MATH 829: Introduction to Data Mining and Analysis
Independent component analysis

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

April 22, 2016

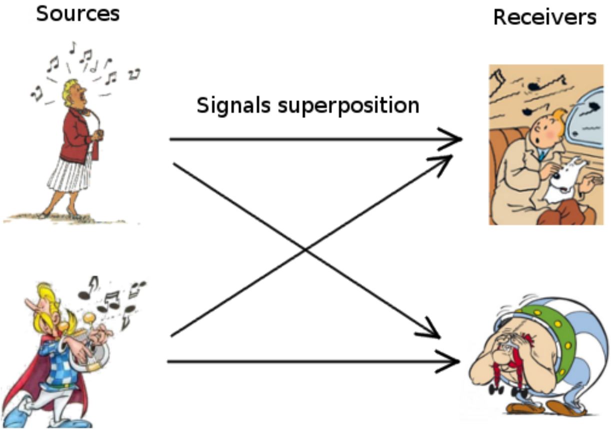
Motivation

- **Blind signal separation:** separation of a mixture of source signals, without (or with very little) information about the sources and the mixing process.

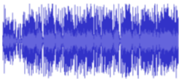
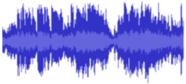
Motivation

- **Blind signal separation:** separation of a mixture of source signals, without (or with very little) information about the sources and the mixing process.
- **Example (the cocktail party problem):** isolate a single conversation in a noisy room with many people talking.

- **Blind signal separation:** separation of a mixture of source signals, without (or with very little) information about the sources and the mixing process.
- **Example (the cocktail party problem):** isolate a single conversation in a noisy room with many people talking.



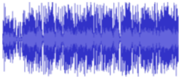
Mathematical formulation

$s_1(t)$  $s_2(t)$ 

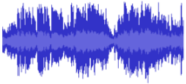
$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$
$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t)$$

Mathematical formulation

$s_1(t)$



$s_2(t)$

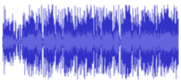
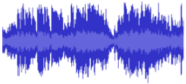


$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$

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

- We have $x(t) = As(t)$, $t = 1, \ldots, T$.

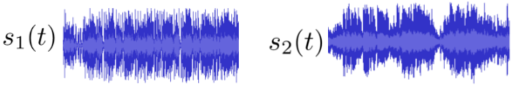
Mathematical formulation

$s_1(t)$  $s_2(t)$ 

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$
$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t)$$

- We have $x(t) = As(t)$, $t = 1, \ldots, T$.
- We observe $x(t)$.

Mathematical formulation

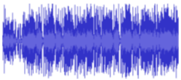
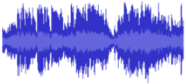


$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$

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

- We have $x(t) = As(t)$, $t = 1, \ldots, T$.
- We observe $x(t)$.
- We don't know what A is (mixing matrix).

Mathematical formulation

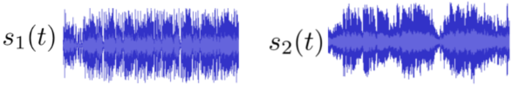
$s_1(t)$  $s_2(t)$ 

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

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

- We have $x(t) = As(t)$, $t = 1, \ldots, T$.
- We observe $x(t)$.
- We don't know what A is (mixing matrix).
- We don't observe $s(t)$.

Mathematical formulation



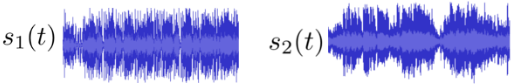
$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$

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

- We have $x(t) = As(t)$, $t = 1, \ldots, T$.
- We observe $x(t)$.
- We don't know what A is (mixing matrix).
- We don't observe $s(t)$.

We want to recover $s(t)$ (and/or A).

Mathematical formulation

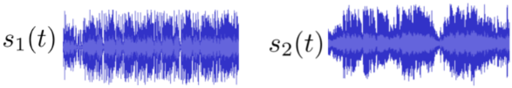


$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$
$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t)$$

- We have $x(t) = As(t)$, $t = 1, \dots, T$.
- We observe $x(t)$.
- We don't know what A is (mixing matrix).
- We don't observe $s(t)$.

We want to recover $s(t)$ (and/or A).

- Current formulation is ill-posed: there are multiple ways of mixing signals to get the output.



