Connectionist Computing COMP 30230/41390

Gianluca Pollastri
office: E0.95, Science East.
email: gianluca.pollastri@ucd.ie

Credits

- **Geoffrey Hinton, University of Toronto.**
  - **borrowed some of his slides for "Neural Networks" and "Computation in Neural Networks" courses.**

- **Ronan Reilly, NUI Maynooth.**
  - **slides from his CS4018.**
- **Paolo Frasconi, University of Florence.**
  - **slides from tutorial on Machine Learning for structured domains.**

Lecture notes on Brightspace

- Strictly confidential...

- Slim PDF version will be uploaded later, typically the same day as the lecture.

- If there is demand, I can upload onto Brightspace last year's narrated slides.. (should be very similar to this year's material)

Books

- No book covers large fractions of this course.

- Parts of chapters 4, 6, (7), 13 of Tom Mitchell's "Machine Learning"
- Parts of chapter V of Mackay's "Information Theory, Inference, and Learning Algorithms", available online at:

http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html

- Chapter 20 of Russell and Norvig's "Artificial Intelligence: A Modern Approach", also available at:

http://aima.cs.berkeley.edu/newchap20.pdf

- More materials later..

Marking

- **3 landmark papers to read, and submit a 10-line summary on Brightspace about: each worth 6-7%**
- **a connectionist model to build and play with on some sets, write a report: 30%**
- **Final Exam in the RDS (50%)**

Assignment 1

- Read the first section of the following article by Marvin Minsky:
- http://web.media.mit.edu/~minsky/papers/SymbolicVs.Connectionist.html
- down to ".. we need more research on how to combine both types of ideas."

- Submit through BrightSpace a 250 word MAX summary by October 2nd at 23:59, any time zone of your choice (Baker Island?).

- One third of a grade down every 2 days late, that is: if you deserve an A and you're 1-2 days late you get an A-, 3-4 days late a B+, etc.
- Make sure it's gone through!

From Hopfield nets to Boltzmann machine

- **Hopfield nets (attempt to) minimise an energy function:**

$$E(y) = -\frac{1}{2}\sum_{i,j} w_{ij} y_i y_j = -\frac{1}{2}\mathbf{y}^{\mathbf{T}}\mathbf{W}\mathbf{y}$$

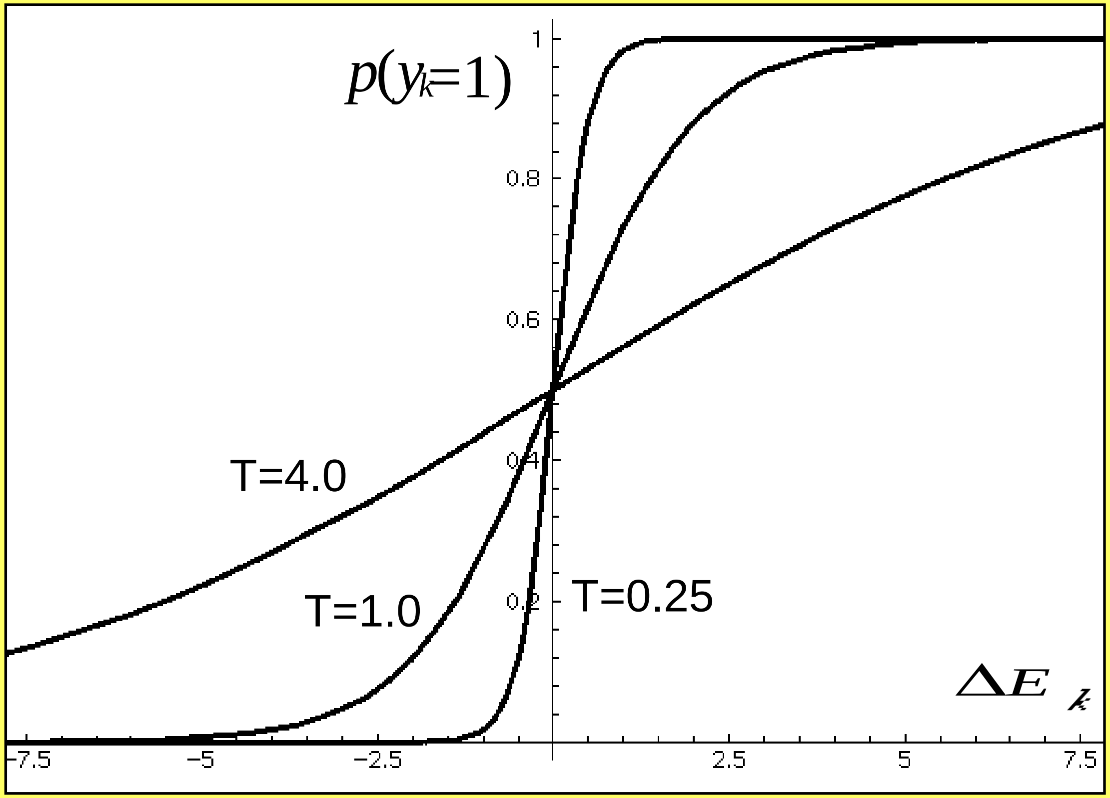
$$\mathbf{y} = (y_1, .., y_m)^T$$

$$\mathbf{W} = \begin{pmatrix} w_{11} & .. & w_{1i} & .. & w_{1n} \\ w_{j1} & .. & w_{ji} & .. & w_{jn} \\ w_{m1} & .. & w_{mi} & .. & w_{mn} \end{pmatrix}$$

Stochastic units

- The probability that a given unit is switched on is a function of the amount of energy that its change of state would contribute to the network's overall energy, and the "temperature" T.

$$\frac{exp(z_i/T)}{exp(z_i/T) + exp(-z_i/T)} = \frac{1}{1 + exp(-2z_i/T)}$$



p(yk=1)
1
0.8
0.6
0.4
0.2
T=4.0
T=1.0
T=0.25
ΔE k
-7.5
-5
-2.5
2.5
5
7.5

Implementing a probability distribution

- It can be shown that the Boltzmann machine generates configurations according to the distribution:

$$P(\mathbf{y}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} exp(\frac{1}{2}\mathbf{y}^{\mathbf{T}}\mathbf{W}\mathbf{y})$$

- Where:

$$Z(\mathbf{W}) = \sum_{\mathbf{y}} exp(\frac{1}{2}\mathbf{y}^{\mathbf{T}}\mathbf{W}\mathbf{y})$$

Learning in the Boltzmann machine

- Working on the probability distribution it is possible to devise the following learning rule:

$$\Delta w_{ij} = \eta \sum_{p=1}^{P} \left[y_i^{(p)} y_j^{(p)} - \sum_{\mathbf{y}} y_i y_j P(\mathbf{y}|\mathbf{W}) \right]$$

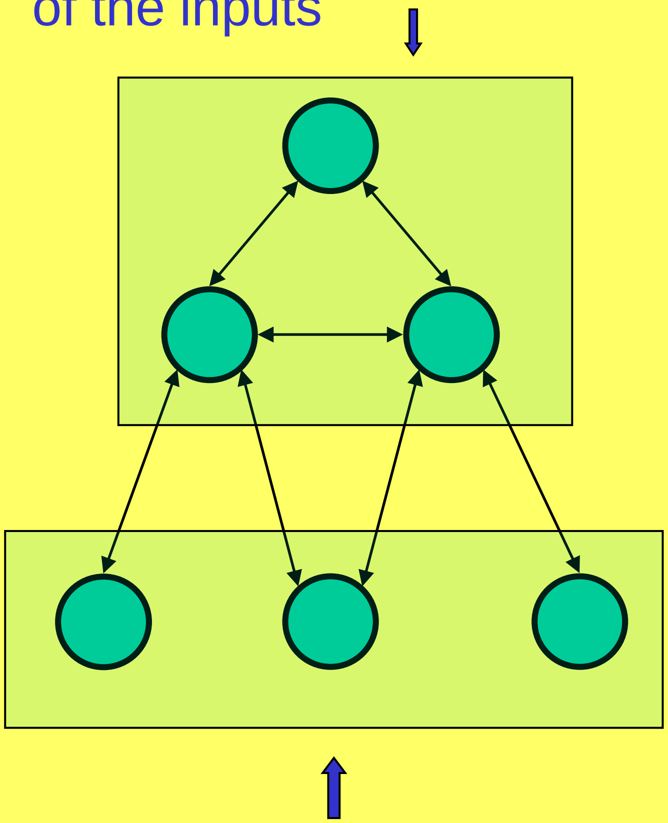
Interpretation of learning terms

- First term: the network is awake and measures the correlations in the real world.
- Second term: the network sleeps and dreams about the world using the model it has of it.
- Once dream and reality coincide learning reaches an end. It is interesting to notice that the network *unlearns* its dreams.

