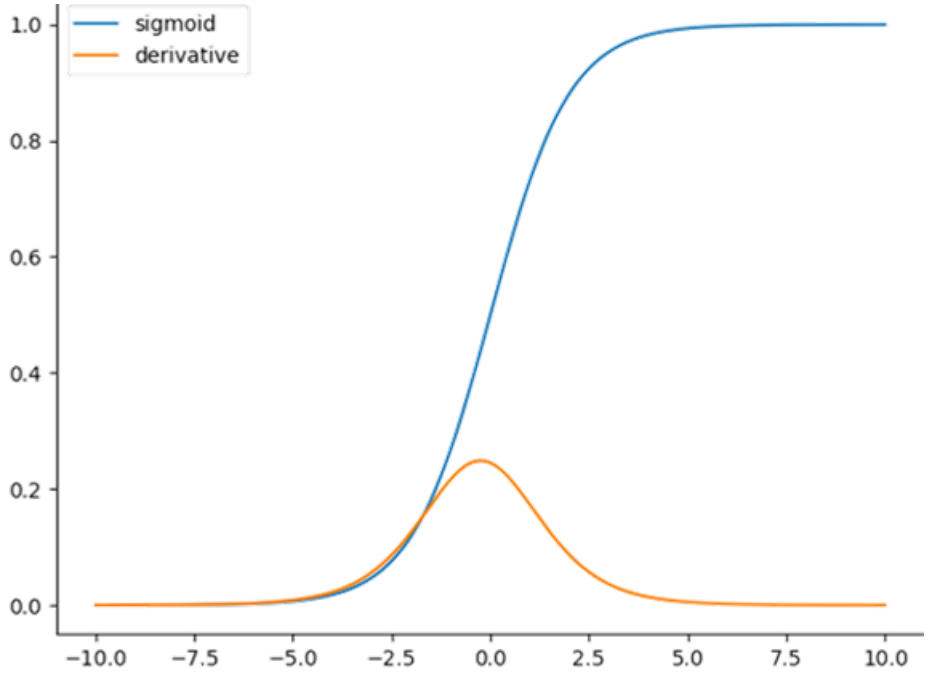
DERIVATION:

Therefore, we have that the **density** of the **sources** $\boldsymbol{p_s(s)}$ is **defined** as follows:

$$\boldsymbol{p_s(s) = g'(z) = g(z)(1 - g(z))}$$

An important note is that **if** you **have prior knowledge** that the **sources' densities** take a **certain form**, then it is a **good idea** to **substitute** that in **here**.

In the **absence of such knowledge**, the **sigmoid function** can be thought of as a **reasonable default** that **seems** to **work well** for **many problems.**



INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

The **log-likelihood** can be **represented** as **follows**:

$$p\left(x^{(i)}\right) = \log \prod_{j=1}^{n} p_s\left(w^T{}_j x^{(i)}\right) \cdot |W|$$

$$l(W) = \log \prod_{i=1}^{m} \prod_{j=1}^{n} p_s\left(w^T{}_j x^{(i)}\right) \cdot |W|$$

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

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

The **objective** is to **compute** the **unmixing matrix** W that **maximizes** the **likelihood**:

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

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} log\, g(w^T{}_j x^{(i)})(1 - g\left(w^T{}_j x^{(i)}\right)) + log|W|\right)$$

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} log\, g(w^T{}_j x^{(i)}) + \log\left(1 - g\left(w^T{}_j x^{(i)}\right)\right)\right) + mn\, log|W|$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} log\frac{1}{1+e^{-w^T{}_j x^{(i)}}} + \log\left(1-\frac{1}{1+e^{-w^T{}_j x^{(i)}}}\right)\right) + mn\,log|W|$$

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} log\,1 - \log\left(1+e^{-w^T{}_j x^{(i)}}\right) + \log\left(\frac{1+e^{-w^T{}_j x^{(i)}}-1}{1+e^{-w^T{}_j x^{(i)}}}\right)\right) + mn\,log|W|$$

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} -log\left(1+e^{-w^T{}_j x^{(i)}}\right) + log\left(e^{-w^T{}_j x^{(i)}}\right) - \log\left(1+e^{-w^T{}_j x^{(i)}}\right)\right) + mn\,log|W|$$

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} -2\,log\left(1+e^{-w^T{}_j x^{(i)}}\right) + log\left(e^{-w^T{}_j x^{(i)}}\right)\right) + mn\,log|W|$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

$$l(W) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} -2\,log\left(1 + e^{-w^T{}_j x^{(i)}}\right) + log\left(e^{-w^T{}_j x^{(i)}}\right)\right) + mn\,log|W|$$

$$l(W) = \left(\sum_{i=1}^{m}\left(\sum_{j=1}^{n} -2\,log\left(1 + e^{-w^T{}_j x^{(i)}}\right) - w^T{}_j x^{(i)}\right)\right) + mn\,log|W|$$