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

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

- We have $x(t) = As(t)$, $t = 1, \ldots, T$.
- We observe $x(t)$.
- We don't know what A is (mixing matrix).
- We don't observe $s(t)$.

We want to recover $s(t)$ (and/or A).

- Current formulation is ill-posed: there are multiple ways of mixing signals to get the output.
- We will seek a solution where the components of s are *as independent as possible*.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.)

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.
- Then $x \sim N(\mathbf{0}_{2\times 1}, AA^T)$.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.
- Then $x \sim N(\mathbf{0}_{2\times 1}, AA^T)$.
- Let U be an orthogonal matrix, i.e., $UU^T = U^TU = I$.

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.
- Then $x \sim N(\mathbf{0}_{2\times 1}, AA^T)$.
- Let U be an orthogonal matrix, i.e., $UU^T = U^T U = I$.
- Let $A' = AU$.

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.
- Then $x \sim N(\mathbf{0}_{2\times 1}, AA^T)$.
- Let U be an orthogonal matrix, i.e., $UU^T = U^TU = I$.
- Let $A' = AU$.
- Then $x' = A's \sim N(\mathbf{0}_{2\times 1}, A'A'^T) = N(\mathbf{0}_{2\times 1}, AUU^TA^T) = N(\mathbf{0}_{2\times 1}, AA^T)$.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.
- Then $x \sim N(\mathbf{0}_{2\times 1}, AA^T)$.
- Let U be an orthogonal matrix, i.e., $UU^T = U^T U = I$.
- Let $A' = AU$.
- Then $x' = A's \sim N(\mathbf{0}_{2\times 1}, A'A'^T) = N(\mathbf{0}_{2\times 1}, AUU^T A^T) = N(\mathbf{0}_{2\times 1}, AA^T)$.

Thus, there is no way to statistically differentiate if x was obtained from the mixing matrix A or A'.

Assumptions

Note: Signals can only be recovered up to

1. **Permutations:** we can always permute the s_i's and the row/columns of A to obtain new solutions.
2. **Scaling:** we can always rescale the s_i's and rescale the coefficients in A.

(Not a big deal in most applications.) Other problems?

Problem with Gaussian data:

- Suppose $s \sim N(\mathbf{0}_{2\times 1}, I_{2\times 2})$ (independent Gaussian sources).
- Let $x = As$ where $A \in \mathbb{R}^{2\times 2}$.
- Then $x \sim N(\mathbf{0}_{2\times 1}, AA^T)$.
- Let U be an orthogonal matrix, i.e., $UU^T = U^T U = I$.
- Let $A' = AU$.
- Then $x' = A's \sim N(\mathbf{0}_{2\times 1}, A'A'^T) = N(\mathbf{0}_{2\times 1}, AUU^T A^T) = N(\mathbf{0}_{2\times 1}, AA^T)$.

Thus, there is no way to statistically differentiate if x was obtained from the mixing matrix A or A'.

We will therefore assume the sources are **not** Gaussian.

Independence of the sources

- We seek sources that are *as independent as possible*.

Independence of the sources

- We seek sources that are *as independent as possible*.
- Multiple ways to *measure* independence. For example:

Independence of the sources

- We seek sources that are *as independent as possible*.
- Multiple ways to *measure* independence. For example:
  1. Minimization of mutual information.

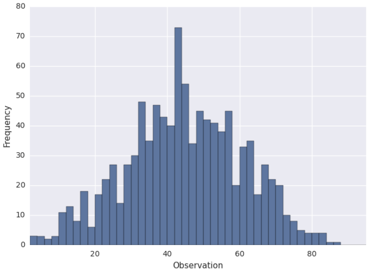
Independence of the sources

- We seek sources that are *as independent as possible*.
- Multiple ways to *measure* independence. For example:
  1. Minimization of mutual information.
  2. Maximization of non-Gaussianity measures (negentropy, kurtosis, etc.).

Independence of the sources

- We seek sources that are *as independent as possible*.
- Multiple ways to *measure* independence. For example:
  1. Minimization of mutual information.
  2. Maximization of non-Gaussianity measures (negentropy, kurtosis, etc.).