Hidden units for Hopfield nets/Boltzmann machines

- **Instead of using the net just to store memories, use it to construct *interpretations* of the input.**
  - **The input is represented by the visible units.**
  - **The interpretation is represented by the states of the hidden units.**
- **Higher order correlations can be represented by hidden units.**
- **More powerful model, but even harder to train.**

Hidden units. Used to represent an interpretation of the inputs



Visible units. Used to represent the inputs

Learning in the Boltzmann machine with hidden units

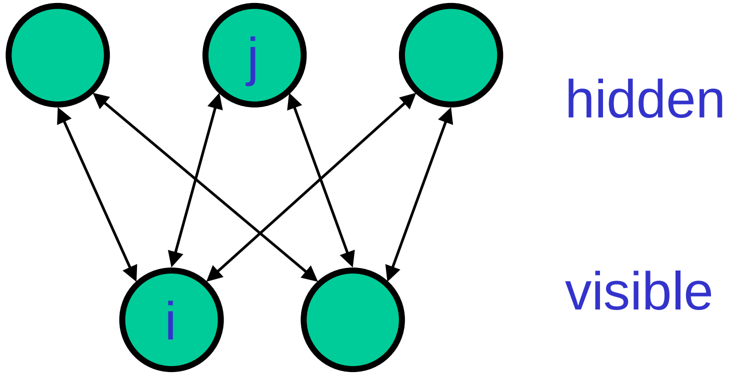
- Even in this case we have two terms in Δw_{ij}.
- The first term is the correlation between neurons i and j when the visible units are clamped to the examples.
- The second is the correlation between neurons i and j when the system is let evolve freely.

Learning in the Boltzmann machine with hidden units

- In this case both terms must be estimated by letting the network evolve many times until equilibrium, in one case with the visible units clamped, in the other freely.
- Not only: the missing units need to be estimated separately for each example.
- Can be computationally very expensive

Restricted Boltzmann Machines (RBM)

- We restrict the connectivity to make inference and learning easier.
  - Only one layer of hidden units.
  - No connections between hidden units.
- It only takes one step to reach equilibrium when the visible units are clamped..



Example 1: no hidden units

- Only two examples:
- (-1, -1, -1, -1, -1, -1)
- (1, 1, 1, 1, 1, 1)

- Train a Boltzmann machine without hidden units on them.

Example 1: BM learning

Iterate:

- **First step: increase (i,j) connection strength by an amount proportional to the correlation between bits i and j <u>of the examples</u>:**

- **Second step: let the BM machine run many times until equilibrium, measure the correlation between bits i and j <u>of the final BM states</u> and decrease (i,j) connection strength by an amount proportional to it.**

Example 1

- **Correlations in:**
- **(-1,-1,-1,-1,-1,-1) and (1,1,1,1,1,1)**

- **All the w_{ij} are 2η**

- **NOTE: We can compute these at the beginning: they don't change.**

Example 1

Learning parameters:

- **Learning rate η=0.1**
- **Temperature T=1**

"let the BM machine run <u>many times</u> <u>until equilibrium</u>":

- **many times = 1000**
- **until equilibrium = 600 neuron flips**

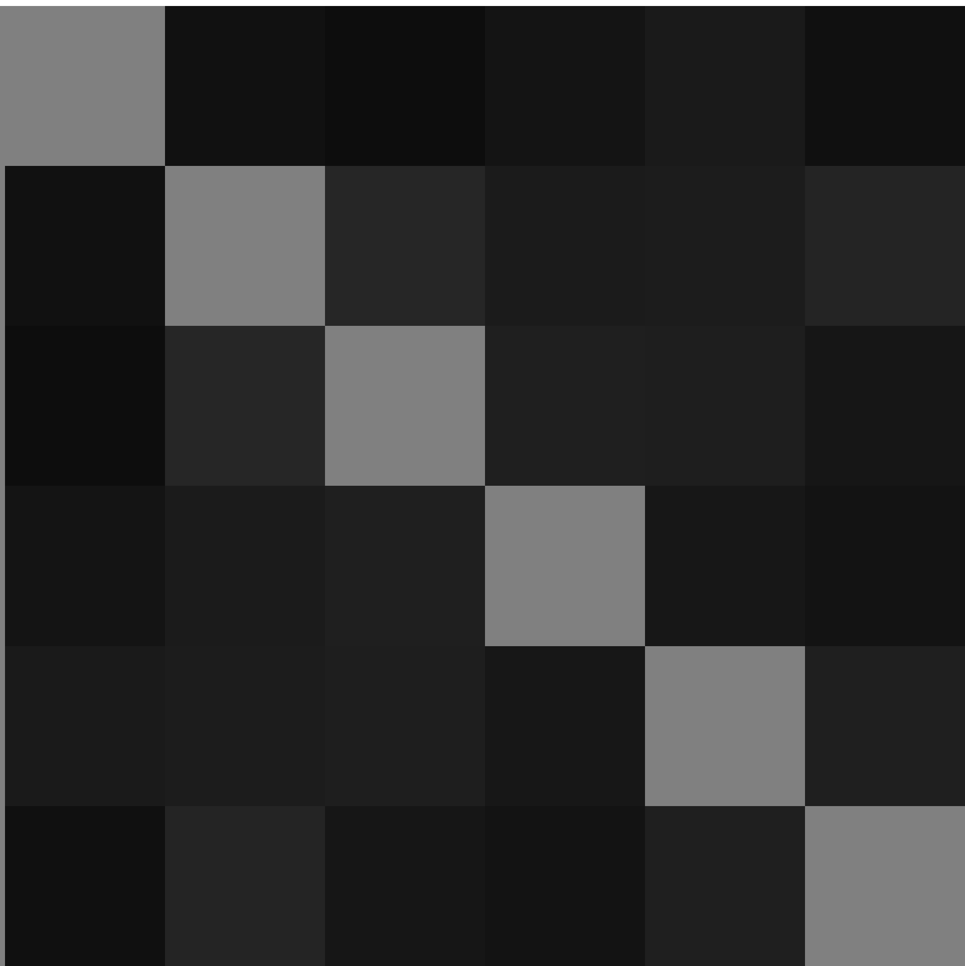
Example 1

- Summary

- Iterate:
  - 1) change all weights by +2η
  - 2) let the BM run from random starts 1000 times until it settles into y, decrease all weights by η times the correlations between the bits of y.
- until no change

Step 0

- **Reality**
- **Dream**



Step 1

- **Reality**
- **Dream**

Step 2

- **Reality**
- **Dream**

Step 3

- **Reality**
- **Dream**

Step 4

- **Reality**
- **Dream**

Step 5

- **Reality**

- **Dream**

Step 10

- **Reality**
- **Dream**

Step 20

- **Reality**
- **Dream**

Example 1

- **What happens if we let the trained BM evolve from random starts now?**

- `-1 -1 -1 -1 -1 -1`
- `-1 1 1 1 1 1`
- `1 1 1 1 1 1`
- `1 1 1 1 1 1`
- `1 1 1 1 1 1`
- `1 1 -1 1 1 1`
- `-1 -1 -1 -1 -1 -1`
- `1 1 1 1 1 1`
- `1 1 1 1 1 1`

- `1 1 1 1 1 1`
- `-1 -1 -1 -1 -1 -1`
- `1 1 1 1 1 1`
- `-1 1 -1 -1 -1 -1`
- `1 1 1 1 1 1`
- `-1 -1 -1 -1 -1 -1`
- `-1 -1 -1 -1 -1 -1`
- `-1 -1 -1 -1 -1 -1`
- `1 1 1 1 1 1`
- `1 1 1 1 1 1`
- `-1 -1 -1 -1 -1 -1`

A few rebels

Example 2: still no HU

- **Three examples now:**
- `(1 -1 -1 1 1 1)`
- `(-1 1 -1 1 1 1)`
- `(-1 -1 1 1 1 1)`

- **Train a BM on them**

Example 2

Correlations in the examples:

- `(1 -1 -1 1 1 1)`
- `(-1 1 -1 1 1 1)`
- `(-1 -1 1 1 1 1)`

This is what we get:

- `0.3 -0.1 -0.1 -0.1 -0.1 -0.1`
- `-0.1 0.3 -0.1 -0.1 -0.1 -0.1`
- `-0.1 -0.1 0.3 -0.1 -0.1 -0.1`
- `-0.1 -0.1 -0.1 0.3 0.3 0.3`
- `-0.1 -0.1 -0.1 0.3 0.3 0.3`
- `-0.1 -0.1 -0.1 0.3 0.3 0.3`

x η

Example 2

Learning parameters, same as in example 1:

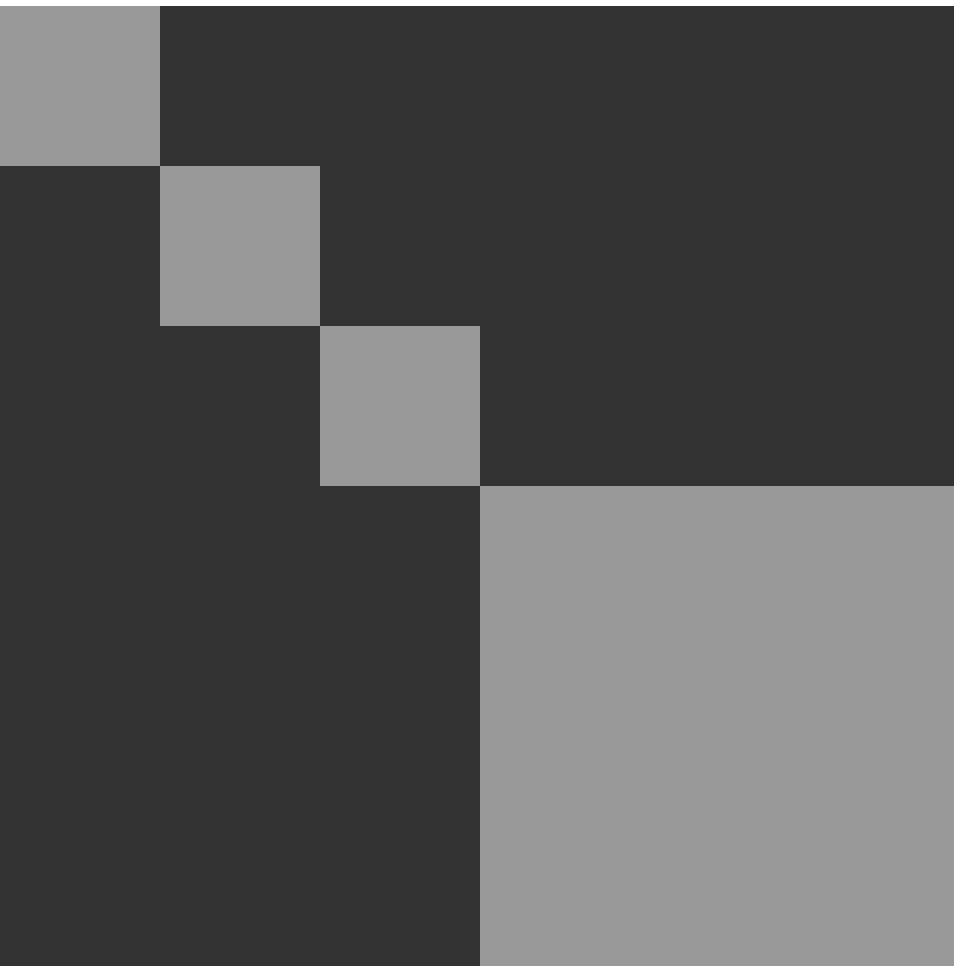
- **Learning rate η=0.1**
- **Temperature T=1**

