Deep Learning Seminar

Chapter 15 Representation Learning

Heechul Lim

Department of Information and Communication Engineering

DGIST

2017.01.16

Contents

- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Contents

- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Unsupervised Pre-training(UP)

- **Details**
  - Learning without labelling
  - Learning only once before joint training
  - Regularizer or parameter init that reduces test error
- **Basic idea of UP**
  - Features from UP are also useful for SL by using representation
  - To draw a car, you need to the number of wheels

Property of UP

- Because it is **not mathematically well-defined**, the way to benefit from UL is not well defined
- The results of whether **pretraining is good or bad**
- On average, pretraining is slightly **detrimental**
- But it can be a **significant help** in many tasks
- You need to understand **when and why it works**, So you can use UL properly

Background of UP

- This way of learning has been recognized for years as a way of learning about in-depth networks since 2006
- But it is rarely used later.
- The reason for this is that as of 2010
- ReLU, dropout, maxout, data augmentation and deterministic batch normalization methods are introduced
- It is possible to get better performance by SL without using UP

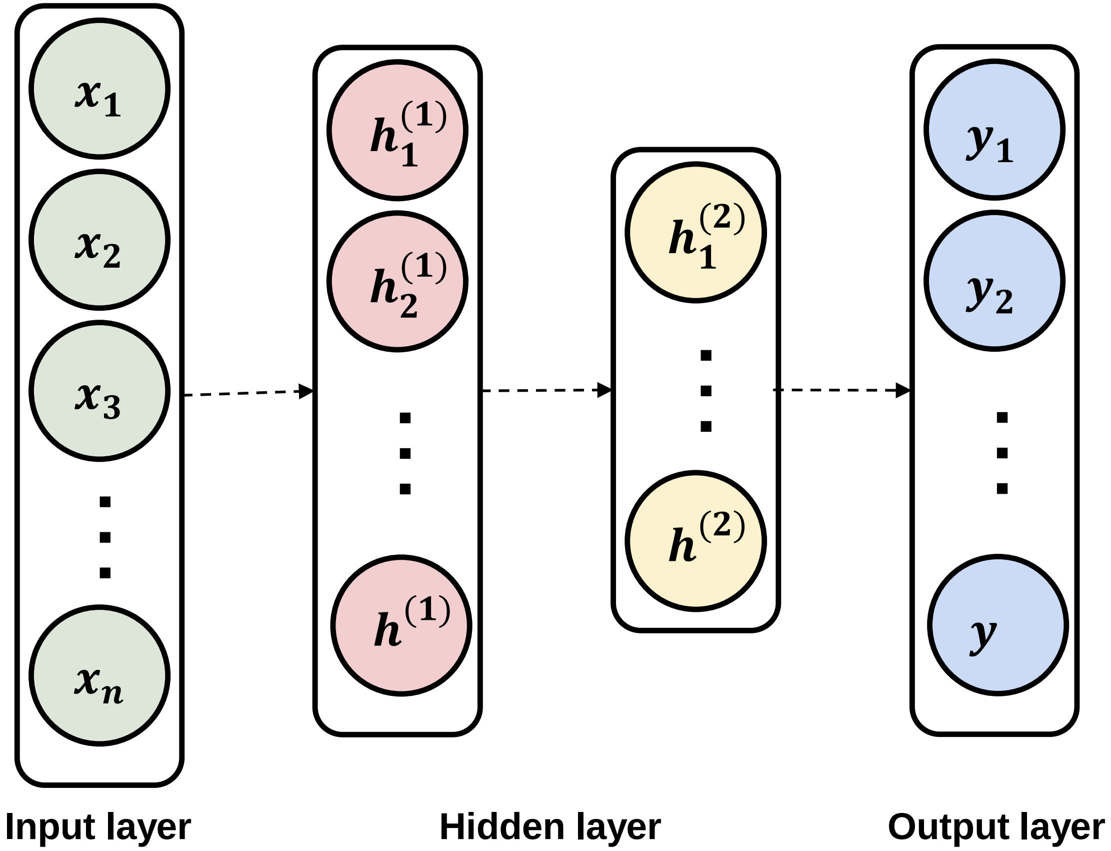
Pros of unsupervised pre-training(UP)

- Shared representations
  - To handle multiple domains
  - To transfer learned knowledge to many other tasks
- Data sets used
  - In order to learn neural network, many data are needed
  - However, there are many data which do not have label but little label data
  - In this case, it is possible to pre-train each layer

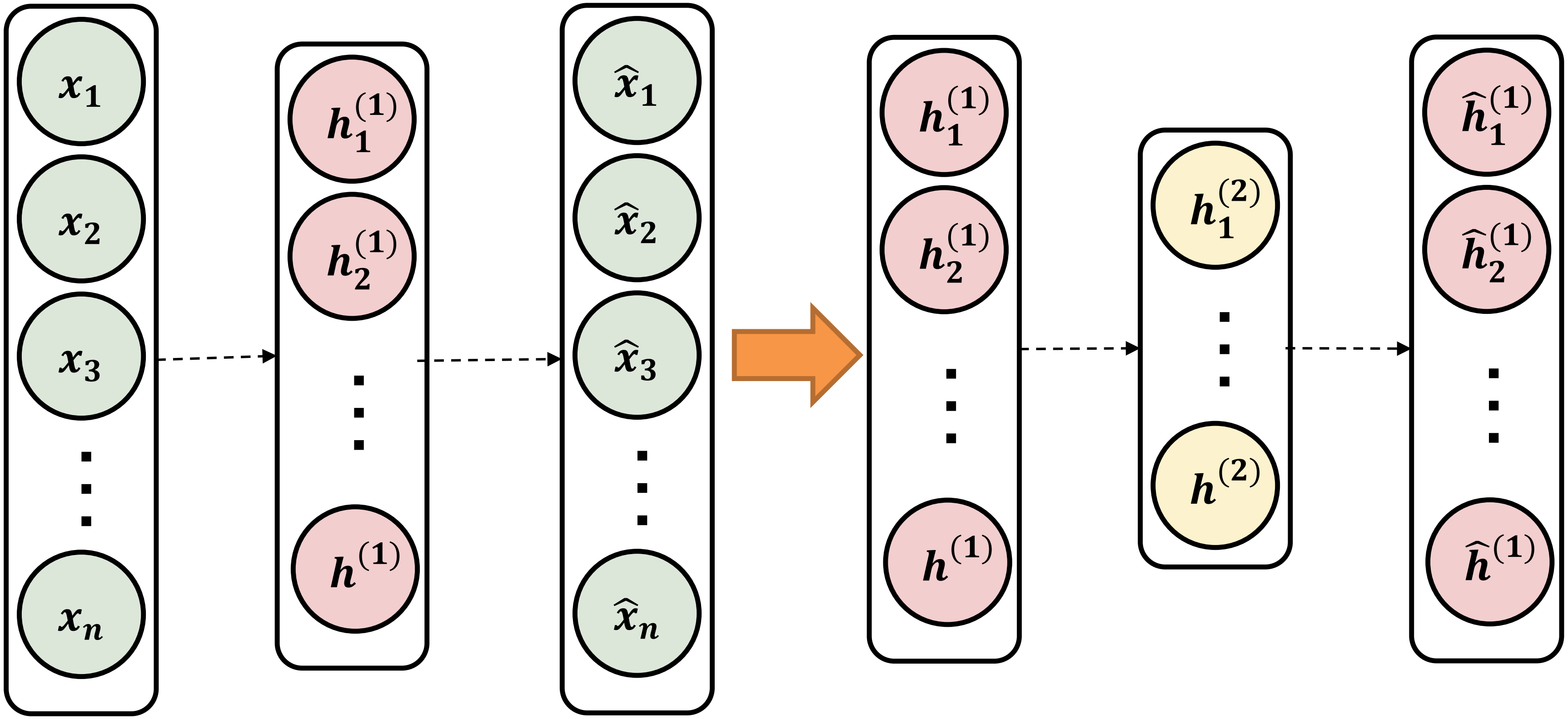
Greedy layer-wise UP in the book

- UL per a single layer basis
- Greedy
  - It progresses by layer
- Pretraining
  - It runs only once before joint training
- Regularizer or parameter init that reduces test error

Pre-training for deep neural network example


x_1
x_2
x_3
x_n
$h_1^{(1)}$
$h_2^{(1)}$
$h^{(1)}$
$h_1^{(2)}$
$h^{(2)}$
y_1
y_2
y
Input layer
Hidden layer
Output layer

Greedy layer-wise UP

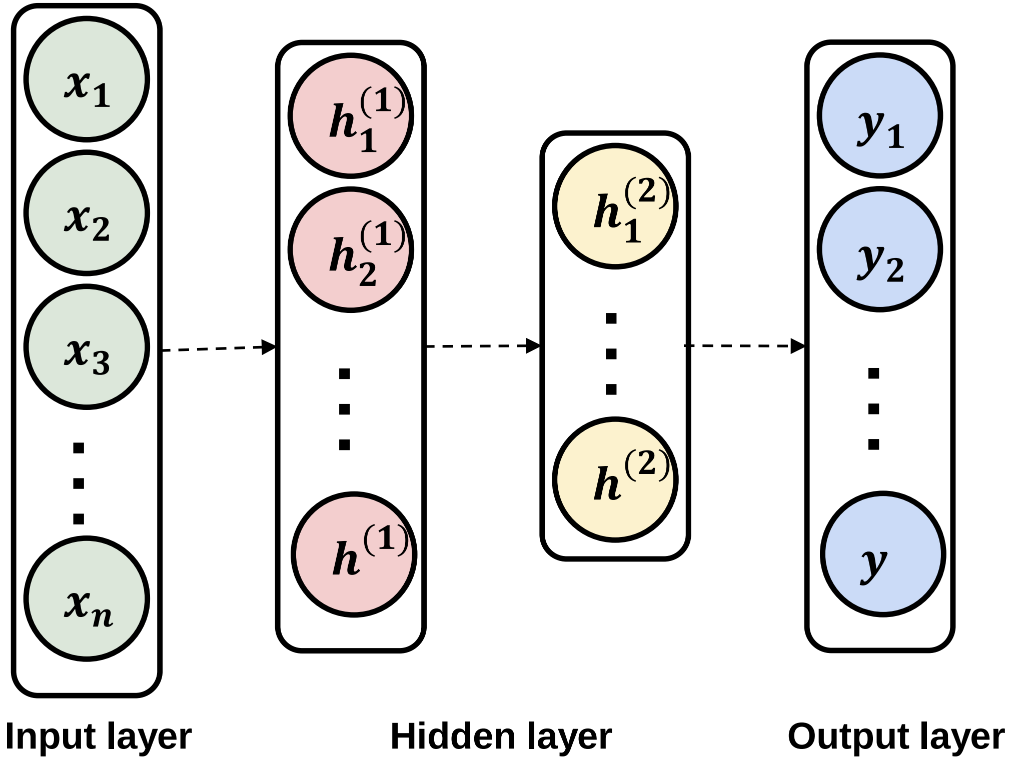

x_1
x_2
x_3
x_n
$h_1^{(1)}$
$h_2^{(1)}$
$h^{(1)}$
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_n$
$h_1^{(1)}$
$h_2^{(1)}$
$h^{(1)}$
$h_1^{(2)}$
$h^{(2)}$
$\hat{h}_1^{(1)}$
$\hat{h}_2^{(1)}$
$\hat{h}^{(1)}$

1st autoencoders

2nd autoencoders

Combining layers after pre-training

- **After combining layers**
  - The parameters for each layer have been determined
  - It is then possible to perform supervised fine-tuning


x_1
x_2
x_3
x_n
$h_1^{(1)}$
$h_2^{(1)}$
$h^{(1)}$
$h_1^{(2)}$
$h^{(2)}$
y_1
y_2
y