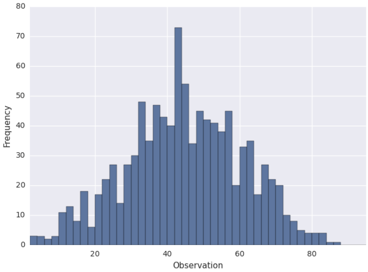
Motivation for (2) comes from the central limit theorem: a sum of independent random variables should be "more Gaussian".



Independence of the sources

- We seek sources that are *as independent as possible*.
- Multiple ways to *measure* independence. For example:
  1. Minimization of mutual information.
  2. Maximization of non-Gaussianity measures (negentropy, kurtosis, etc.).

Motivation for (2) comes from the central limit theorem: a sum of independent random variables should be "more Gaussian".



To explain the above notions, we briefly discuss some concepts from *information theory*.

Entropy of a random variable

- Let X be a random variable taking values $x_1, \ldots, x_N$ with probabilities $P(X = x_i) = p_i$.

Entropy of a random variable

- Let X be a random variable taking values $x_1, \dots, x_N$ with probabilities $P(X = x_i) = p_i$.
- The *entropy* of X is given by:

$$H(X) = E(-\log p) = -\sum_{i=1}^{N} p_i \log p_i.$$

(usually, we take the $\log$ in base 2).

Entropy of a random variable

- Let X be a random variable taking values $x_1, \ldots, x_N$ with probabilities $P(X = x_i) = p_i$.
- The *entropy* of X is given by:

$$H(X) = E(-\log p) = -\sum_{i=1}^{N} p_i \log p_i.$$

(usually, we take the $\log$ in base 2).

- Similarly, if X is a continuous random variable with density $f(x)$, we define:

$$H(X) = -\int f(x) \log f(x) \; dx$$

Entropy of a random variable

- Let X be a random variable taking values $x_1, \ldots, x_N$ with probabilities $P(X = x_i) = p_i$.
- The *entropy* of X is given by:

$$H(X) = E(-\log p) = -\sum_{i=1}^{N} p_i \log p_i.$$

(usually, we take the $\log$ in base 2).

- Similarly, if X is a continuous random variable with density $f(x)$, we define:

$$H(X) = -\int f(x) \log f(x) \; dx$$

The entropy is a measure of the uncertainty or complexity of a random variable.

Entropy of a random variable

- Let X be a random variable taking values $x_1, \dots, x_N$ with probabilities $P(X = x_i) = p_i$.
- The *entropy* of X is given by:

$$H(X) = E(-\log p) = -\sum_{i=1}^{N} p_i \log p_i.$$

(usually, we take the $\log$ in base 2).

- Similarly, if X is a continuous random variable with density $f(x)$, we define:

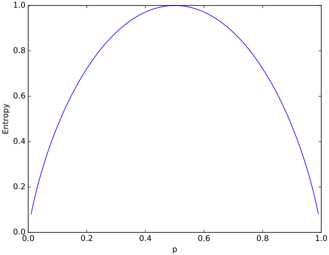
$$H(X) = -\int f(x) \log f(x) \; dx$$

The entropy is a measure of the uncertainty or complexity of a random variable.

Example: If X is a (discrete) uniform on $\{1, \dots, N\}$, then

$$H(X) = -\sum_{i=1}^{N} \frac{1}{N} \log\left(\frac{1}{N}\right) = \log N.$$

Example: $X \sim \mathrm{Bernoulli}(p)$, i.e., $P(X = 1) = p$, $P(X = 0) = 1 - p$. The more "uncertain" the outcome is, the larger the entropy.



Entropy and information

We would like to define a measure of *information* $I(p)$ of an event occurring with probability p. This functions should satisfy:

Entropy and information

We would like to define a measure of *information* $I(p)$ of an event occurring with probability p. This functions should satisfy:

- $I(p) \geq 0$.
- $I(1) = 0$ (the information gained from observing a certain event is 0).
- $I(p_1 p_2) = I(p_1) + I(p_2)$ (information gained from observing two independent event is sum of information).
- I should be continuous and monotonic.

Entropy and information

We would like to define a measure of *information* $I(p)$ of an event occurring with probability p. This functions should satisfy:

- $I(p) \geq 0$.
- $I(1) = 0$ (the information gained from observing a certain event is 0).
- $I(p_1 p_2) = I(p_1) + I(p_2)$ (information gained from observing two independent event is sum of information).
- I should be continuous and monotonic.