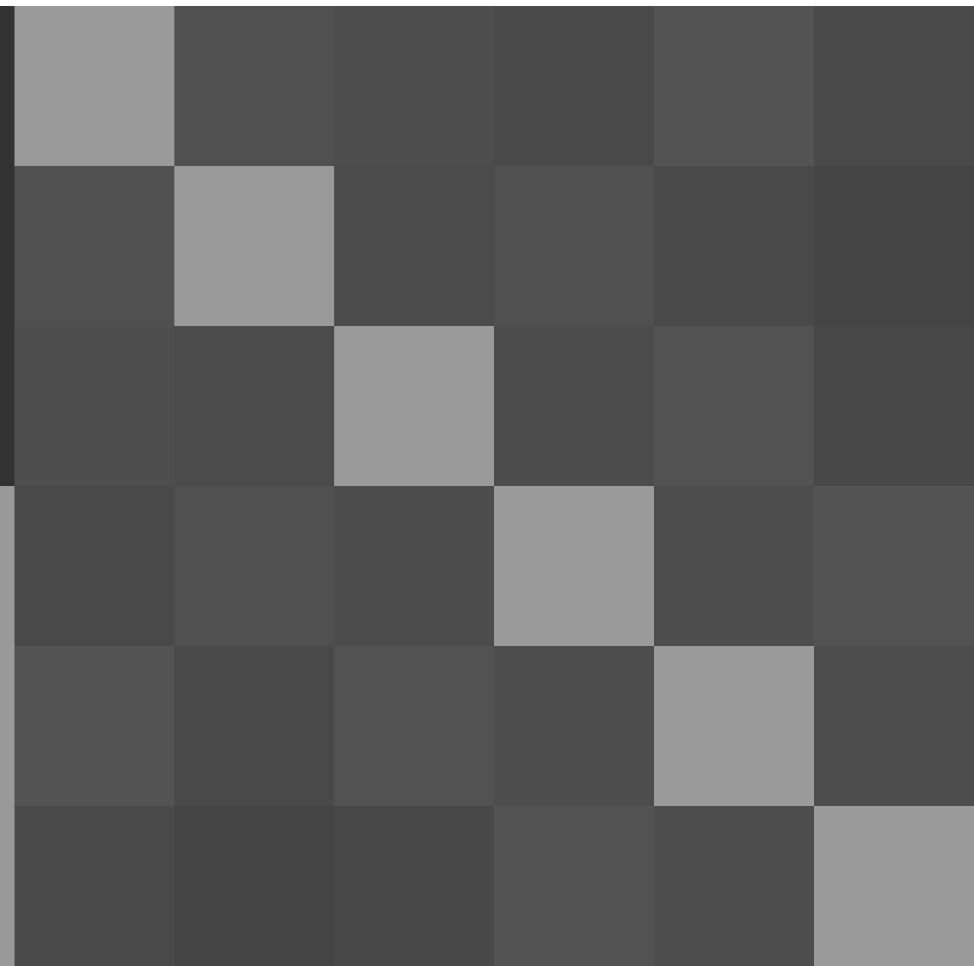
"let the BM machine run <u>many times</u> <u>until equilibrium</u>":