Input layer
Hidden layer
Output layer

Contents

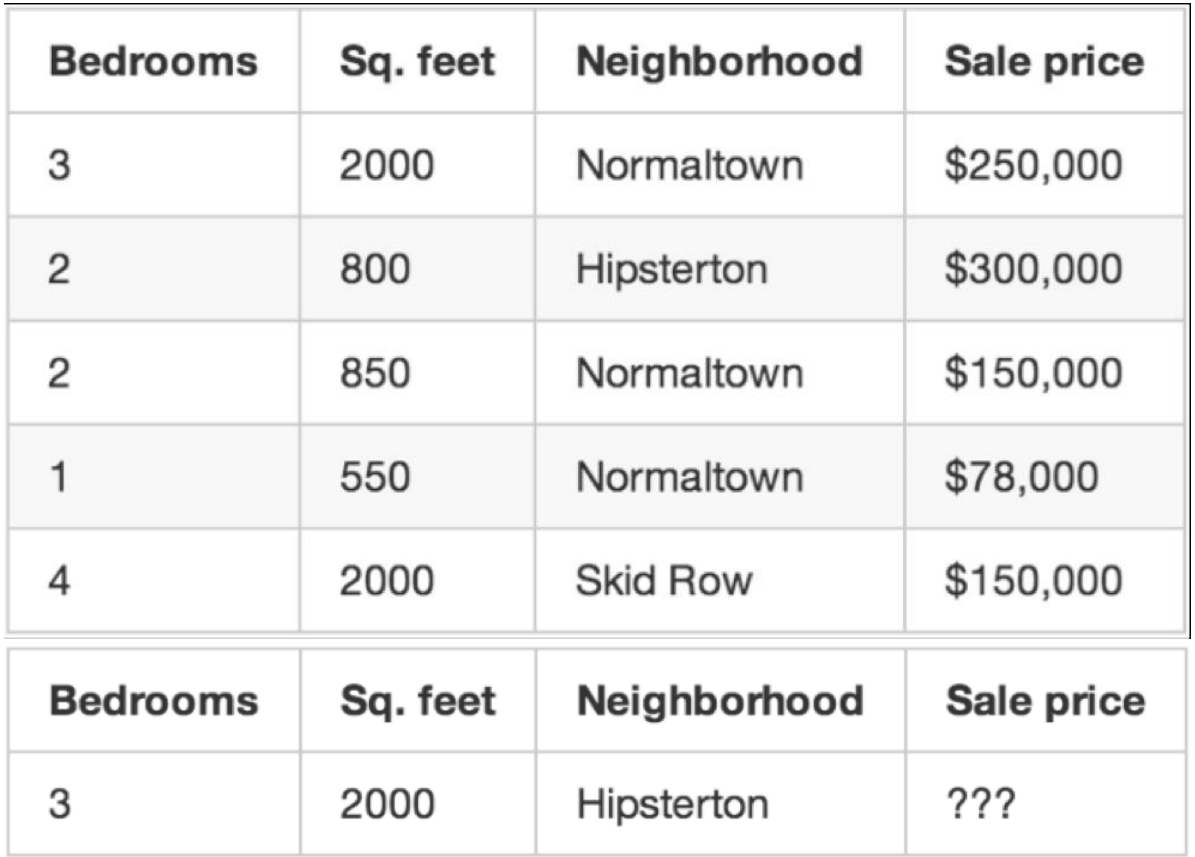
- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Introduction of SL and UL

- **Supervised Learning**
  - **Classification**
    A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”
  - **Regression**
    A regression problem is when the output variable is a real value, such as “dollars” or “weight”
- **Unsupervised Learning**
  - **Clustering**
    It is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior

Introduction of SL and UL

- **Supervised Learning**
  - Let’s say you are a real estate agent



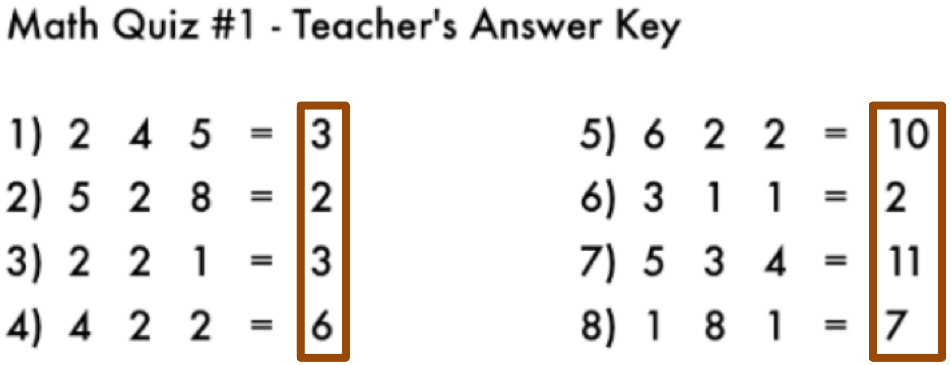
Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	$250,000
2	800	Hipsterton	$300,000
2	850	Normaltown	$150,000
1	550	Normaltown	$78,000
4	2000	Skid Row	$150,000

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Hipsterton	???

https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471

Supervised Learning

- **Let’s solve the problem with solutions(labels)**



Math Quiz #1 - Teacher's Answer Key

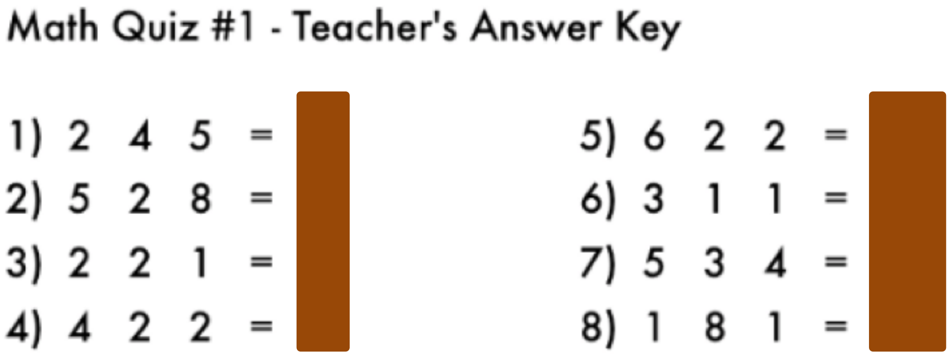
1) 2 4 5 = 3
2) 5 2 8 = 2
3) 2 2 1 = 3
4) 4 2 2 = 6

5) 6 2 2 = 10
6) 3 1 1 = 2
7) 5 3 4 = 11
8) 1 8 1 = 7

https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471

Unsupervised Learning

- **I don’t know what these numbers mean but maybe you(computer) can figure out**
- **Goal**
  - **modeling the underlying structure or distribution in the data in order to learn more about the data**



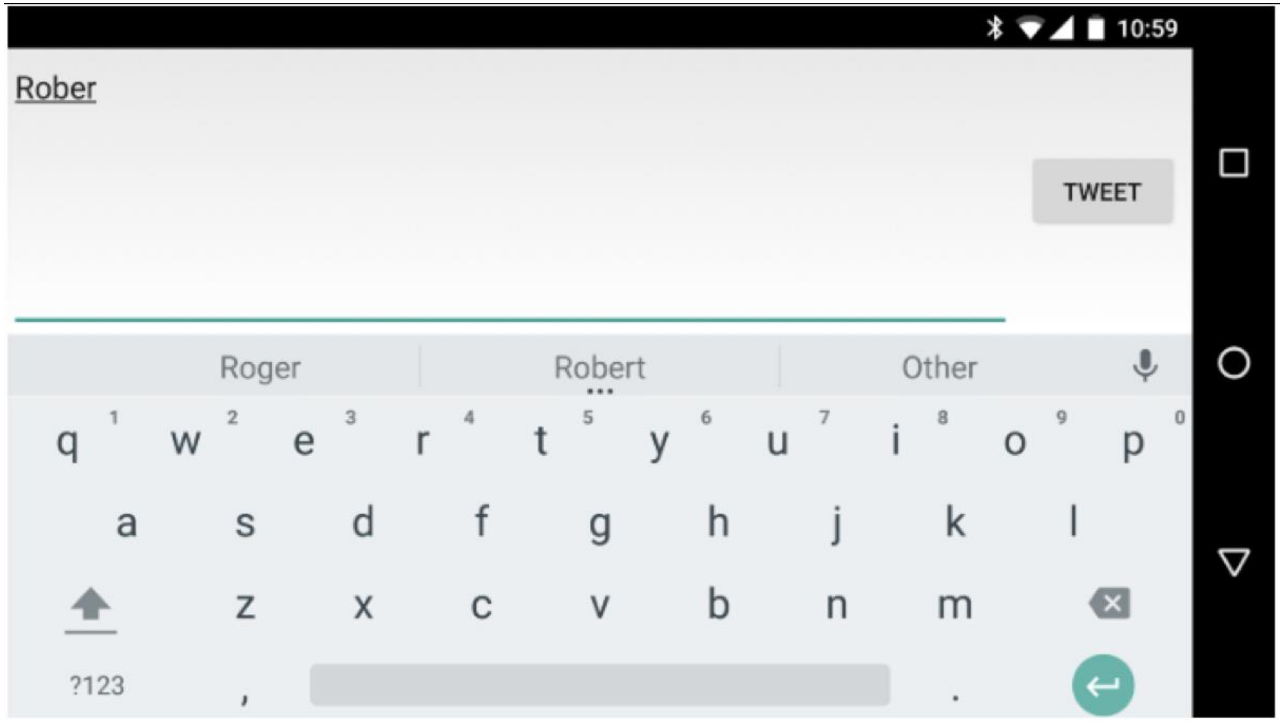
Math Quiz #1 - Teacher's Answer Key

1) 2 4 5 =
2) 5 2 8 =
3) 2 2 1 =
4) 4 2 2 =
5) 6 2 2 =
6) 3 1 1 =
7) 5 3 4 =
8) 1 8 1 =

https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471

Predicting the next letter in Phone keyboard

- One cool use might be auto-predict for a phone keyboard



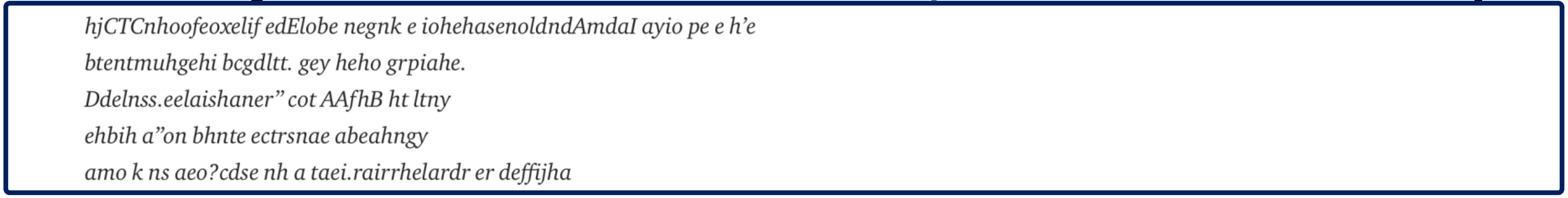
10:59
TWEET
Roger
Robert
Other
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123

Rober

https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

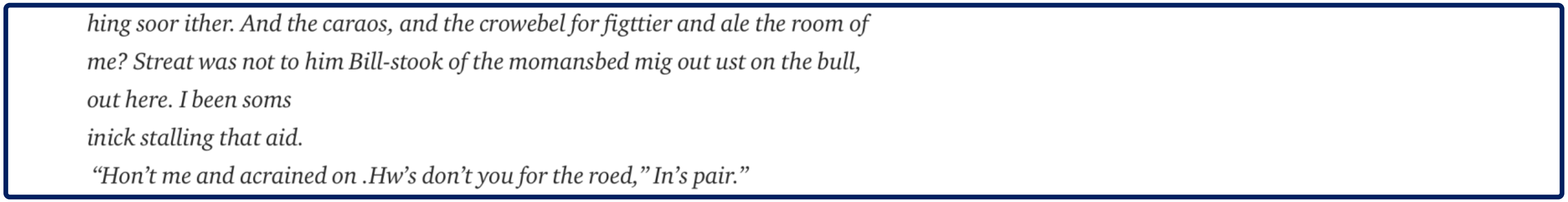
Predicting the next letter in a story

- **As we just start to train the RNN, after a 100 train loops**



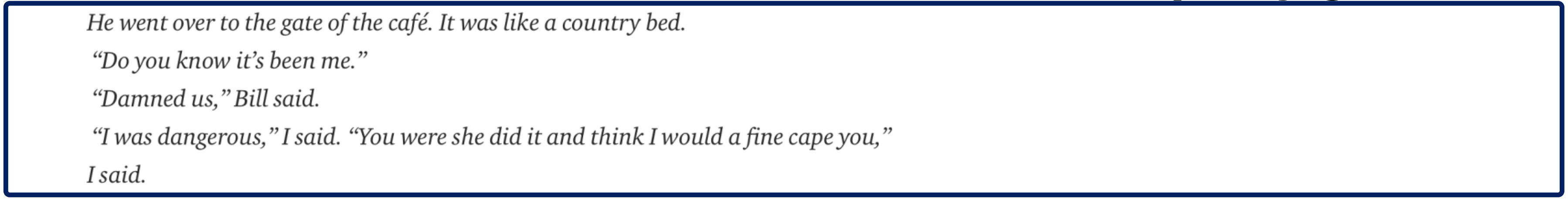
hjCTCnhoofeoxelif edElobe negnk e iohehasenoldndAmdaI ayio pe e h'e
btentmuhgehi bcgdltt. gey heho grpiahe.
Ddelnss.eelaishaner" cot AAfhB ht ltny
ehbih a"on bhnte ectrsnae abeahngy
amo k ns aeo?cdse nh a taei.rairrhelardr er deffijha

- **After about 1000 iterations**



hing soor ither. And the caraos, and the crowebel for figttier and ale the room of
me? Streat was not to him Bill-stook of the momansbed mig out ust on the bull,
out here. I been soms
inick stalling that aid.
"Hon't me and acrained on .Hw's don't you for the roed," In's pair."