The above properties imply $I(p) = \log_b \frac{1}{p}$ for some base b.

Entropy and information

We would like to define a measure of *information* $I(p)$ of an event occurring with probability p. This functions should satisfy:

- $I(p) \geq 0$.
- $I(1) = 0$ (the information gained from observing a certain event is 0).
- $I(p_1 p_2) = I(p_1) + I(p_2)$ (information gained from observing two independent event is sum of information).
- I should be continuous and monotonic.

The above properties imply $I(p) = \log_b \frac{1}{p}$ for some base b.

The entropy of X is the **average information** "contained" in X:

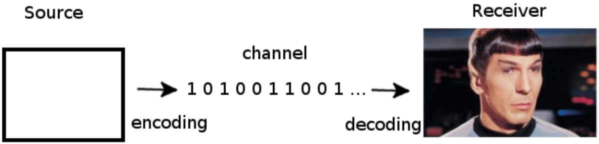
$$H(X) = \sum_{i=1}^{N} I(p_i) p_i.$$

Entropy and communication

- Suppose we can only transmit 0s and 1s.
- We need to encode our message (e.g. choose a code for each letter).
- How efficiently can we encore the message?

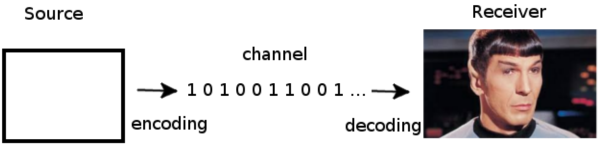
Entropy and communication

- Suppose we can only transmit 0s and 1s.
- We need to encode our message (e.g. choose a code for each letter).
- How efficiently can we encore the message?



Entropy and communication

- Suppose we can only transmit 0s and 1s.
- We need to encode our message (e.g. choose a code for each letter).
- How efficiently can we encore the message?



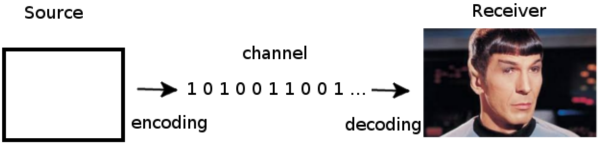
Example: Our source sends the letters A, B, C, D. Each letter is equally likely to be transmitted.

$$A \to 00 \qquad C \to 10$$
$$B \to 01 \qquad D \to 11$$

We send on average (actually, exactly!) 2 bits per symbol.

Entropy and communication

- Suppose we can only transmit 0s and 1s.
- We need to encode our message (e.g. choose a code for each letter).
- How efficiently can we encore the message?



Example: Our source sends the letters A, B, C, D. Each letter is equally likely to be transmitted.

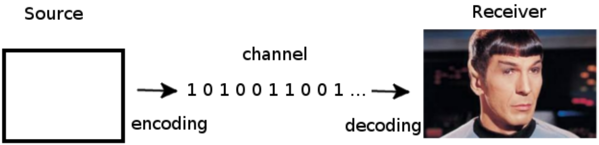
$$A \to 00 \qquad C \to 10$$
$$B \to 01 \qquad D \to 11$$

We send on average (actually, exactly!) 2 bits per symbol.

- If the symbols an not equally likely, it is not hard to see that one can do better (i.e., send less bits per symbol on average).

Entropy and communication

- Suppose we can only transmit 0s and 1s.
- We need to encode our message (e.g. choose a code for each letter).
- How efficiently can we encore the message?



Example: Our source sends the letters A, B, C, D. Each letter is equally likely to be transmitted.

$A \to 00$ $\quad C \to 10$

$B \to 01$ $\quad D \to 11$

We send on average (actually, exactly!) 2 bits per symbol.

- If the symbols an not equally likely, it is not hard to see that one can do better (i.e., send less bits per symbol on average).
- The entropy provides a **lower bound** on the average number of bits required per symbol.