- **many times = 1000**
- **until equilibrium = 600 neuron flips**

Step 0

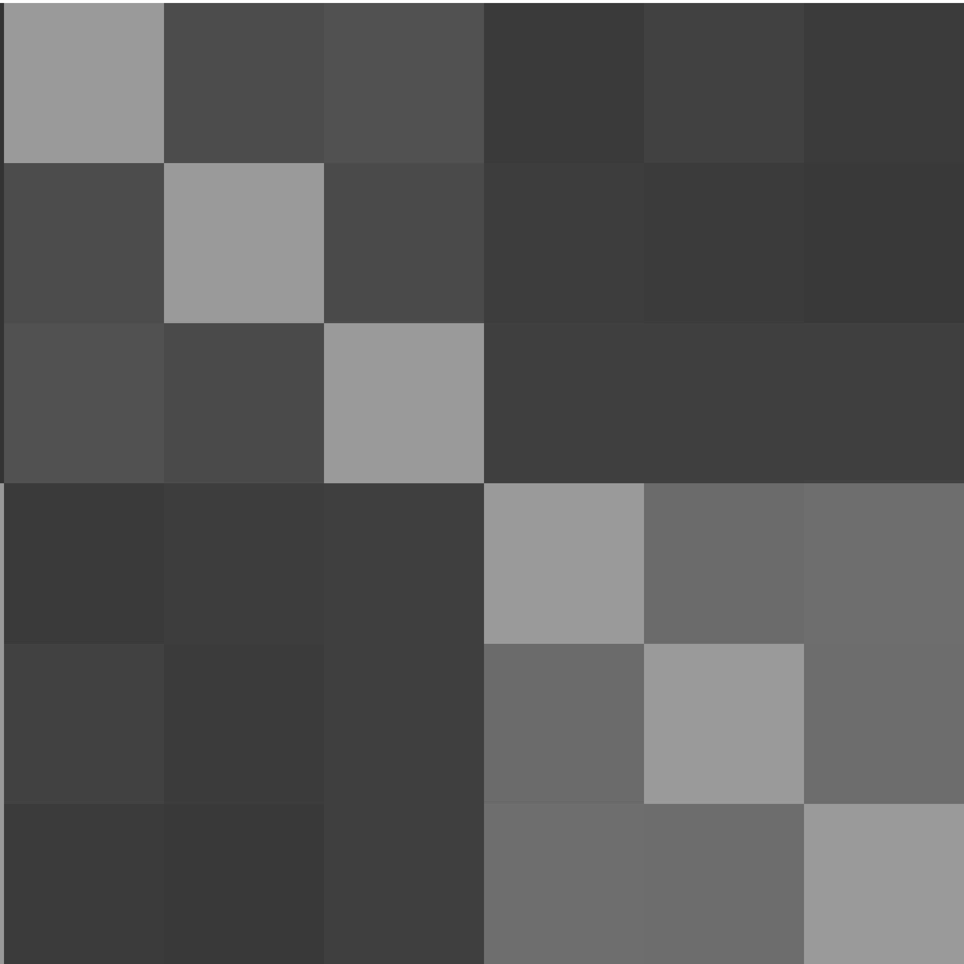
- **Reality**
- **Dream**





Step 1

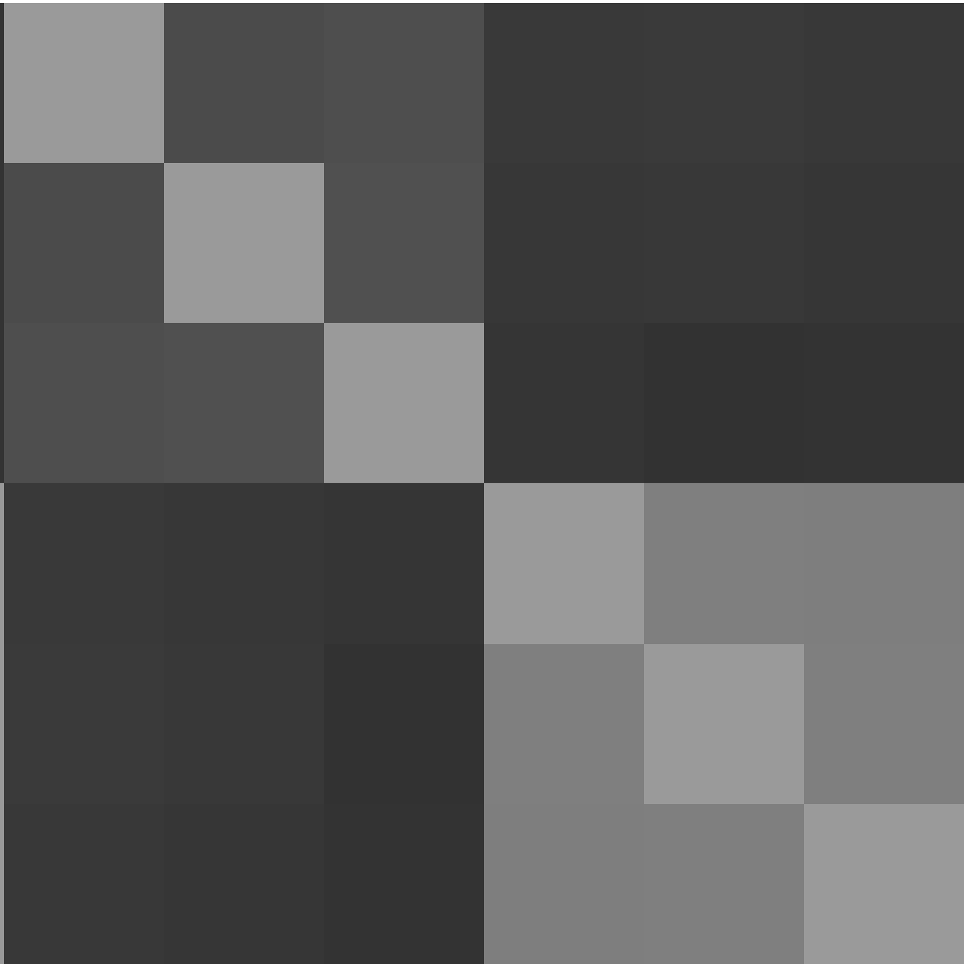
- **Reality**
- **Dream**



Step 2

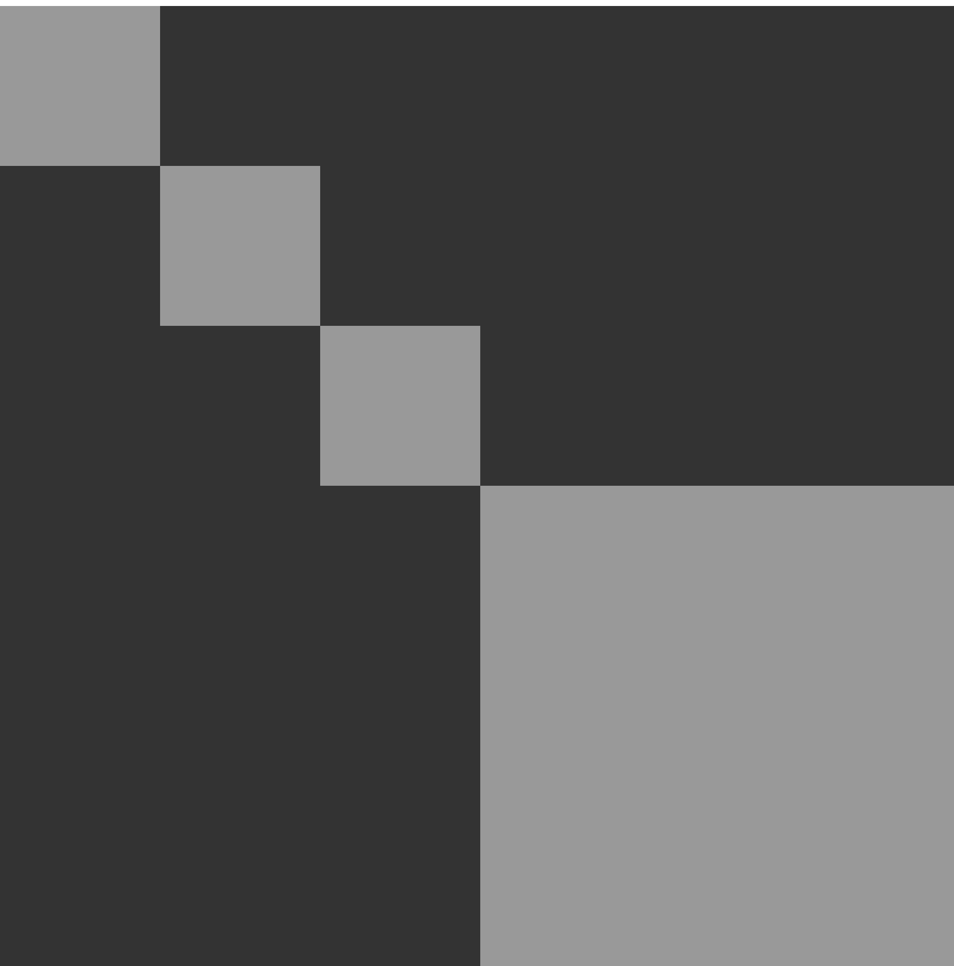
- **Reality**

- **Dream**

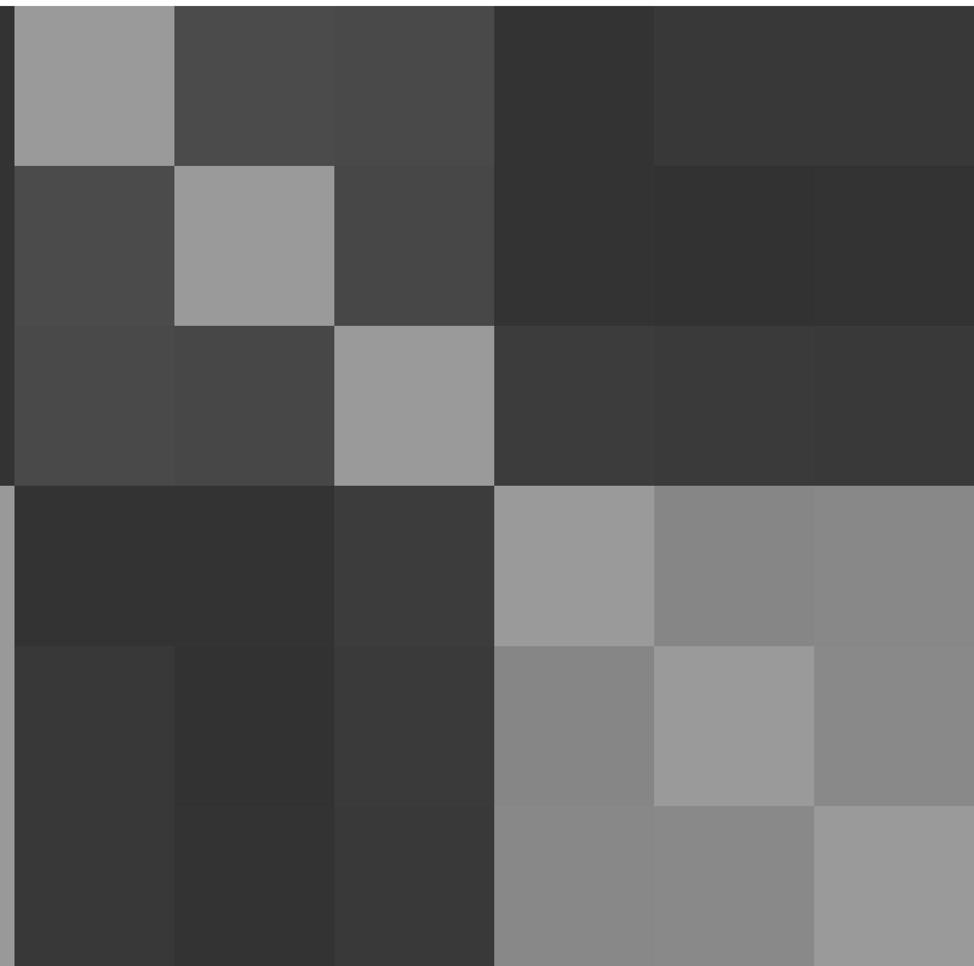