- **After several thousand more iterations, it looks pretty good**

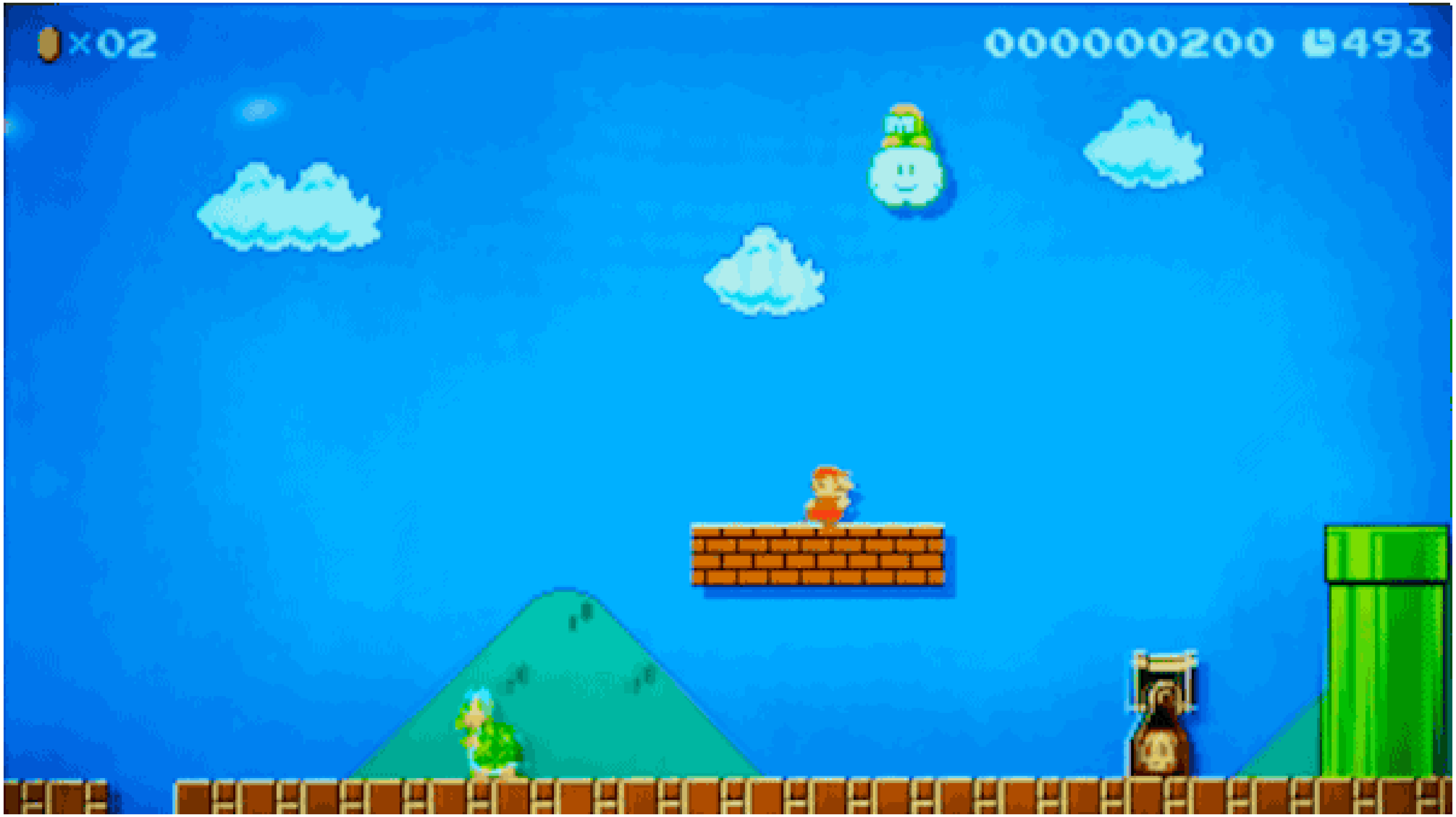


He went over to the gate of the café. It was like a country bed.
"Do you know it's been me."
"Damned us," Bill said.
"I was dangerous," I said. "You were she did it and think I would a fine cape you,"
I said.

https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Mario game

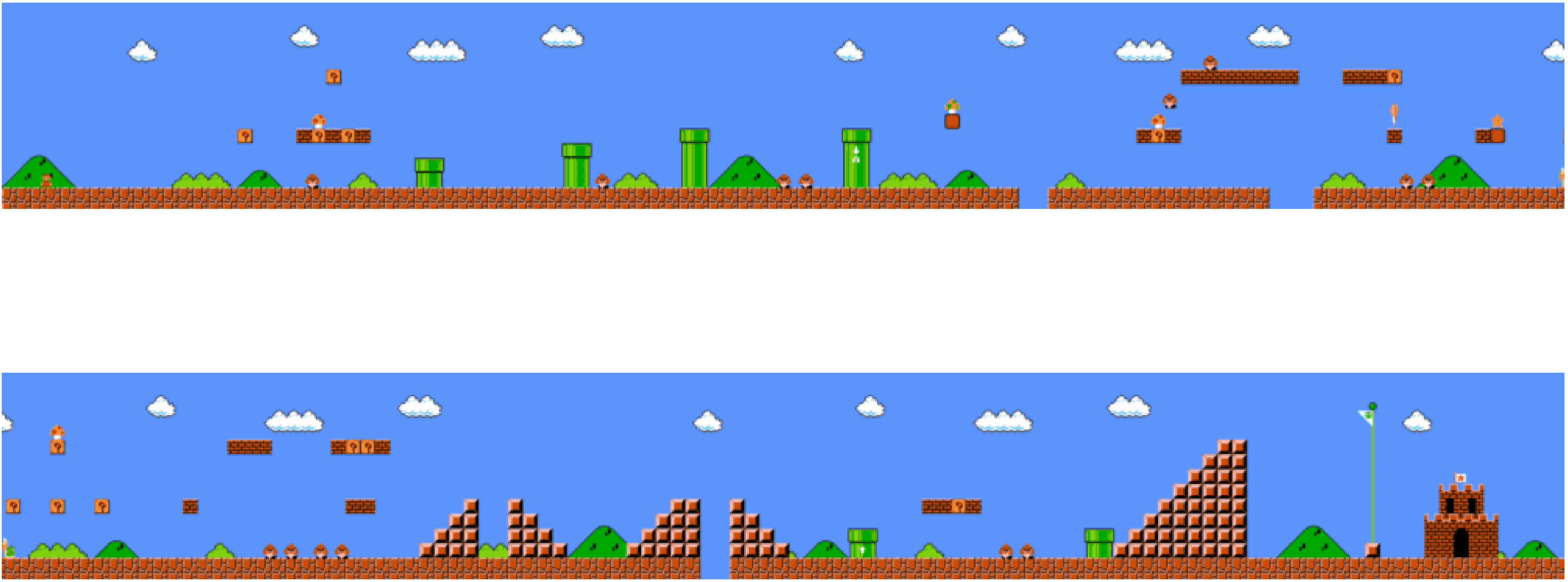
- **Making Mario without actually Making Mario**



https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Mario game

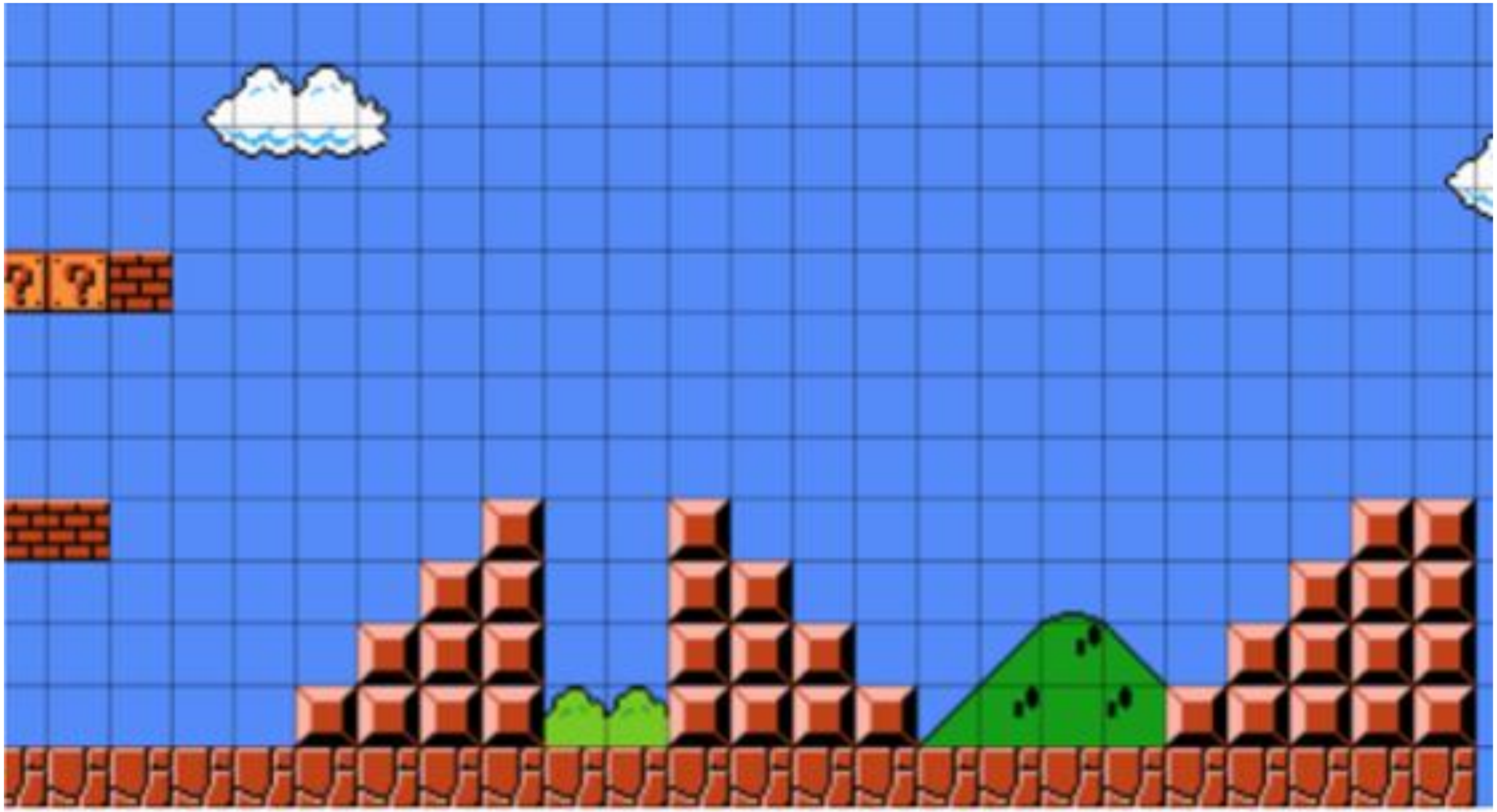
- **The first level from the game**



https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Mario game

- **Simple grid of objects**

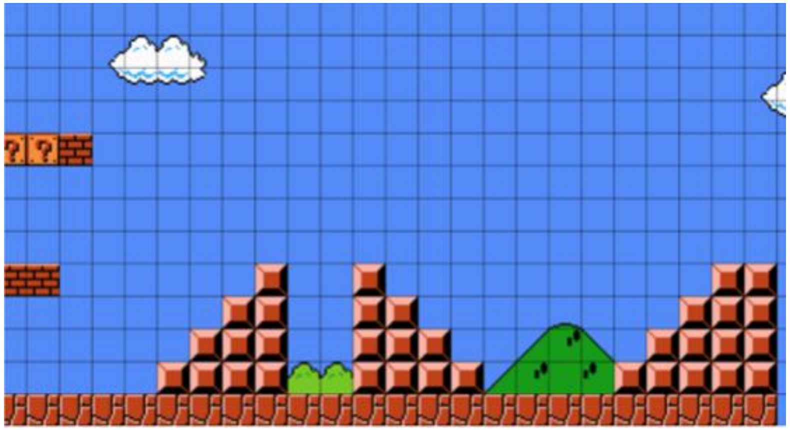


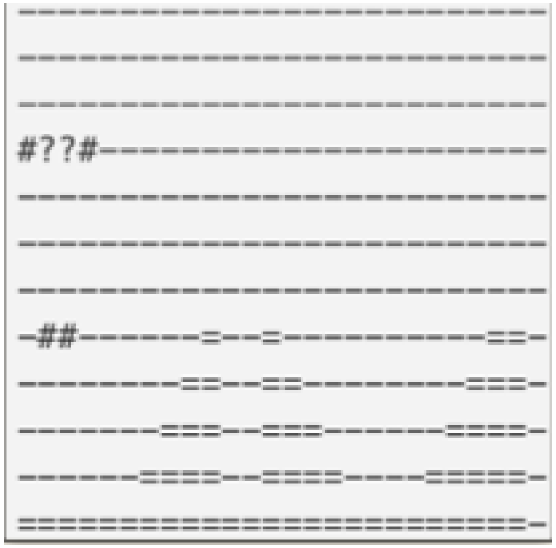
https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Mario game

- **Converting this grid as a sequence of characters**

‘-’ is a blank space
‘=’ is a solid block
‘#’ is a breakable brick
‘?’ is a coin block