Kullback–Leibler divergence

Given two (discrete) probability distributions P and Q, we define the *Kullback–Leibler divergence* by

$$D_{\mathrm{KL}}(P||Q) := \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Kullback–Leibler divergence

Given two (discrete) probability distributions P and Q, we define the *Kullback–Leibler divergence* by

$$D_{\mathrm{KL}}(P||Q) := \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Similarly, when P and Q are continuous with densities $p(x)$ and $q(x)$ respectively, we define

$$D_{\mathrm{KL}}(P||Q) := \int p(x) \log \frac{p(x)}{q(x)} \, dx.$$

Kullback–Leibler divergence

Given two (discrete) probability distributions P and Q, we define the *Kullback–Leibler divergence* by

$$D_{\mathrm{KL}}(P||Q) := \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Similarly, when P and Q are continuous with densities $p(x)$ and $q(x)$ respectively, we define

$$D_{\mathrm{KL}}(P||Q) := \int p(x) \log \frac{p(x)}{q(x)}\ dx.$$

Intuitive interpretation:

- A source send symbols with distribution P.

Kullback–Leibler divergence

Given two (discrete) probability distributions P and Q, we define the *Kullback–Leibler divergence* by

$$D_{\mathrm{KL}}(P||Q) := \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Similarly, when P and Q are continuous with densities $p(x)$ and $q(x)$ respectively, we define

$$D_{\mathrm{KL}}(P||Q) := \int p(x) \log \frac{p(x)}{q(x)}\ dx.$$

Intuitive interpretation:

- A source send symbols with distribution P.
- We encode the messages as if the source had distribution Q.

Kullback–Leibler divergence

Given two (discrete) probability distributions P and Q, we define the *Kullback–Leibler divergence* by

$$D_{\mathrm{KL}}(P||Q) := \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Similarly, when P and Q are continuous with densities $p(x)$ and $q(x)$ respectively, we define

$$D_{\mathrm{KL}}(P||Q) := \int p(x) \log \frac{p(x)}{q(x)}\, dx.$$

Intuitive interpretation:

- A source send symbols with distribution P.
- We encode the messages as if the source had distribution Q.
- $D_{\mathrm{KL}}(P||Q)$ is the number of supplementary bits per symbol that we send for not using the "right" distribution.

Kullback–Leibler divergence

Given two (discrete) probability distributions P and Q, we define the *Kullback–Leibler divergence* by

$$D_{\mathrm{KL}}(P||Q) := \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Similarly, when P and Q are continuous with densities $p(x)$ and $q(x)$ respectively, we define

$$D_{\mathrm{KL}}(P||Q) := \int p(x) \log \frac{p(x)}{q(x)} \, dx.$$

Intuitive interpretation:

- A source send symbols with distribution P.
- We encode the messages as if the source had distribution Q.
- $D_{\mathrm{KL}}(P||Q)$ is the number of supplementary bits per symbol that we send for not using the "right" distribution.

The KL divergence is used as a measure of distance between distributions (note however that $D_{\mathrm{KL}}(P||Q) \neq D_{\mathrm{KL}}(Q||P)$ in general).

Mutual information

- $(X_1, \ldots, X_n)$ random vector with distribution $p(x_1, \ldots, x_n)$.

Mutual information

- $(X_1, \ldots, X_n)$ random vector with distribution $p(x_1, \ldots, x_n)$.
- Let $p(x_1), \ldots, p(x_n)$ denote the marginals of p (i.e., the distribution of each variable X_i).

Mutual information

- $(X_1, \ldots, X_n)$ random vector with distribution $p(x_1, \ldots, x_n)$.
- Let $p(x_1), \ldots, p(x_n)$ denote the marginals of p (i.e., the distribution of each variable X_i).
- Let $(Y_1, \ldots, Y_n)$ have distribution $p(x_1)p(x_2)\ldots p(x_n)$ (so Y_i has the same distribution as X_i, but the Y_is are independent).

Mutual information

- $(X_1, \ldots, X_n)$ random vector with distribution $p(x_1, \ldots, x_n)$.
- Let $p(x_1), \ldots, p(x_n)$ denote the marginals of p (i.e., the distribution of each variable X_i).
- Let $(Y_1, \ldots, Y_n)$ have distribution $p(x_1)p(x_2)\ldots p(x_n)$ (so Y_i has the same distribution as X_i, but the Y_is are independent).

The *mutual information* of $(X_1, \ldots, X_n)$ is given by

$$I(X_1, \ldots, X_n) = D_{\mathrm{KL}}(p(x_1, \ldots, x_n) || p(x_1) \ldots p(x_n)).$$

- We have $I(X, Y) = 0$ if and only if X, Y are independent.