Step 3

- **Reality**

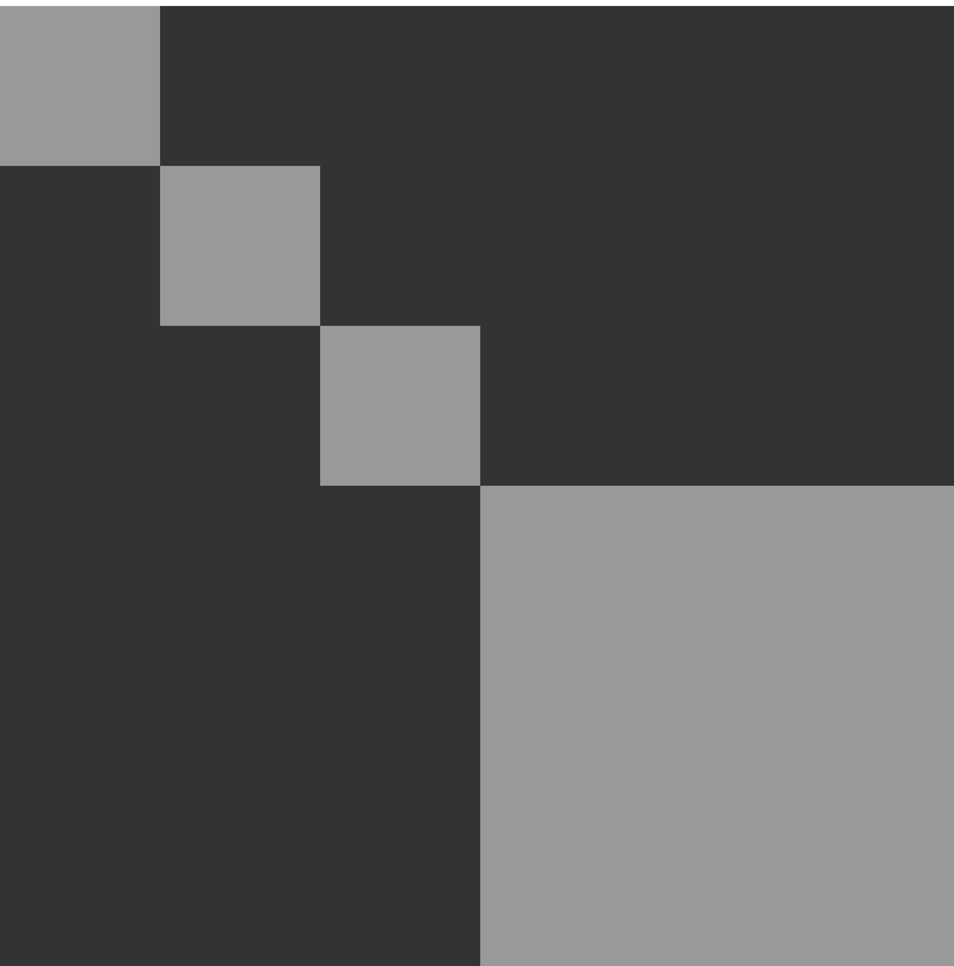


- **Dream**

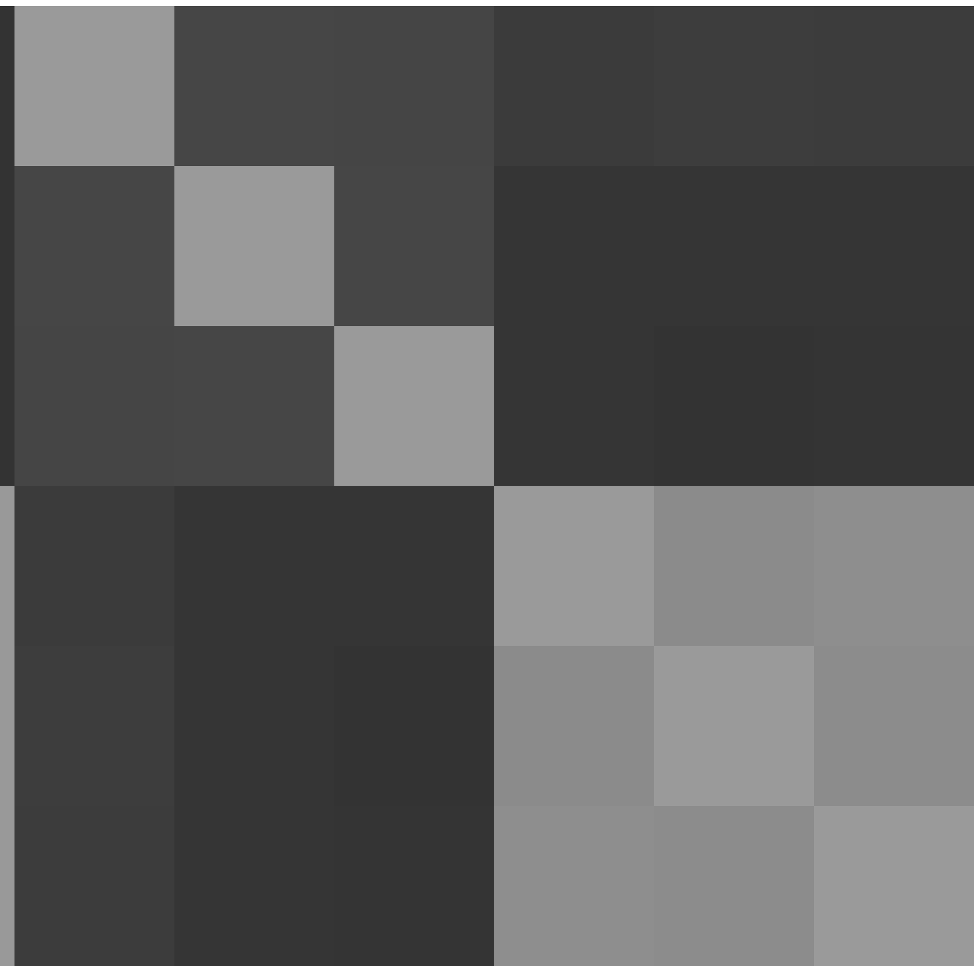


Step 4

- **Reality**

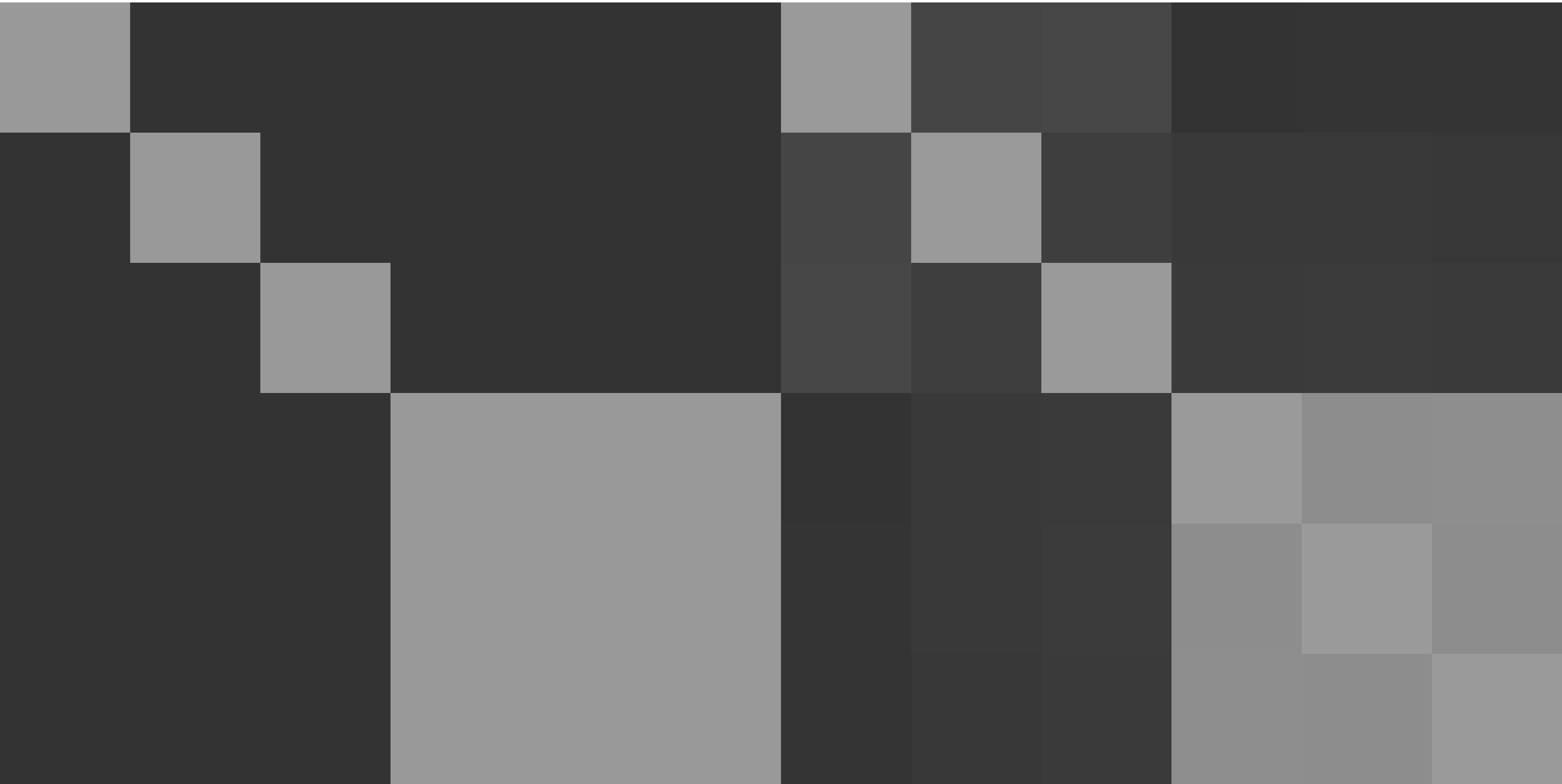


- **Dream**



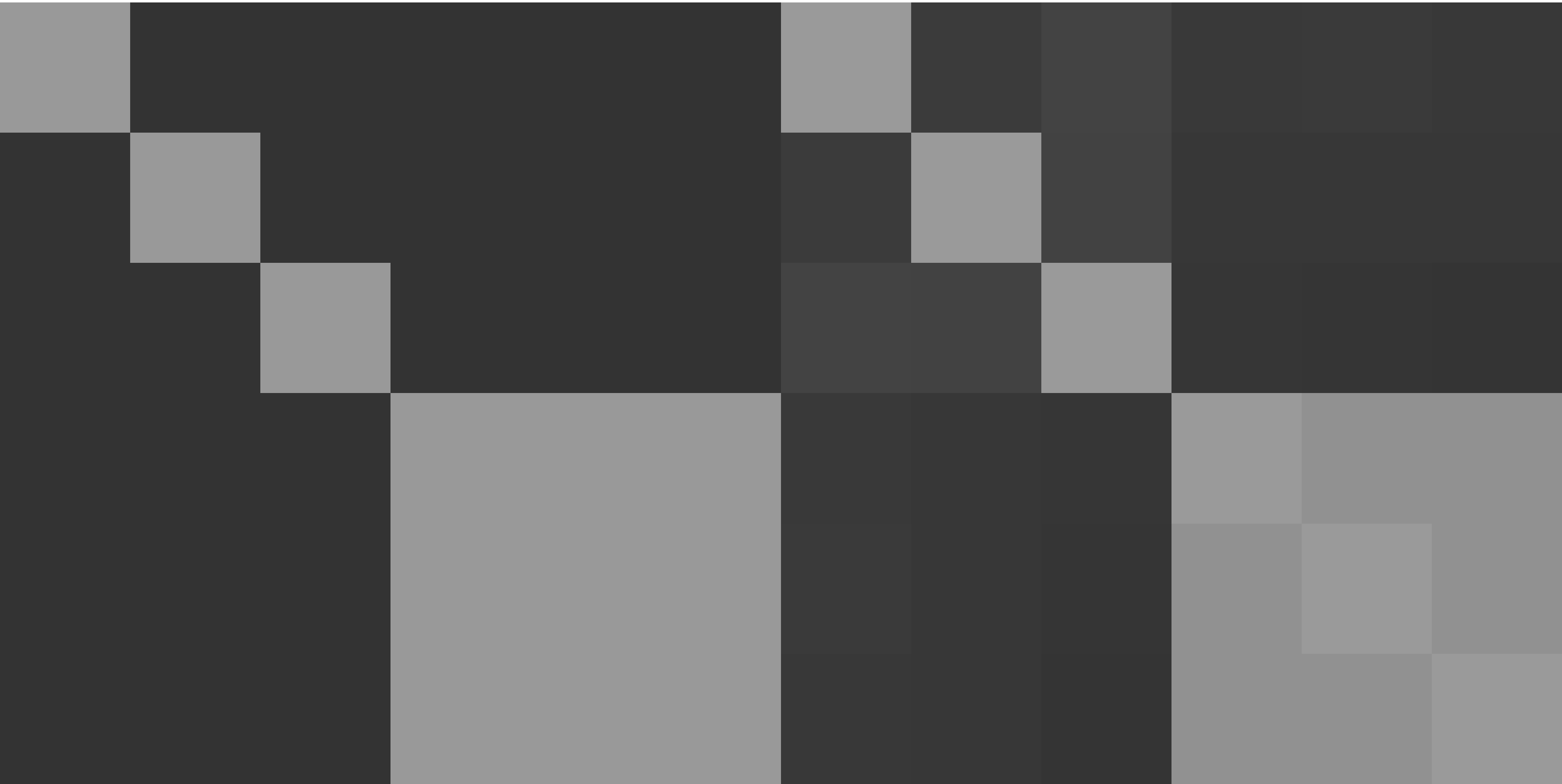
Step 5

- Reality
- Dream



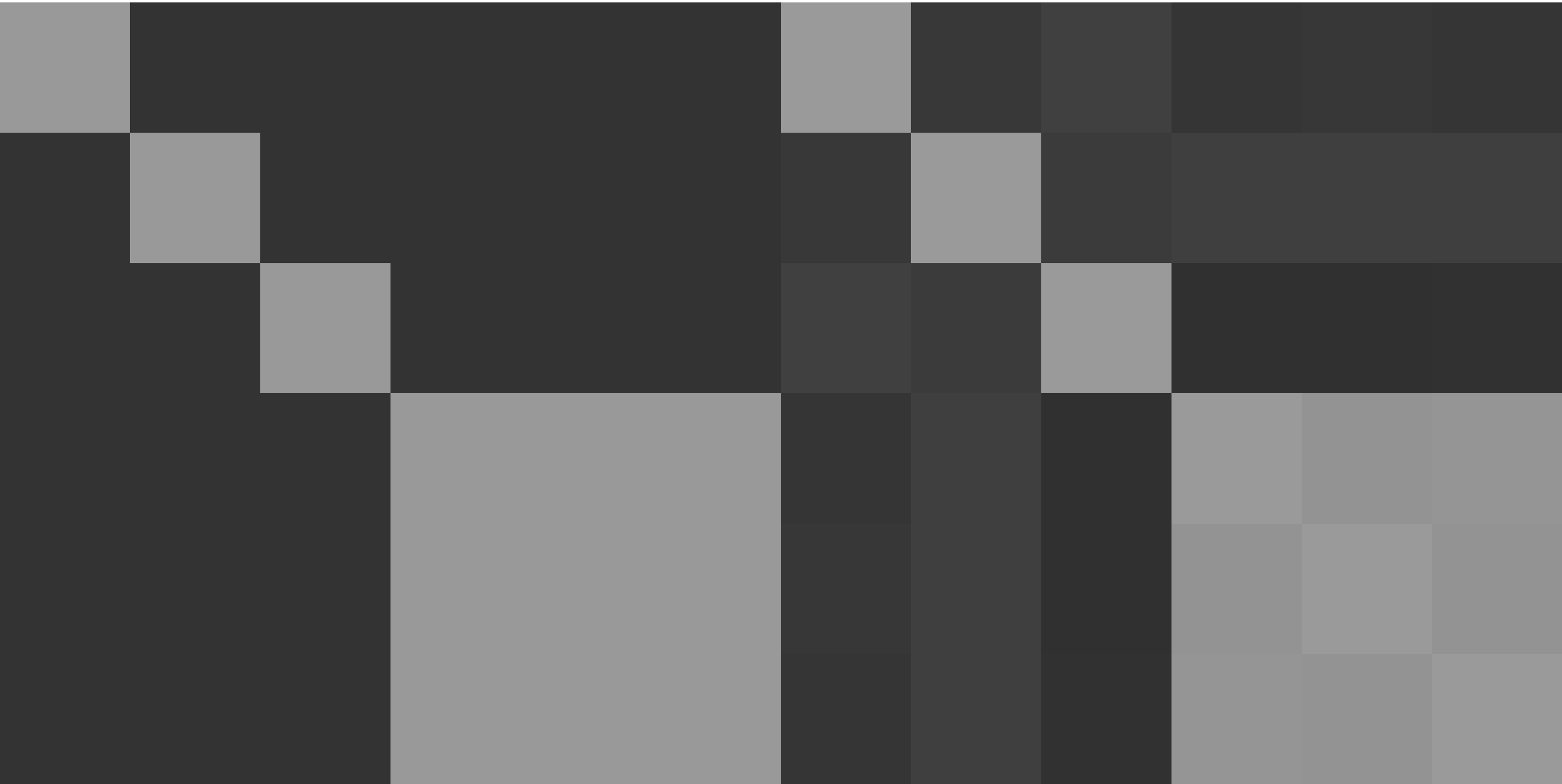