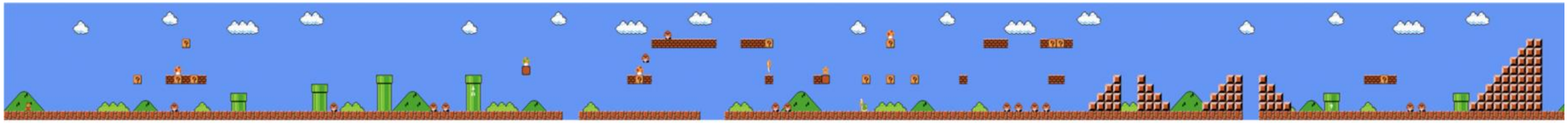
```
--------------------------
--------------------------
--------------------------
#??#----------------------
--------------------------
--------------------------
--------------------------
-##------=--=----------==-
--------==--==--------===-
-------===--===------====-
------====--====----=====-
=========================-
```

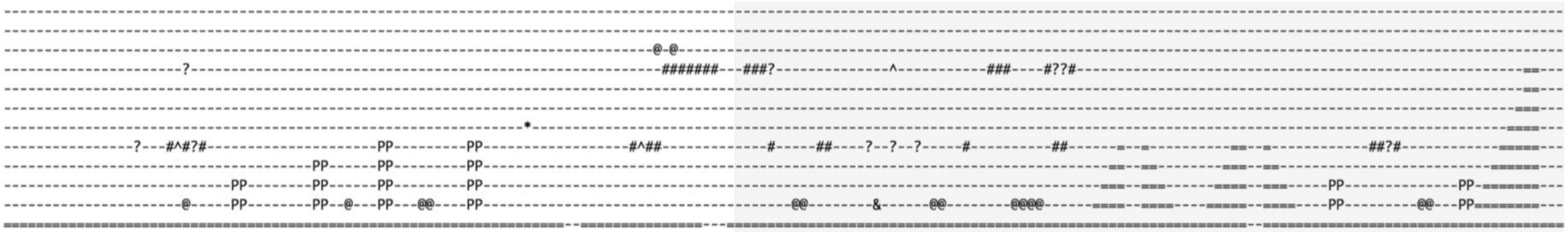
https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Mario game

- **Converting this grid as a sequence of characters**

'-' is a blank space
'=' is a solid block
'#' is a breakable brick
'?' is a coin block





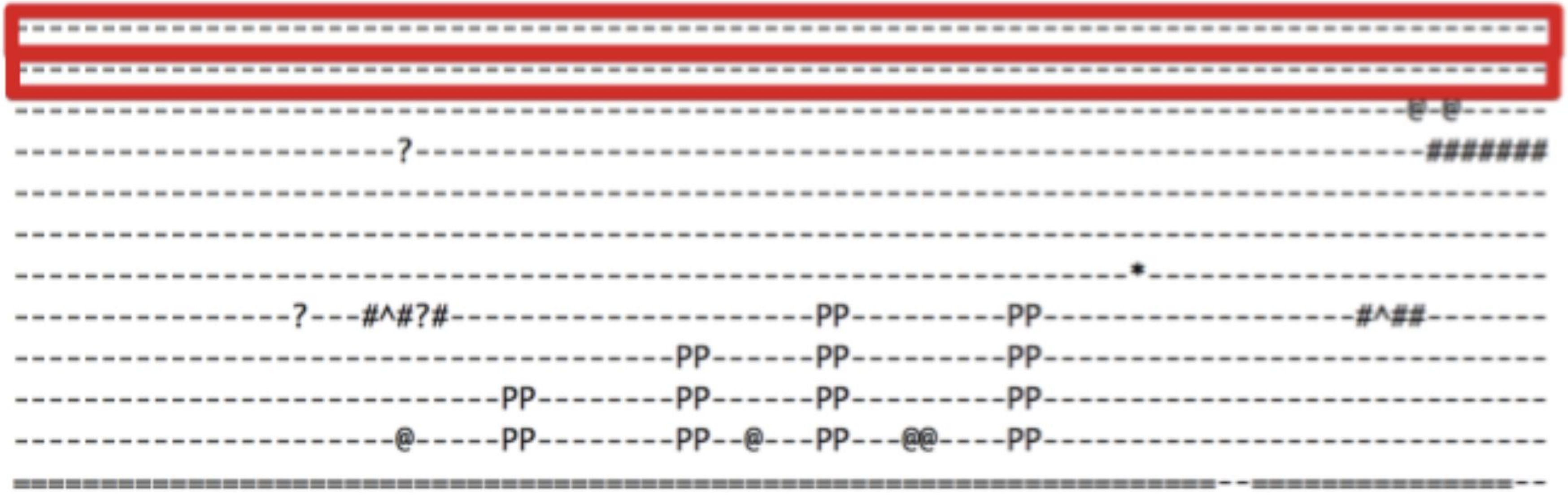
```
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------@-@-------------------------------------------------------------------------------------------------------------
----------------------?----------------------------------------------------------#######---###?--------------^-----------###----#??#-------------------------------------------------------==---
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------==---
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------===---
----------------------------------------------------------------*------------------------------------------------------------------------------------------------------------------------====---
----------------?---#^#?#---------------------PP---------PP------------------#^##-------------#-----##----?--?--?-----#----------##------=--=----------==--=------------##?#------------=====---
--------------------------------------PP------PP---------PP-----------------------------------------------------------------------------==--==--------===--==--------------------------======---
----------------------------PP--------PP------PP---------PP----------------------------------------------------------------------------===--===------====--===-----PP--------------PP-=======---
----------------------@-----PP--------PP--@---PP---@@----PP--------------------------------------@@--------&------@@--------@@@@------====--====----=====--====----PP---------@@---PP========---
=====================================================================--===============---================================================================--=====================================
```

https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Patterns

- **Mario levels don’t really have a pattern in row-by-row**

‘-’ is a blank space
‘=’ is a solid block
‘#’ is a breakable brick
‘?’ is a coin block

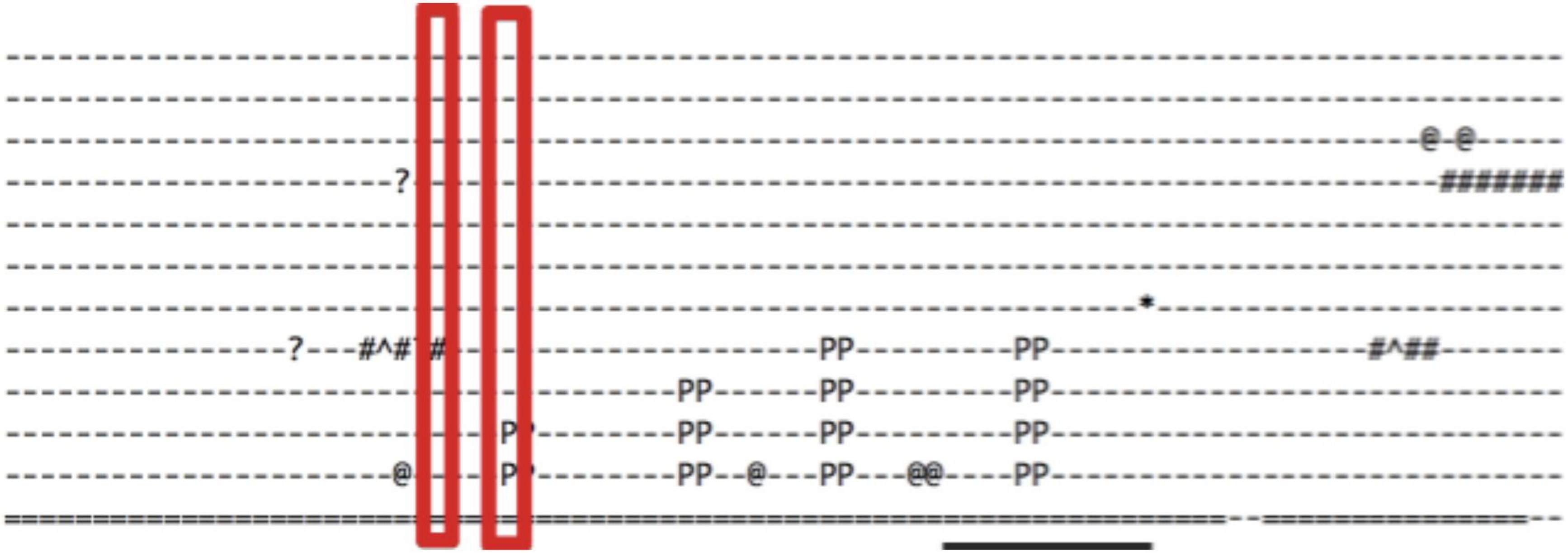


https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Patterns

- **There's a real pattern in column-by-column**

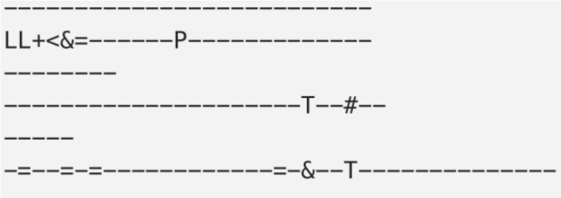
'-' is a blank space
'=' is a solid block
'#' is a breakable brick
'?' is a coin block



https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

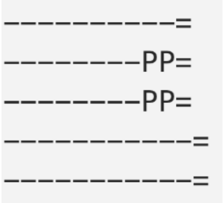
Generating mario level

- **After a little training, our model is generating junk**



```
--------------------------
LL+<&=------P-------------
--------
---------------------T--#--
-----
-=--=-=------------=-&--T--------------
```

- **After several thousand iterations**
  - **The model has almost figured out that each line should be the same length.**
  - **It has even started to figure out some of the logic of Mario**
  - **The pipes in mario are two blocks wide and at least two blocks high, so the “P”s in the data should appear in 2x2 clusters**

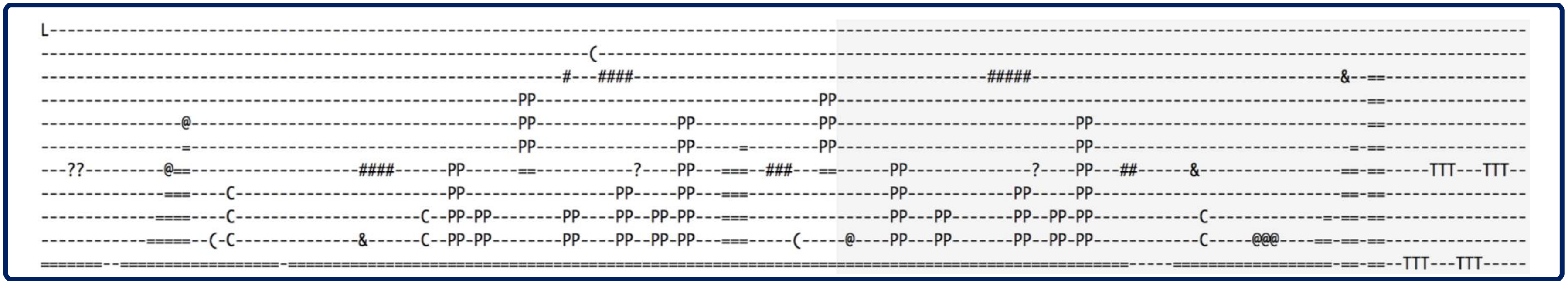


```
----------=
--------PP=
--------PP=
-----------=
-----------=
```

https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Generating mario level

- **With a lot more training**
  - It put a Lakitu (the monster that floats on a cloud) up in the sky at the beginning of the level—just like in a real Mario level.
  - It knows that pipes floating in the air should be resting on top of solid blocks and not just hanging in the air.
  - It places enemies in logical places.
  - It doesn't create anything that would block a player from moving forward.
  - It feels like a real level from Super Mario Bros. 1



```
L-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------(---------------------------------------------------------------------------------------------------------
-----------------------------------------------------------#---####----------------------------------------#####-----------------------------------&--==----------------
------------------------------------------------------PP--------------------------------PP------------------------------------------------------------==----------------
----------------@-------------------------------------PP----------------PP--------------PP---------------------------PP-------------------------------==----------------
----------------=-------------------------------------PP----------------PP-----=--------PP---------------------------PP-----------------------------=-==----------------
---??---------@==-------------------####------PP------==-----------?----PP---===--###---==------PP--------------?----PP---##------&----------------==-==-----TTT---TTT--
--------------===----C------------------------PP-----------------PP-----PP---===----------------PP------------PP-----PP----------------------------==-==----------------
-------------====----C---------------------C--PP-PP--------PP----PP--PP-PP---===----------------PP---PP-------PP--PP-PP------------C-------------=-==-==----------------
------------=====--(-C--------------&------C--PP-PP--------PP----PP--PP-PP---===-----(-----@----PP---PP-------PP--PP-PP------------C-----@@@----==-==-==----------------
=======--==================-===============================================================================================-----==================-==-==--TTT---TTT-----
```

https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3

Mario game

- This model is generated from very little data
- There just aren’t enough levels in the Mario game
- With the thousands of user-created levels,
  we could make an amazing model.
- The difference between a good and a bad program
  - How much data you have to train your models

Unsupervised pre-training(UP)

- **Details**
  - **Learning without labelling**
  - **Learning only once before joint training**
  - **Regularizer or parameter init that reduces test error**
- **Basic idea of UP**
  - **Features from UP are also useful for SL by using representation**
  - **To draw a car, you need to the number of wheels**