Mutual information

- $(X_1, \ldots, X_n)$ random vector with distribution $p(x_1, \ldots, x_n)$.
- Let $p(x_1), \ldots, p(x_n)$ denote the marginals of p (i.e., the distribution of each variable X_i).
- Let $(Y_1, \ldots, Y_n)$ have distribution $p(x_1)p(x_2)\ldots p(x_n)$ (so Y_i has the same distribution as X_i, but the Y_is are independent).

The *mutual information* of $(X_1, \ldots, X_n)$ is given by

$$I(X_1, \ldots, X_n) = D_{\mathrm{KL}}(p(x_1, \ldots, x_n) || p(x_1) \ldots p(x_n)).$$

- We have $I(X, Y) = 0$ if and only if X, Y are independent.
- Therefore, $I(X_1, \ldots, X_n)$ provides a numerical measure of how independent random variables are.

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\mathrm{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\mathrm{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

  - Measures the "propensity to produce outliers".

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\operatorname{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

  - Measures the "propensity to produce outliers".
  - The Gaussian distribution has kurtosis equal to 3.

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\mathrm{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

  - Measures the "propensity to produce outliers".
  - The Gaussian distribution has kurtosis equal to 3.
  - Can thus use the "excess kurtosis" $\mathrm{Kurt}(X) - 3$ to test for "non-Gaussianity".

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\mathrm{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

  - Measures the "propensity to produce outliers".
  - The Gaussian distribution has kurtosis equal to 3.
  - Can thus use the "excess kurtosis" $\mathrm{Kurt}(X) - 3$ to test for "non-Gaussianity".

- The **negentropy** of a random variable X is given by

$$J(X) := H(X_{\mathrm{gauss}}) - H(X),$$

where X_{gauss} is a Gaussian random variable with the same mean and variance as X.

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\mathrm{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

  - Measures the "propensity to produce outliers".
  - The Gaussian distribution has kurtosis equal to 3.
  - Can thus use the "excess kurtosis" $\mathrm{Kurt}(X) - 3$ to test for "non-Gaussianity".

- The **negentropy** of a random variable X is given by

$$J(X) := H(X_{\mathrm{gauss}}) - H(X),$$

where X_{gauss} is a Gaussian random variable with the same mean and variance as X.

  - Motivated by the fact that the Gaussian distribution has the largest entropy among all continuous distributions with a given mean and variance.

Measures of non-Gaussianity

- The **kurtosis** (from greek κυρτός, "curved") of a random variable with mean $\mu = E(X)$ is given by

$$\mathrm{Kurt}(X) := \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2}.$$

  - Measures the "propensity to produce outliers".
  - The Gaussian distribution has kurtosis equal to 3.
  - Can thus use the "excess kurtosis" $\mathrm{Kurt}(X) - 3$ to test for "non-Gaussianity".

- The **negentropy** of a random variable X is given by

$$J(X) := H(X_{\mathrm{gauss}}) - H(X),$$

where X_{gauss} is a Gaussian random variable with the same mean and variance as X.

  - Motivated by the fact that the Gaussian distribution has the largest entropy among all continuous distributions with a given mean and variance.
  - Therefore, a variable that is "far from a Gaussian" should have a larger negentropy.

The FastICA algorithm

- FastICA (Hyvärinen, 1999) is an efficient and popular algorithm for computing independent components.

The FastICA algorithm

- FastICA (Hyvärinen, 1999) is an efficient and popular algorithm for computing independent components.
- Finds linear combinations maximizing an approximation of the negentropy.

The FastICA algorithm

- FastICA (Hyvärinen, 1999) is an efficient and popular algorithm for computing independent components.
- Finds linear combinations maximizing an approximation of the negentropy.
- The negentropy is replaced by the approximation

$$J(X) \approx [E(G(X)) - E(G(X_{\text{gauss}}))]^2,$$

where $G(x) = \log\cosh(x)$.

Whitening the data

Before the FastICA algorithm is applied, the data needs to be *prewhitened*.

Whitening the data

Before the FastICA algorithm is applied, the data needs to be *prewhitened*.

- Let $X \in \mathbb{R}^{N \times M}$ be the data matrix.