Step 6

- **Reality**
- **Dream**



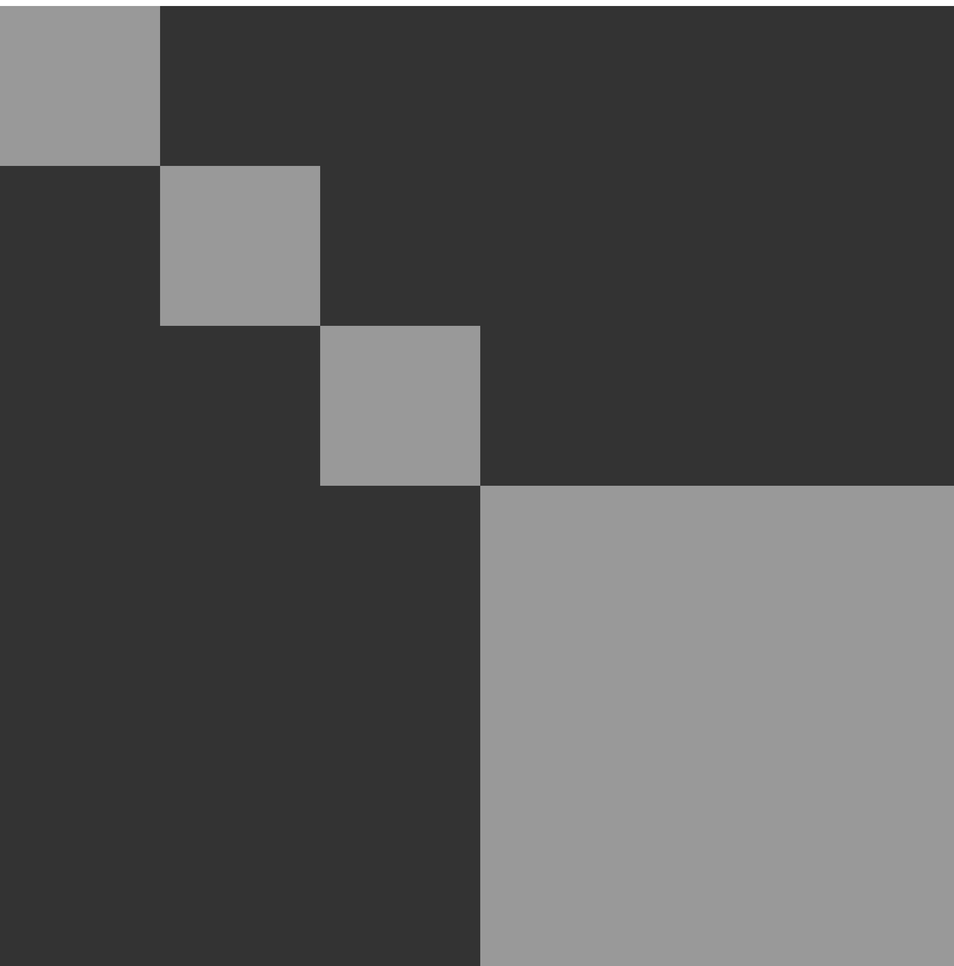
Step 10

- **Reality**
- **Dream**

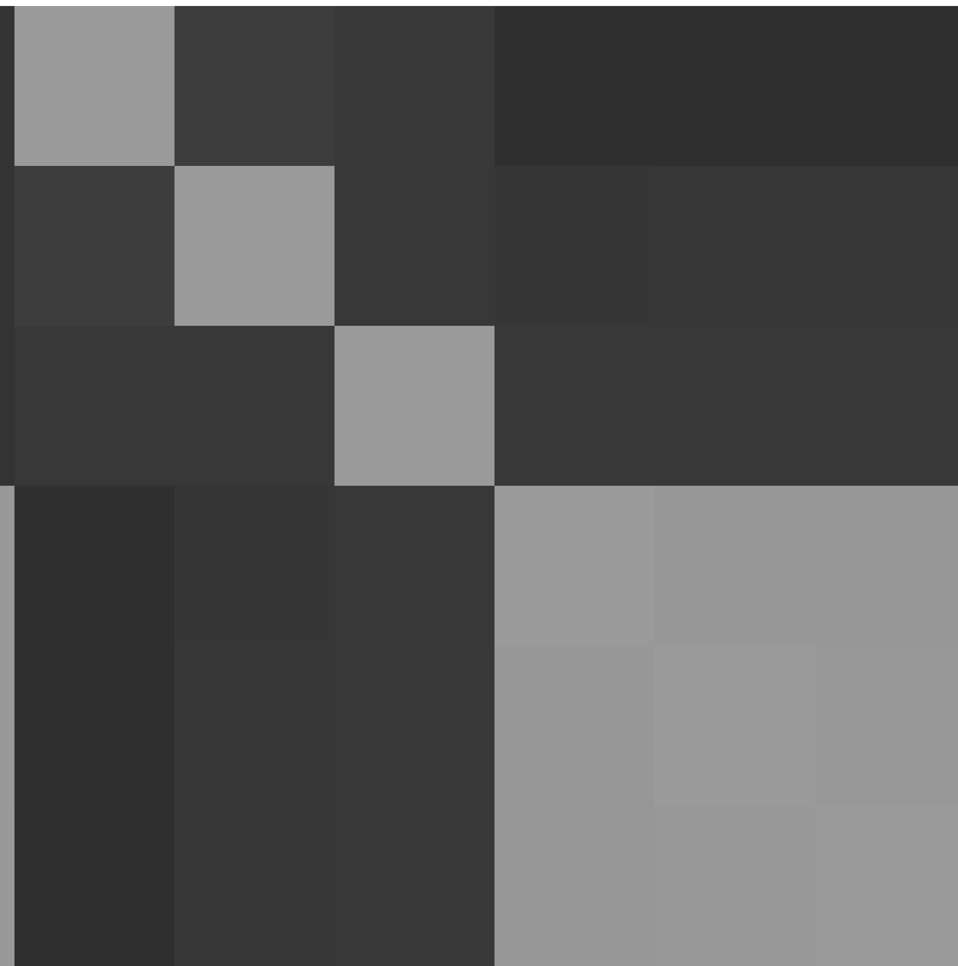


Step 20

- **Reality**

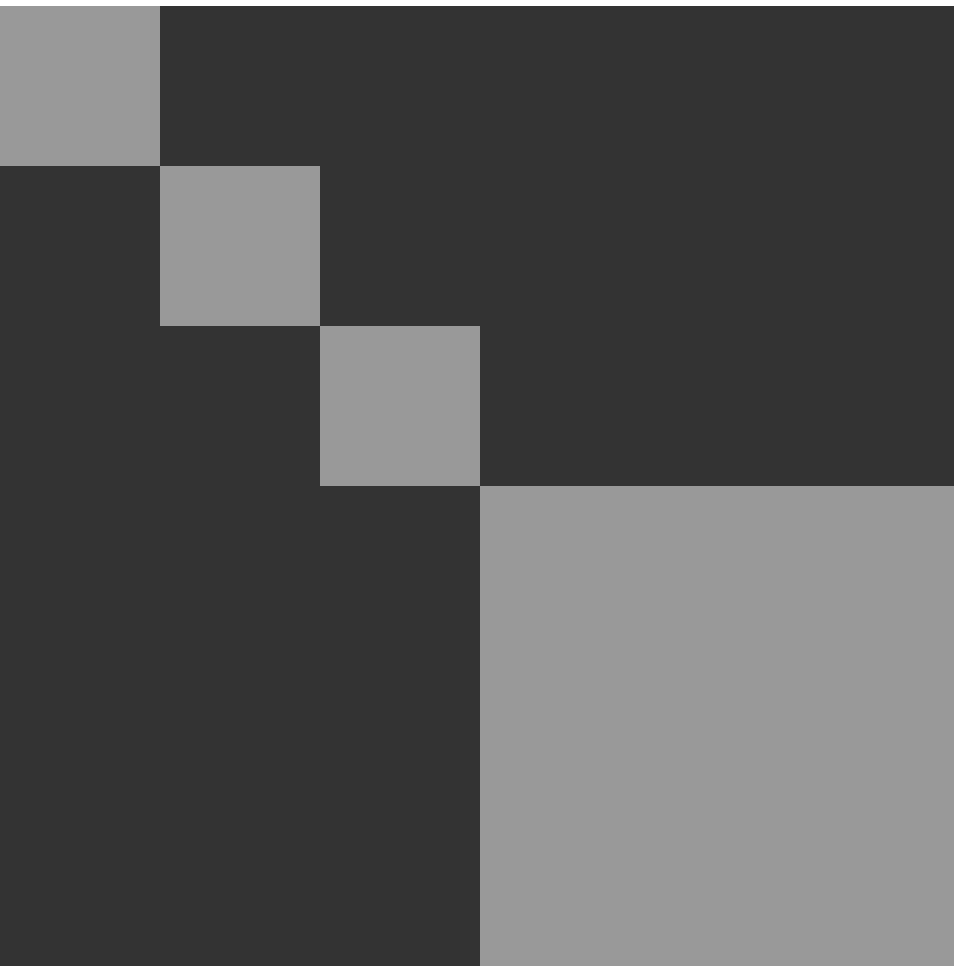


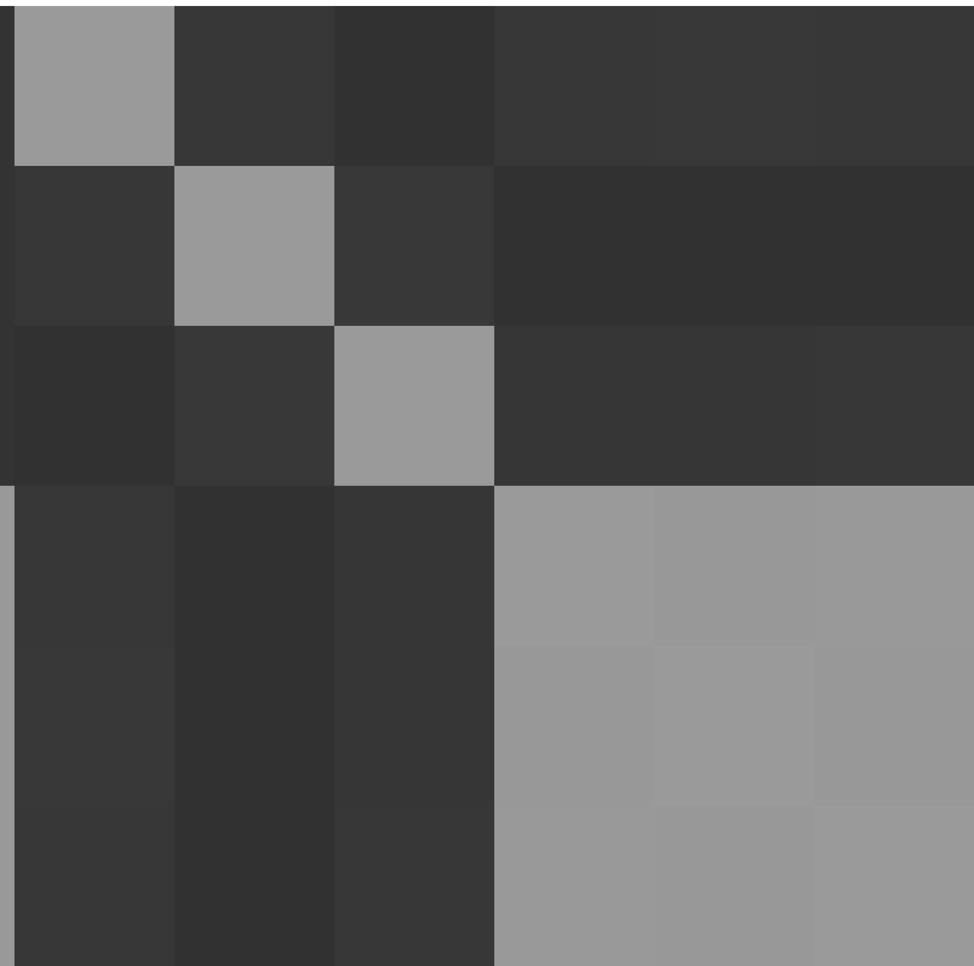
- **Dream**



Step 50

- **Reality**
- **Dream**





Example 2

Original patterns:

- (1 -1 -1 1 1 1)
- (-1 1 -1 1 1 1)
- (-1 -1 1 1 1 1)

- **We now let the BM converge from random points:**

- -1 1 1 -1 -1 -1
- -1 1 -1 1 1 1
- -1 1 1 -1 -1 -1
- 1 -1 1 -1 -1 -1
- -1 1 -1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 -1 1 1 1 1
- -1 -1 1 1 1 1
- -1 -1 1 -1 -1 -1