Contents

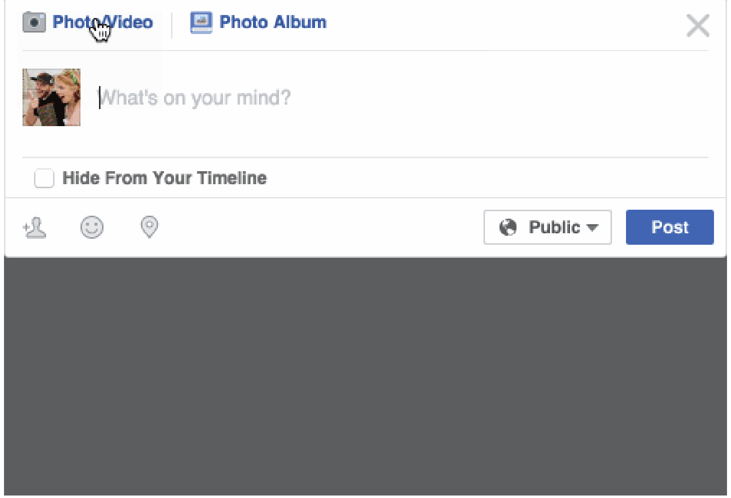
- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Representation

- Information processing becomes easy or difficult
  - Depending on how information is represented
- Numbers are easy to divide
  - But difficult if they are romanized
- Good representation makes it easy to follow up
- Neural network learns how to represent
- The previous steps learn how to express
  - To the last softmax classifier
- Many representation learning faces a tradeoff
  - Between preserving information and obtaining good attributes

Face recognition

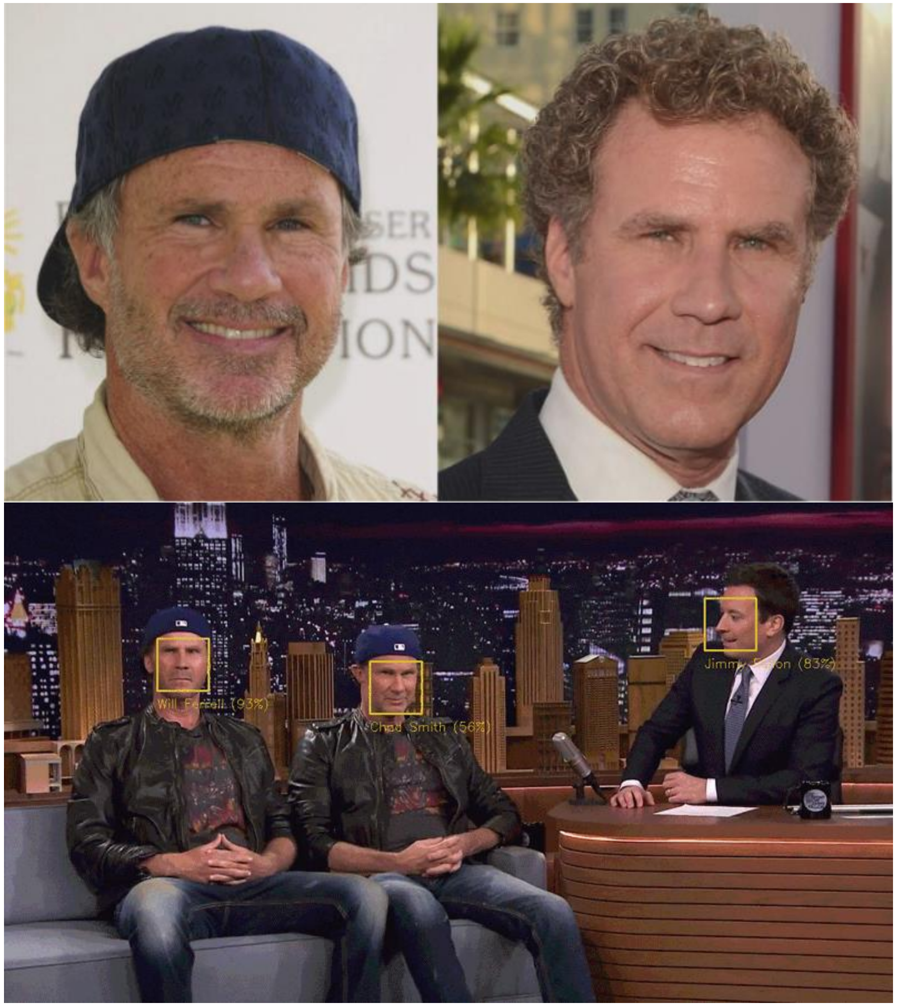
- Taging everyone for you like magic by **Facebook**
- Recognizing your friends' faces
  after they have been tagged only a few times.
- **98% accuracy** which is much **as good as humans can do!**
- Pushing this tech to the limit to **solve a challenging problem**
  - Will Ferrell apart from Chad Smith


Photo/Video
Photo Album
What's on your mind?
Hide From Your Timeline
Public
Post

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition

- **They are different people!**


Will Ferrell (93%)
Chad Smith (56%)
Jimmy
on (83%)

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process

- 1: Look at a picture and find all the faces in it
- 2: Focus on each face and be able to understand that
  - Even if a face is turned in a weird direction or in bad lighting, it is still the same person.
- 3: Pick out unique features of the face
  - You can use to tell it apart from other people
- 4: Compare the unique features of that face to all the people

CHANNEL
NEWS TEAM
4

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

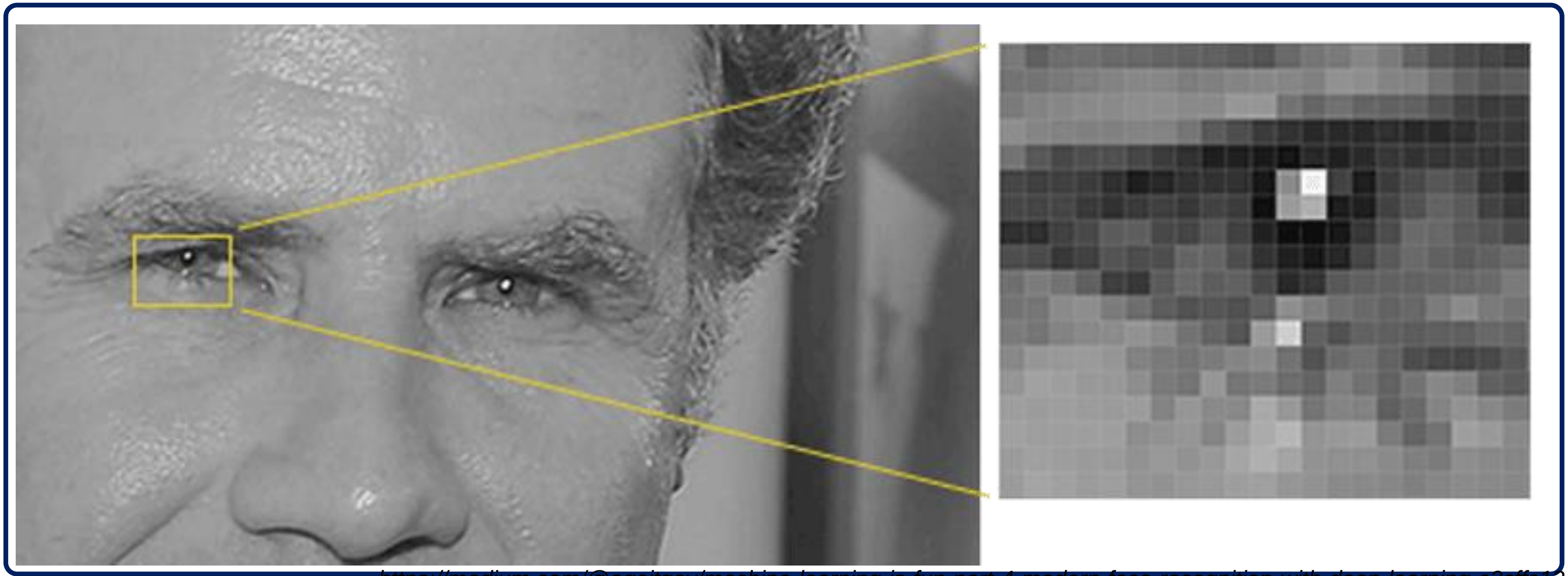
Face recognition process

- To find faces in an image
  - we'll start by making our image black and white because we don't need color data to find faces

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process

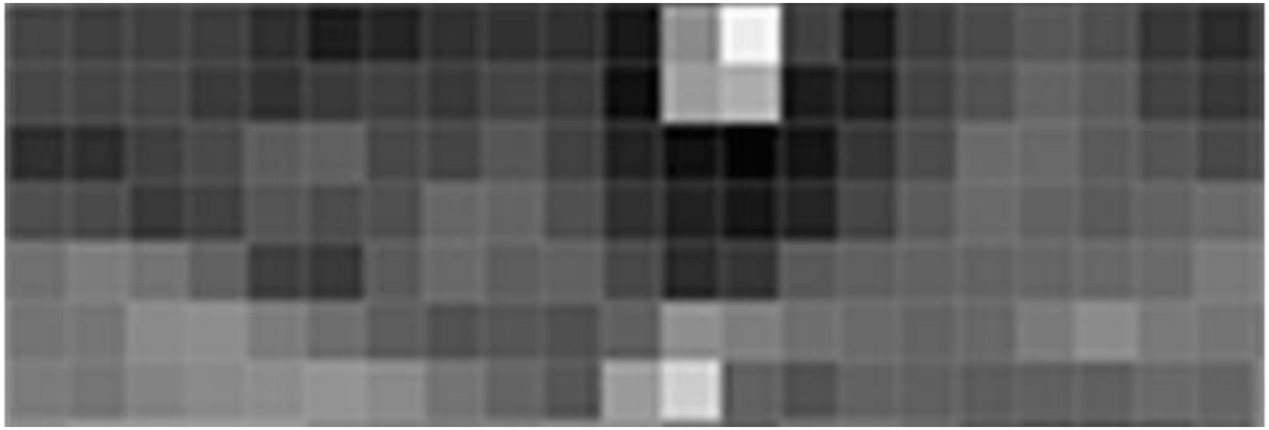
- **For every single pixel, look at the pixels that surrounding it**
- **Our goal**
  - **Figuring out how dark the current pixel**
  - **Then we want to draw an arrow showing in which direction the image is getting darker**



https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process

- Every pixel is replaced by an arrow
- These arrows are called *gradients*
- They show the flow from light to dark



https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process

- There's a reason for replacing the pixels with gradients
  - If we analyze pixels directly, dark images and light images of the same person will have different pixel values
- But saving the gradient for every single pixel
  - Too much detail
  - Missing the forest for the trees

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

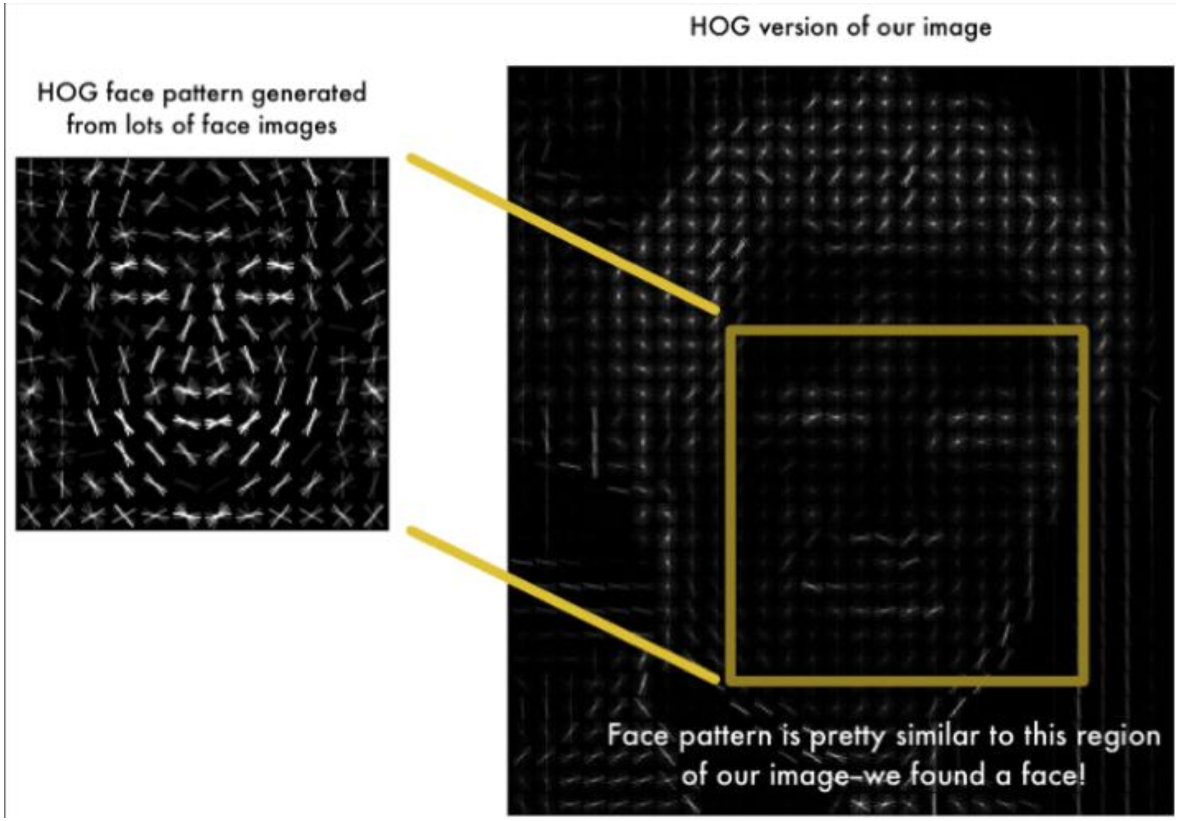
Face recognition process

- Breaking up the image into squares of 16x16 pixels each
- The end result is we turn the original image
  - Into a very simple representation

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process

- **To find faces in this HOG image**
  - Find the part of our image that looks the most similar to a known HOG pattern that was extracted from other faces


HOG face pattern generated from lots of face images
HOG version of our image
Face pattern is pretty similar to this region of our image–we found a face!