Whitening the data

Before the FastICA algorithm is applied, the data needs to be *prewhitened*.

- Let $X \in \mathbb{R}^{N \times M}$ be the data matrix.
- First *center* the rows of X:

$$x_{ij} \leftarrow x_{ij} - \frac{1}{M} \sum_k x_{ik}.$$

Whitening the data

Before the FastICA algorithm is applied, the data needs to be *prewhitened*.

- Let $X \in \mathbb{R}^{N \times M}$ be the data matrix.
- First *center* the rows of X:

$$x_{ij} \leftarrow x_{ij} - \frac{1}{M} \sum_k x_{ik}.$$

- Next, we want the linearly transform the rows of X so that they become *uncorrelated*. We seek a linear transformation $L : \mathbb{R}^{N \times M} \to \mathbb{R}^{N \times M}$ such that

$$\frac{1}{M} L(x) L(x)^T = I_{N \times N}.$$

Whitening the data

Before the FastICA algorithm is applied, the data needs to be *prewhitened*.

- Let $X \in \mathbb{R}^{N\times M}$ be the data matrix.
- First *center* the rows of X:

$$x_{ij} \leftarrow x_{ij} - \frac{1}{M}\sum_k x_{ik}.$$

- Next, we want the linearly transform the rows of X so that they become *uncorrelated*. We seek a linear transformation $L : \mathbb{R}^{N\times M} \to \mathbb{R}^{N\times M}$ such that

$$\frac{1}{M}L(x)L(x)^T = I_{N\times N}.$$

  This is easily achieved using the eigendecomposition of the covariance matrix of the centered data X:

$$\frac{1}{M}XX^T = UDU^T.$$

Whitening the data

Before the FastICA algorithm is applied, the data needs to be *prewhitened*.

- Let $X \in \mathbb{R}^{N \times M}$ be the data matrix.
- First *center* the rows of X:

$$x_{ij} \leftarrow x_{ij} - \frac{1}{M} \sum_k x_{ik}.$$

- Next, we want the linearly transform the rows of X so that they become *uncorrelated*. We seek a linear transformation $L : \mathbb{R}^{N \times M} \to \mathbb{R}^{N \times M}$ such that

$$\frac{1}{M} L(x) L(x)^T = I_{N \times N}.$$

  This is easily achieved using the eigendecomposition of the covariance matrix of the centered data X:

$$\frac{1}{M} X X^T = U D U^T.$$

- Define the *whitened data matrix* by

$$X_{\text{white}} := U D^{-1/2} U^T X.$$

The FastICA algorithm

We want to extract independent components of the form $w^T X$ where $w \in \mathbb{R}^N$.

The FastICA algorithm

We want to extract independent components of the form $w^T X$ where $w \in \mathbb{R}^N$.

The FastICA algorithm:

- Find a first direction w_1 maximizing the (approximation of) the negentropy (can use a fixed point method).
- Estimate a second direction $w_2 \perp w_1$ maximizing the (approximation of) the negentropy.
- etc..

Python - example

We mix two sound files, and recover them using ICA.

```
import scipy.io.wavfile
import numpy as np

rate, data1 = scipy.io.wavfile.read('daft-punk.wav')
rate2, data2 = scipy.io.wavfile.read('weather.wav')

mix1 = np.int16(0.3*data1+0.5*data2)[:,0]
mix2 = np.int16(0.2*data1+0.4*data2)[:,0]

scipy.io.wavfile.write('./out/mix1.wav',rate,mix1)
scipy.io.wavfile.write('./out/mix2.wav',rate,mix2)

from sklearn.decomposition import FastICA

ica = FastICA(n_components = 2)

X = np.vstack([mix1,mix2]).T

S_ = ica.fit_transform(X)
A_ = ica.mixing_

# Rescale components to have approximately the same mean amplitude as the first mixed signal
m = abs(mix1).mean()

m1 = abs(S_[:,0]).mean()
m2 = abs(S_[:,1]).mean()

S1 = np.int16(S_[:,0]*m/m1)
S2 = np.int16(S_[:,1]*m/m2)

scipy.io.wavfile.write('./out/estimated_source1.wav',rate,S1)
scipy.io.wavfile.write('./out/estimated_source2.wav',rate,S2)
```