- -1 1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1
- 1 -1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1
- -1 1 1 -1 -1 -1
- 1 1 -1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 -1 1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1

Example 2

Original patterns:

- (1 −1 −1 1 1 1)
- (−1 1 −1 1 1 1)
- (−1 −1 1 1 1 1)

- **We now let the BM converge from random points:**

- −1 1 1 −1 −1 −1
- −1 1 −1 1 1 1
- −1 1 1 −1 −1 −1
- 1 −1 1 −1 −1 −1
- −1 1 −1 1 1 1
- 1 −1 1 −1 −1 −1
- −1 −1 1 1 1 1
- −1 −1 1 1 1 1
- −1 −1 1 −1 −1 −1

- −1 1 1 −1 −1 −1
- 1 1 −1 −1 −1 −1
- 1 −1 1 −1 −1 −1
- 1 1 −1 −1 −1 −1
- −1 1 1 −1 −1 −1
- 1 1 −1 1 1 1
- 1 −1 1 −1 −1 −1
- −1 −1 1 1 1 1
- 1 −1 1 −1 −1 −1
- −1 1 1 −1 −1 −1
- 1 1 −1 −1 −1 −1

Just a few right ones

Example 2

Original patterns:

- (1 -1 -1 1 1 1)
- (-1 1 -1 1 1 1)
- (-1 -1 1 1 1 1)

- We now let the BM converge from random points:

- -1 1 1 -1 -1 -1
- -1 1 -1 1 1 1
- -1 1 1 -1 -1 -1
- 1 -1 1 -1 -1 -1
- -1 1 -1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 -1 1 1 1 1
- -1 -1 1 1 1 1
- -1 -1 1 -1 -1 -1

- -1 1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1
- 1 -1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1
- -1 1 1 -1 -1 -1
- 1 1 -1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 -1 1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1

A good few opposites

Example 2

Original patterns:

- (1 -1 -1 1 1 1)
- (-1 1 -1 1 1 1)
- (-1 -1 1 1 1 1)

- We now let the BM converge from random points:

- -1 1 1 -1 -1 -1
- -1 1 -1 1 1 1
- -1 1 1 -1 -1 -1
- 1 -1 1 -1 -1 -1
- -1 1 -1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 -1 1 1 1 1
- -1 -1 1 1 1 1
- -1 -1 1 -1 -1 -1

- -1 1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1
- 1 -1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1
- -1 1 1 -1 -1 -1
- 1 1 -1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 -1 1 1 1 1
- 1 -1 1 -1 -1 -1
- -1 1 1 -1 -1 -1
- 1 1 -1 -1 -1 -1

Rebels

Example 2

- **Problem here: second order relations are just not enough to model even this simple problem.**
- **Need hidden units?**

Hidden Units

- Hidden Units take time to be trained, and it is problematic to run a machine with them in any realistic case using the standard algorithm on a CPU.
- Later in the course I'll talk about ways to overcome the issue.

Learning

- We will need concepts of learning theory.
- What is learning exactly, after all?

Types of learning task

- **Supervised learning:**
  - learning to predict output when examples of input and corresponding output are available.
- **Reinforcement learning:**
  - no real supervision, only occasional payoff: e.g. I don’t know if a chess move is correct but I know if I win.
- **Unsupervised learning:**
  - I want to create an internal representation of the data, e.g. in form of clusters, extract relevant features for further tasks, etc.

Supervised learning

- A set of examples: <x,f(x)>
  - x is some object (instance) $\in \mathcal{X}$ (instance space)
  - f(·) is an *unknown* function
- Learning algorithm will guess h(·) ≈ f(·)
- Inductive learning hypothesis
  - Any h(·) that approximates f(·) well on training examples will also approximate f(·) well on new (unseen) instances x. This isn't necessarily true, of course..

Types of supervised learning task

- **f(x) is a real value/an array of real values: *regression***
- **f(x) is a discrete value; $f(x) \in \{y_1 .. y_K\}$: *classification***
  - **if K=2, *binary classification***

Example

- **Medical diagnosis (classification)**
  - **x: set of properties for a patient (symptoms, lab tests, previous diseases)**
  - **f(x): disease**
  - **<x,f(x)>: Database of past medical records**
  - **Type of x: a fixed-length array (attribute/value representation), maybe with missing attributes**
  - **Type of f(x): a discrete value or a fixed-width array**

Example

- **Making a computer read (classification?)**
  - **x: letters of sentences in some language**
  - **f(x): corresponding sound features (in context)**
  - **<x,f(x)>: Database of texts/corresponding sound features extracted from humans reading.**
  - **Type of x: a fixed-length array**
  - **Type of f(x): a fixed-length array**

Supervised learning

- Training examples: <x,f(x)>
- Hypothesis space: set H of functions that (hopefully) contains the target function f(·)
- Result of learning: a function h(·)∈H approximating f(·)
- Examples are used to guide the algorithm to choose a good h(·)∈H

Supervised learning

- h(·) is called a *consistent hypothesis* if it agrees with f(·) on all training examples
  - After observing the data, only some hypotheses in H are consistent. They form the so called version space :

    $$\{h(\cdot)\in H : h(x)=f(x)\ \forall \text{ example } x\}$$

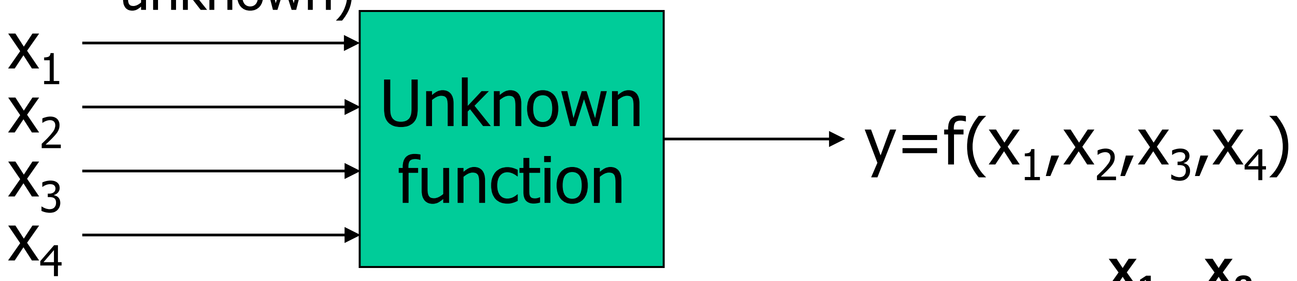
- A *consistent learner* always outputs a consistent hypothesis (i.e., it is 100% accurate on the training set)
- The empirical error is the fraction of training examples such that $h(x)\neq f(x)$ (0% in a consistent learner)

Example (boolean functions)

Four boolean variables as input features:

$$x=(x_1,x_2,x_3,x_4)\in\{0,1\}^4$$

$f(x_1,x_2,x_3,x_4) = \neg x_2 \wedge x_4$ (but it is unknown)



Training examples

x_1	x_2	x_3	x_4	y
0	0	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Hypothesis space 1

- No knowledge: H is the set of all boolean functions on 4 variables. $|H|=2^{16}\approx 64\times 10^3$

x_1	x_2	x_3	x_4	Y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	?
0	0	1	1	?
0	1	0	0	?
0	1	0	1	?
0	1	1	0	?
0	1	1	1	?
1	0	0	0	?
1	0	0	1	?
1	0	1	0	?
1	0	1	1	?
1	1	0	0	?
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Hypothesis space 1

- **After observing 5 examples we are left with $2^{16-5}=2^{11}=2048$ consistent hypotheses**
- **A randomly-picked consistent h(·) is unlikely to approximate f(·) well…**
- **What is wrong?**

x_1	x_2	x_3	x_4	Y
0	**0**	**0**	**0**	**?**
0	**0**	**0**	**1**	**?**
0	**0**	**1**	**0**	**?**
0	0	1	1	1
0	**1**	**0**	**0**	**?**
0	**1**	**0**	**1**	**?**
0	**1**	**1**	**0**	**?**
0	**1**	**1**	**1**	**?**
1	**0**	**0**	**0**	**?**
1	0	0	1	1
1	**0**	**1**	**0**	**?**
1	0	1	1	1
1	1	0	0	0
1	**1**	**0**	**1**	**?**
1	1	1	0	0
1	**1**	**1**	**1**	**?**

Hypothesis space 2

- A literal is a variable x_i or its negation $\neg x_i$
- A term is the conjunction of literals
- H= {terms over x_1,x_2,x_3,x_4}
- $|H|=3^4=81$ (each variable is affirmed, is negated, isn't present)
- Learning algorithm:
  - Initial h(·) = conjunction of all possible literals
  - Remove literals associated with inconsistent *positive* examples

Learning conjunctions

1. $h = \neg x_1 x_1 \neg x_2 x_2 \neg x_3 x_3 \neg x_4 x_4$

2. Observe 1st training example, remove literals x_1, x_2, $\neg x_3$, and $\neg x_4$
   $h = \neg x_1 \neg x_2\, x_3\, x_4$
3. Observe 2nd training example, remove literals $\neg x_1$ and x_3
   $h = \neg x_2\, x_4$
4. Observe 3rd training example: nothing to do ($h = \neg x_2\, x_4$)

5. No more positive training examples
   Output $h = \neg x_2\, x_4$

x_1	x_2	x_3	x_4	y
0	0	1	1	1
1	0	0	1	1
1	0	1	1	1