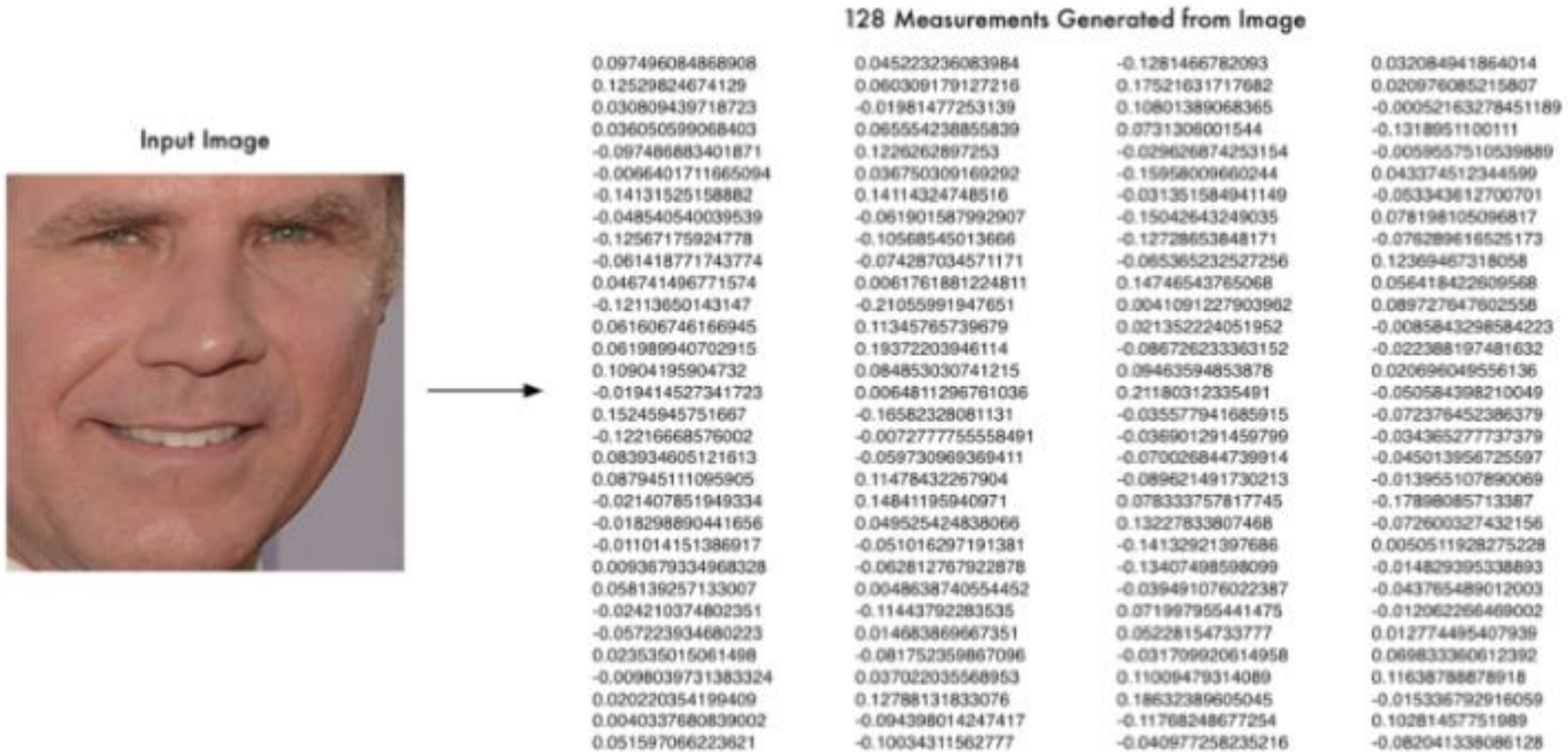
https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process

- There's actually a huge problem with that approach
  - That would take way too long
  - They need to be able to recognize faces in milliseconds, not hours
- What we need is a way to extract a few measurements from each face
  - The size of each ear, the space between the eyes, etc.
- After repeating this step millions of times for millions of images of thousands of different people
  - The neural network learns to reliably generate 128 measurements for each person
  - Machine learning people call the 128 measurements of each face an embedding

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Face recognition process



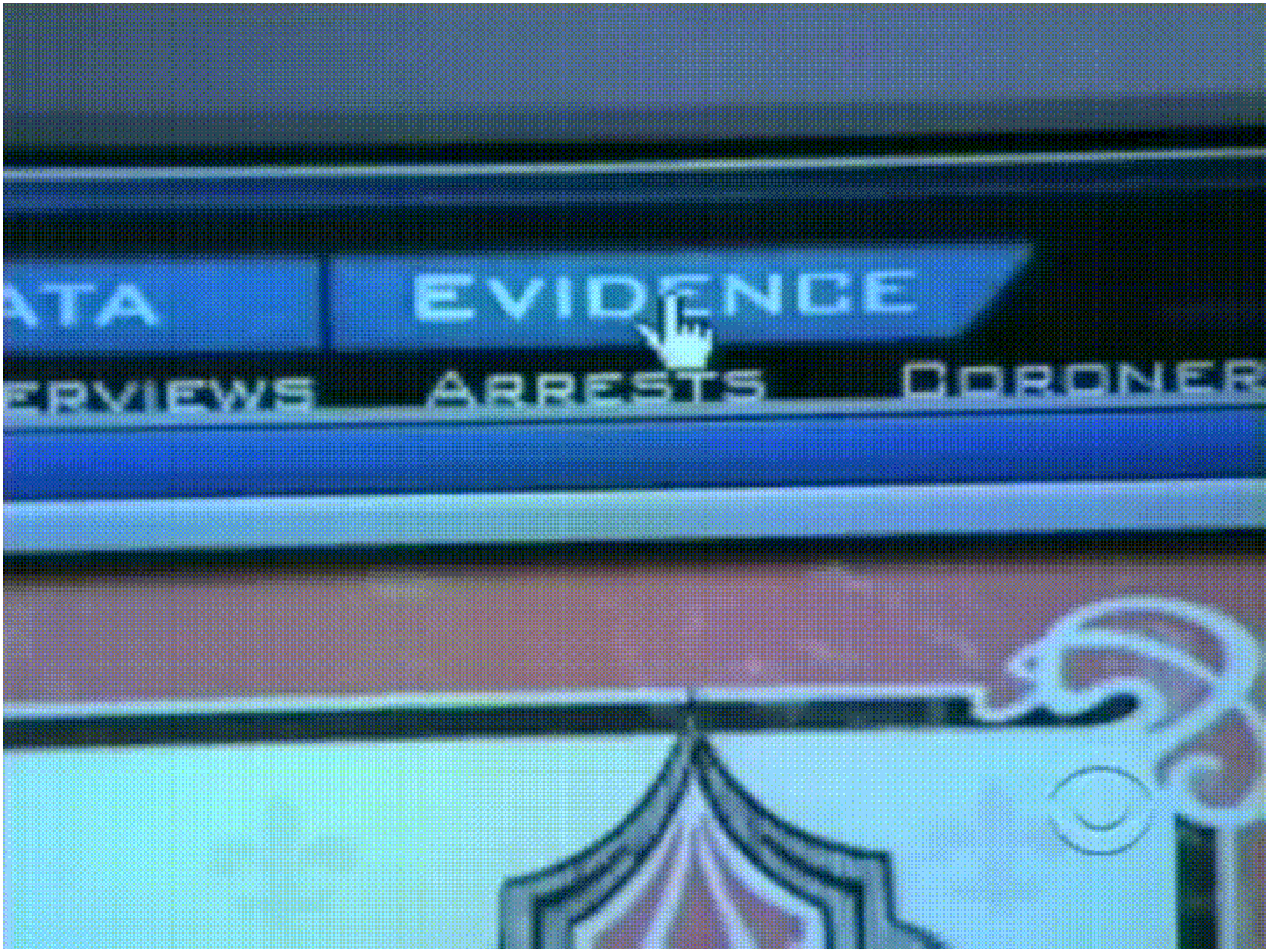
Input Image

128 Measurements Generated from Image

```
0.097496084868908      0.045223236083984      -0.1281466782093       0.032084941864014
0.12529824674129       0.060309179127216      0.17521631717682       0.020976085215807
0.030809439718723      -0.01981477253139      0.10801389068365       -0.00052163278451189
0.036050599068403      0.065554238855839      0.0731306001544        -0.1318951100111
-0.097486883401871     0.1226262897253        -0.029626874253154     -0.0059557510539889
-0.0066401711665094    0.036750309169292      -0.15958009660244      0.043374512344599
-0.14131525158882      0.14114324748516       -0.031351584941149     -0.053343612700701
-0.048540540039539     -0.061901587992907     -0.15042643249035      0.078198105096817
-0.12567175924778      -0.10568545013666      -0.12728653848171      -0.076289616525173
-0.061418771743774     -0.074287034571171     -0.065365232527256     0.12369467318058
0.046741496771574      0.0061761881224811     0.14746543765068       0.056418422609568
-0.12113650143147      -0.21055991947651      0.0041091227903962     0.089727647602558
0.061606746166945      0.11345765739679       0.021352224051952      -0.0085843298584223
0.061989940702915      0.19372203946114       -0.086726233363152     -0.022388197481632
0.10904195904732       0.084853030741215      0.09463594853878       0.020696049556136
-0.019414527341723     0.0064811296761036     0.21180312335491       -0.050584398210049
0.15245945751667       -0.16582328081131      -0.035577941685915     -0.072376452386379
-0.12216668576002      -0.0072777755558491    -0.036901291459799     -0.034365277737379
0.083934605121613      -0.059730969369411     -0.070026844739914     -0.045013956725597
0.087945111095905      0.11478432267904       -0.089621491730213     -0.013955107890069
-0.021407851949334     0.14841195940971       0.078333757817745      -0.17898085713387
-0.018298890441656     0.049525424838066      0.13227833807468       -0.072600327432156
-0.011014151386917     -0.051016297191381     -0.14132921397686      0.0050511928275228
0.0093679334968328     -0.062812767922878     -0.13407498598099      -0.014829395338893
0.058139257133007      0.0048638740554452     -0.039491076022387     -0.043765489012003
-0.024210374802351     -0.11443792283535      0.071997955441475      -0.012062266469002
-0.057223934680223     0.014683869667351      0.05228154733777       0.012774495407939
0.023535015061498      -0.081752359867096     -0.031709920614958     0.069833360612392
-0.0098039731383324    0.037022035568953      0.11009479314089       0.11638788878918
0.020220354199409      0.12788131833076       0.18632389605045       -0.015336792916059
0.0040337680839002     -0.094398014247417     -0.11768248677254      0.10281457751989
0.051597066223621      -0.10034311562777      -0.040977258235216     -0.082041338086128
```