Taking the **gradient** with **respect** to W and **applying** the **following property** $\nabla_W|W| = |W|\left(W^{-1}\right)^T$:

$$\nabla_W l(W) = \left(\sum_{i=1}^{m}\left(\sum_{j=1}^{n} -2\frac{-x^{(i)}e^{-w^T{}_j x^{(i)}}}{1 + e^{-w^T{}_j x^{(i)}}} - x^{(i)}\right)\right) + mn\left(\frac{1}{|W|}\right)|W|\left(W^{-1}\right)^T$$

$$\nabla_W l(W) = \left(\sum_{i=1}^{m}\left(\sum_{j=1}^{n} 2\frac{e^{-w^T{}_j x^{(i)}}}{1 + e^{-w^T{}_j x^{(i)}}}x^{(i)} - x^{(i)}\right)\right) + mn\left(W^{-1}\right)^T$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

$$\nabla_W l(W) = \left(\sum_{i=1}^{m} \left(\sum_{j=1}^{n} 2 \frac{e^{-w^T{}_j x^{(i)}}}{1 + e^{-w^T{}_j x^{(i)}}} x^{(i)} - x^{(i)} \right) \right) + mn\left(W^{-1}\right)^T$$

$$\nabla_W l(W) = \left(\sum_{i=1}^{m} \left(\sum_{j=1}^{n} 2 \frac{e^{-w^T{}_j x^{(i)}}}{1 + e^{-w^T{}_j x^{(i)}}} - 1 \right) x^{(i)} \right) + mn\left(W^{-1}\right)^T$$

Substituting back the **sigmoid function** $g\left(w^T{}_j x^{(i)}\right) = \frac{e^{-w^T{}_j x^{(i)}}}{1+e^{-w^T{}_j x^{(i)}}}$**:**

$$\nabla_W l(W) = \left(\sum_{i=1}^{m} \left(\sum_{j=1}^{n} 2g\left(w^T{}_j x^{(i)}\right) - 1 \right) x^{(i)} \right) + mn\left(W^{-1}\right)^T$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

$$\nabla_W l(W) = \left(\sum_{i=1}^{m} \left(\sum_{j=1}^{n} 2g\left(w^T{}_j x^{(i)}\right) - 1 \right) x^{(i)} \right) + mn\left(W^{-1}\right)^T$$

We **know** that **this** can be **expressed** as a **dot product:**

$$\nabla_W l(W) = \sum_{i=1}^{m} \begin{bmatrix} 2g\left(w^T{}_1 x^{(i)}\right) - 1 \\ 2g\left(w^T{}_2 x^{(i)}\right) - 1 \\ \vdots \\ 2g\left(w^T{}_n x^{(i)}\right) - 1 \end{bmatrix} x^{(i)T} + mn\left(W^{-1}\right)^T$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

DERIVATION:

Now, we can **compute W** for a **single training example** using **stochastic gradient descent**:

$$W := W - \alpha \nabla_W l(W)$$

$$W := W - \alpha \left(\begin{bmatrix} 2g\left(w^T{}_1 x^{(i)}\right) - 1 \\ 2g\left(w^T{}_2 x^{(i)}\right) - 1 \\ \vdots \\ 2g\left(w^T{}_n x^{(i)}\right) - 1 \end{bmatrix} x^{(i)^T} + n\left(W^{-1}\right)^T \right)$$

$$W := W + \alpha \left(\begin{bmatrix} 1 - 2g\left(w^T{}_1 x^{(i)}\right) \\ 1 - 2g\left(w^T{}_2 x^{(i)}\right) \\ \vdots \\ 1 - 2g\left(w^T{}_n x^{(i)}\right) \end{bmatrix} x^{(i)^T} + \left(W^{-1}\right)^T \right)$$

INDEPENDENT COMPONENT ANALYSIS

T H E A L G O R I T H M

SUMMARY:

1. **Initialize** an **unmixing matrix W**.
2. **Compute** $\boldsymbol{W}$ using **stochastic gradient descent** until **convergence**.

$$\boldsymbol{W} \coloneqq \boldsymbol{W} - \boldsymbol{\alpha}\boldsymbol{\nabla}_{\boldsymbol{W}}\boldsymbol{l}(\boldsymbol{W})$$

3. **Compute** $\boldsymbol{s}^{(\boldsymbol{i})} = \boldsymbol{W}\boldsymbol{x}^{(\boldsymbol{i})}$ to **recover** the **original sources**.