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

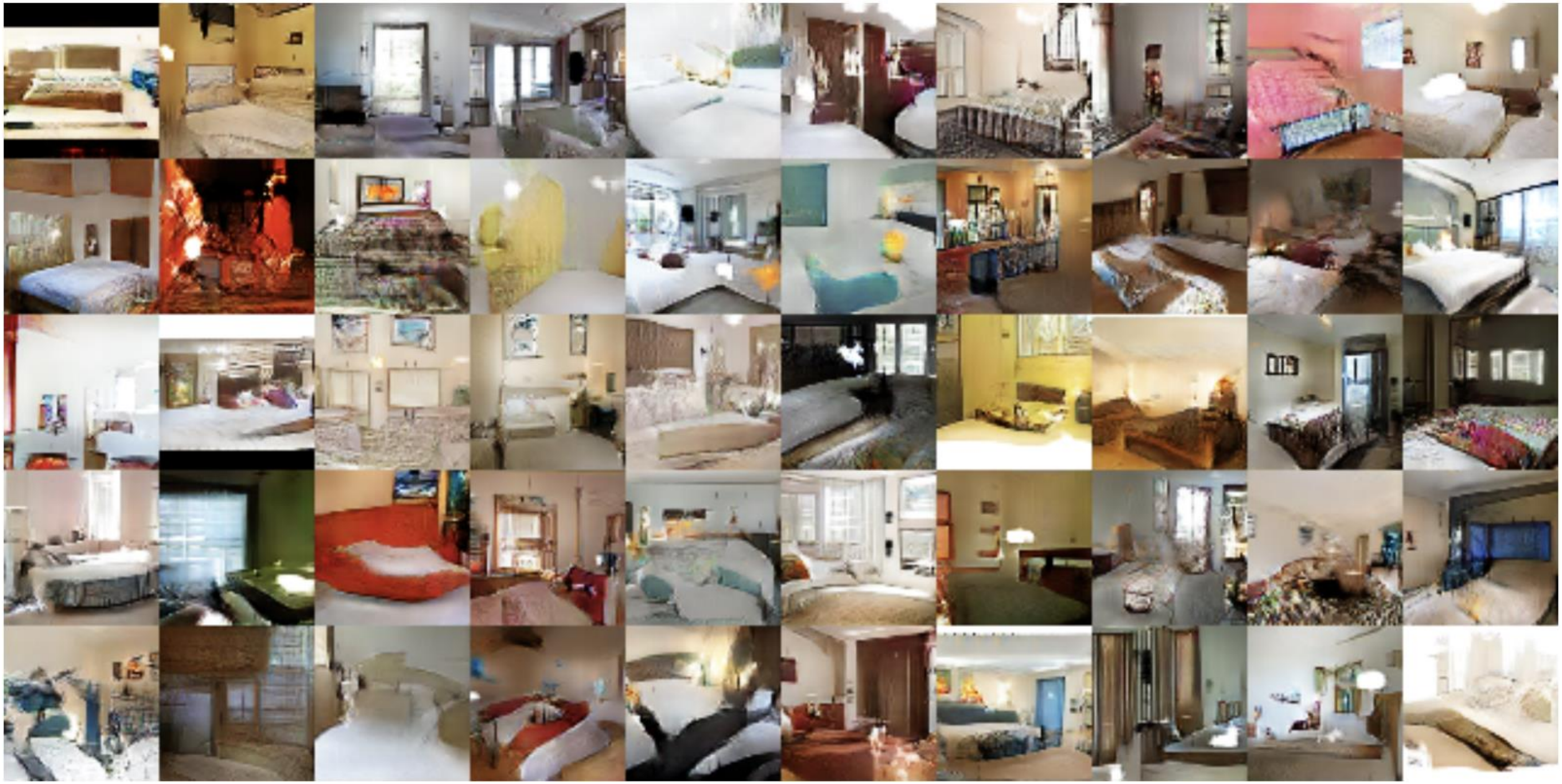
Face recognition process


ATA
EVIDENCE
ERVIEWS
ARRESTS
CORONER

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Generative models

- **These pictures of bedrooms were dreamt up by a DCGAN**



https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Generative models

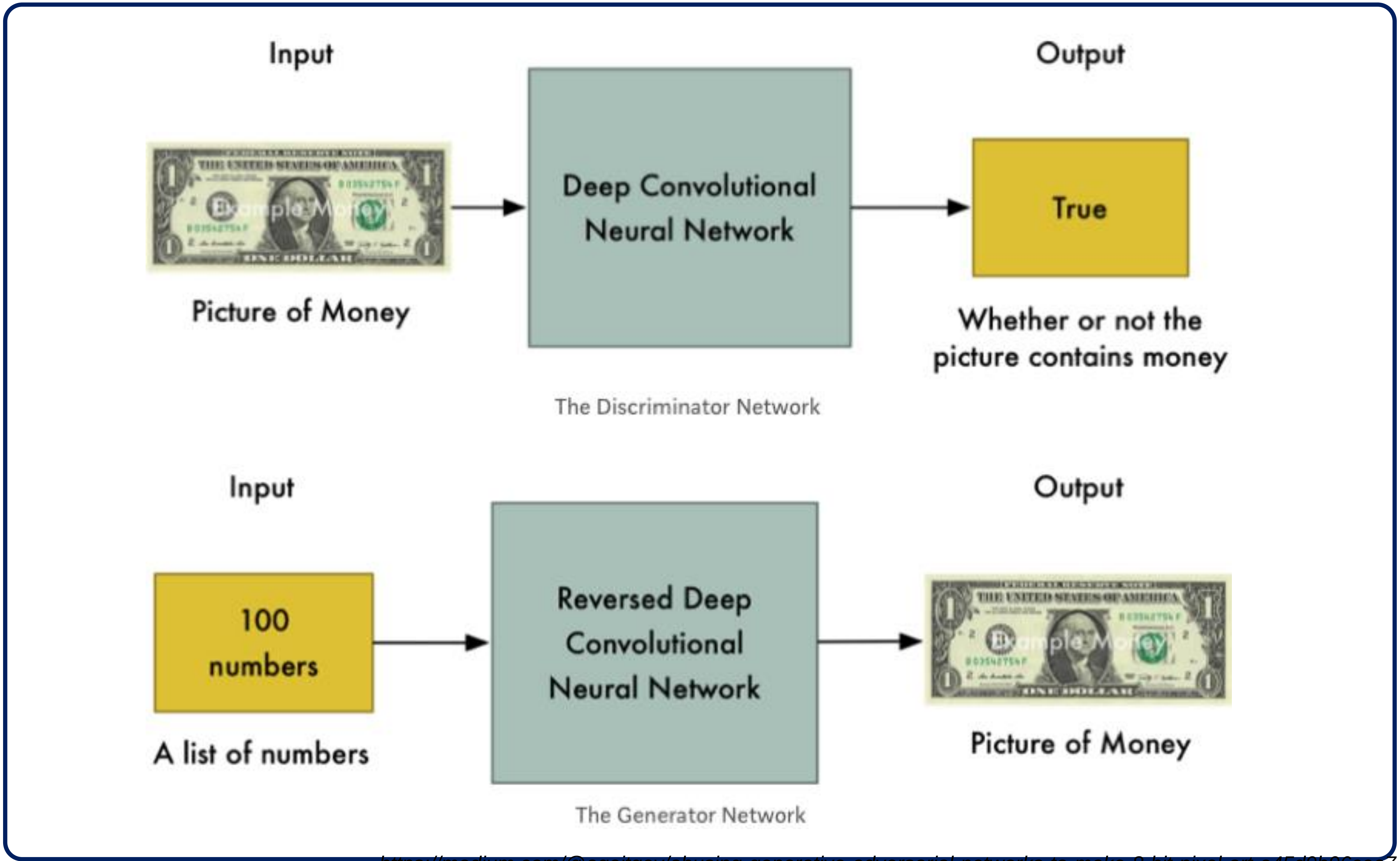
- If the computer was able to do this and the pictures
  - It produced had the right number of legs, tails, and ears
  - Computer knows what parts go into making up a "dog"

https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Generative models

- To build a DCGAN, we create two deep neural networks
- Then we make them fight against each other
- In the process, they both become stronger
- This first neural network is called the Discriminator
- This second neural network is called the Generator

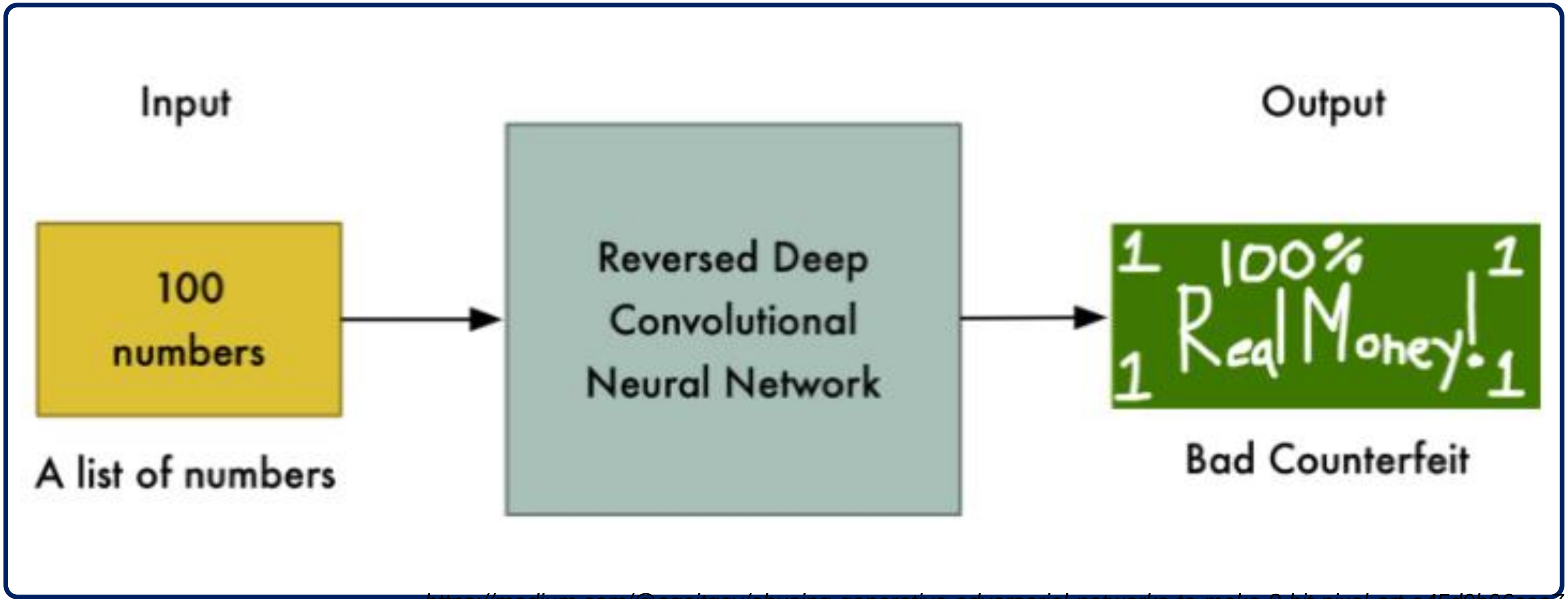
Generative models


Input
Output
Picture of Money
Deep Convolutional Neural Network
True
Whether or not the picture contains money
The Discriminator Network
Input
Output
100 numbers
A list of numbers
Reversed Deep Convolutional Neural Network
Picture of Money
The Generator Network

https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Generative models

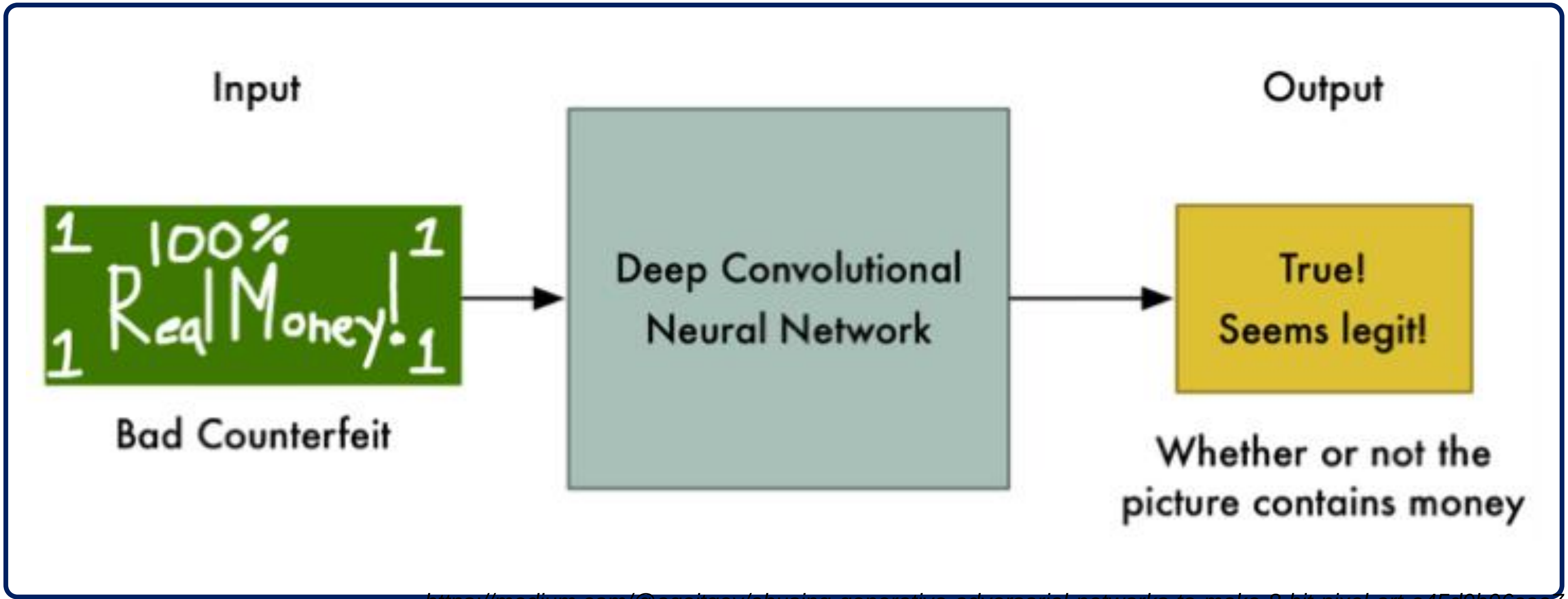
- **Let's make them battle!**
- **In the first round**
  - **The Generator makes the first (terrible) fake dollar**


Input
100 numbers
A list of numbers
Reversed Deep Convolutional Neural Network
Output
1 100% 1
Real Money!
1 1
Bad Counterfeit

https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Generative models

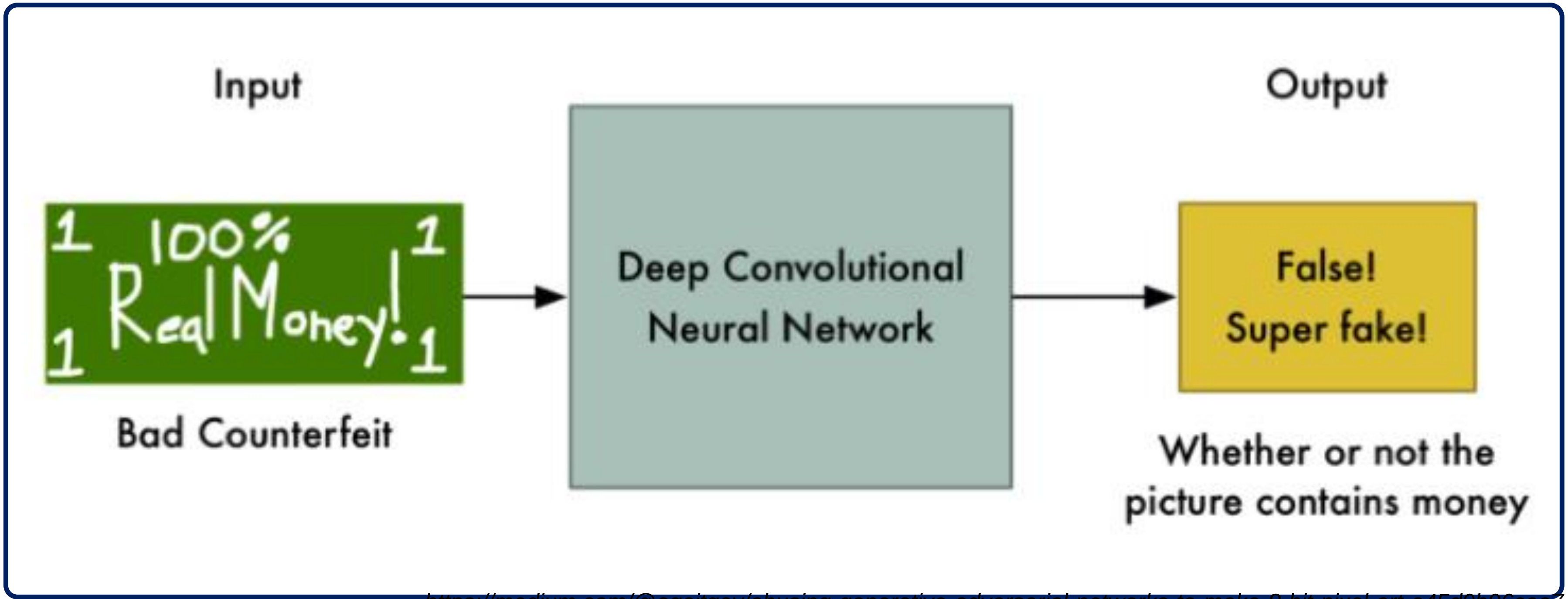
- **The Discriminator is equally terrible at it's job, so it won't know the difference**


Input
1 100% 1
Real Money!
1 1
Bad Counterfeit
Deep Convolutional Neural Network
Output
True! Seems legit!
Whether or not the picture contains money

https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Generative models

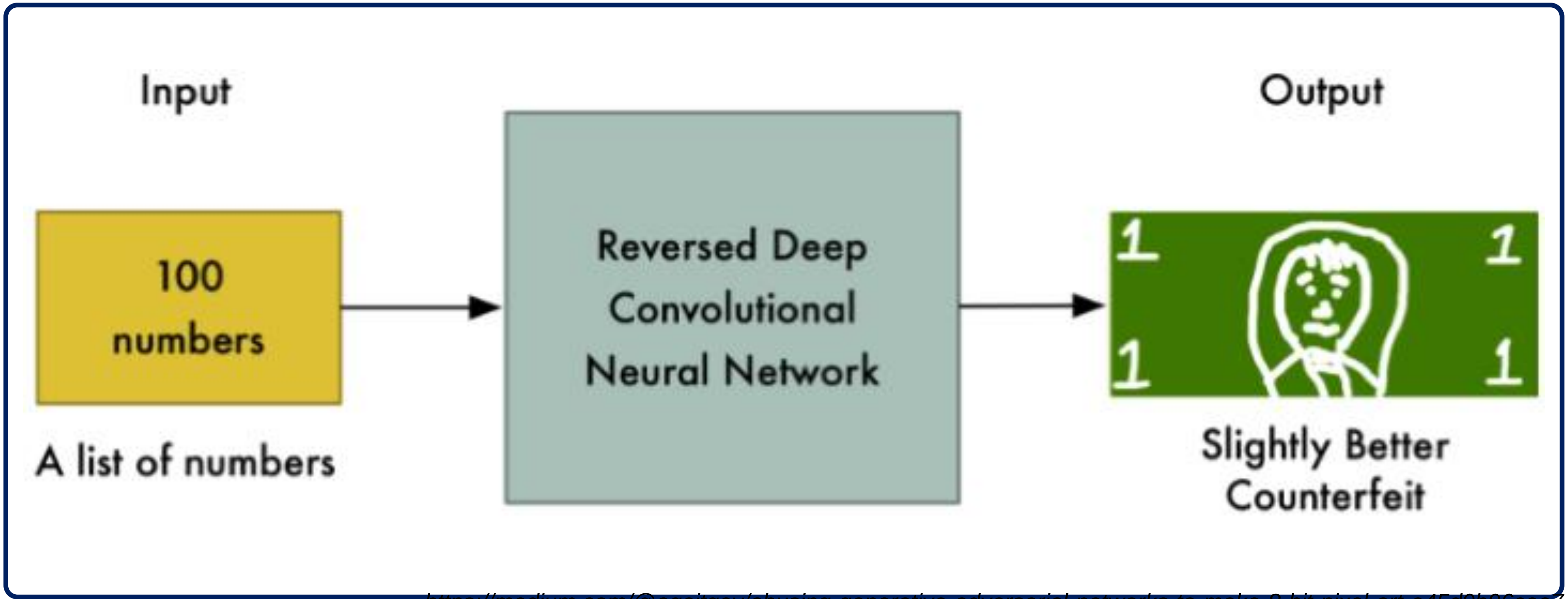
- **The Discriminator levels up!**
- **It now can spot very bad fake dollars.**


Input
Output
1 100% 1
Real Money!
1 1
Bad Counterfeit
Deep Convolutional
Neural Network
False!
Super fake!
Whether or not the
picture contains money

https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Generative models

- **The Generator makes a slightly better counterfeit dollar**
- **This back-and-forth game between them**
  - Continuing thousands of times until both networks are experts


Input
100 numbers
A list of numbers
Reversed Deep Convolutional Neural Network
Output
1
1
1
1
Slightly Better Counterfeit

https://medium.com/@ageitgey/abusing-generative-adversarial-networks-to-make-8-bit-pixel-art-e45d9b96cee7

Contents

- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Clustering

- **Details**
  - Clustering **is a kind of Unsupervised Learning,** finding a **cluster** that best describes given data **without label**
  - Classification requires **data and labels for each data**
  - In practice, there are many cases where **data exists but it does not know what label or category it is**
  - Sometimes it is necessary **to explain the data** through a method other than classfication
- **K-means**
- **Gaussian Mixture Model(GMM)**

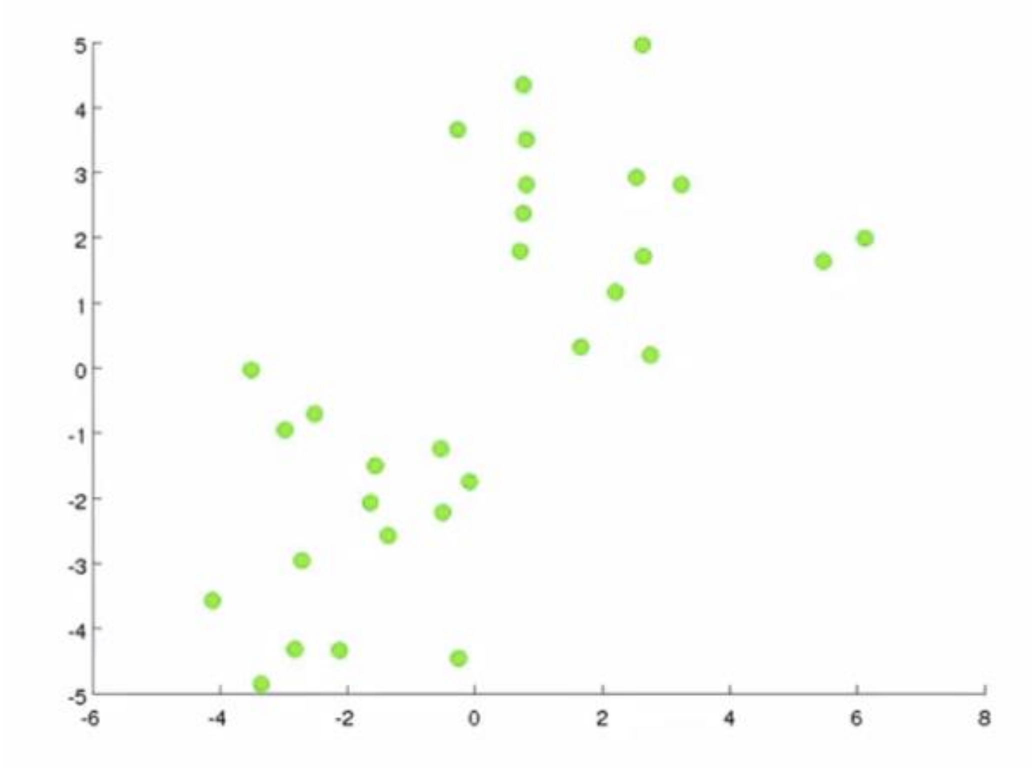
Contents

- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

K-means clustering

- **Find the K clusters that best describes the data**

$$\min_{b,w} \sum_{i}^{n} \sum_{j}^{k} w_{ij} \|x_i - b_j\|_2^2 \text{ s.t. } \sum_{j} w_{ij} = 1, \forall j$$

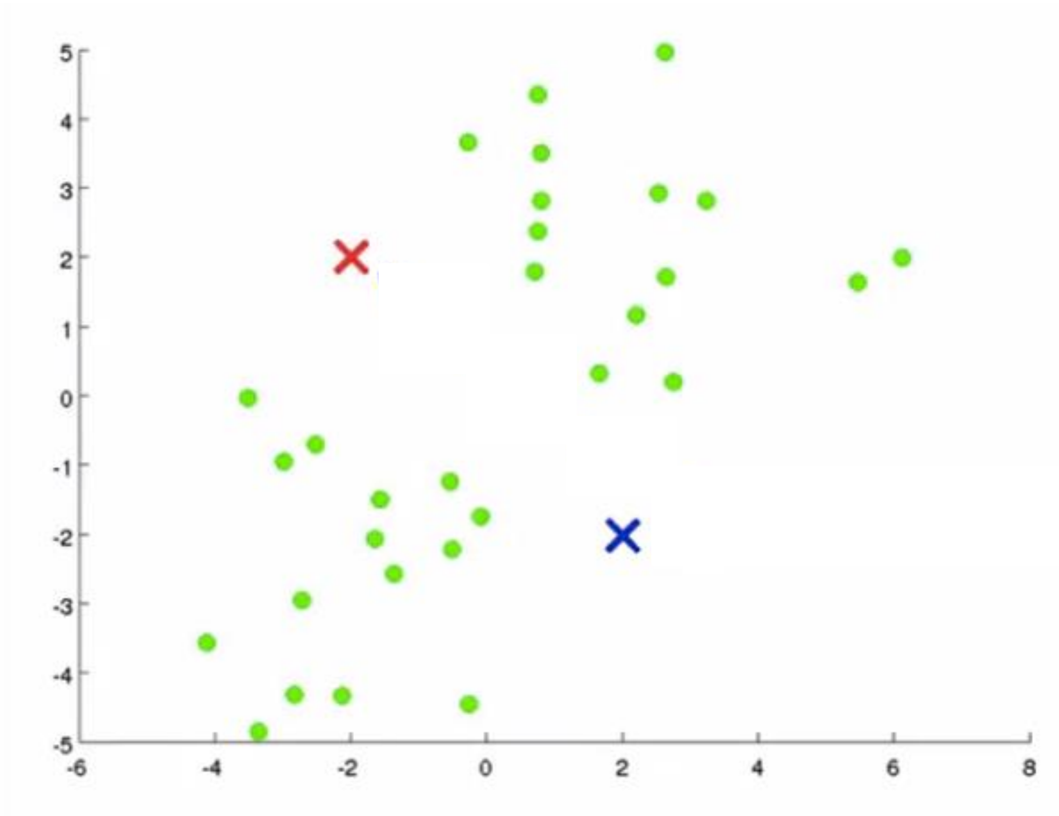


n: # of data
k: # of clusters
I: index of data
j: index of cluster
b_j: center of j-th cluster
w_{ij}: if i-th data is in j-th cluster, it is 1
else, it is 0

** Slides from Andrew Ng(Stanford Univ.), “Machine Learning”*

K-means clustering

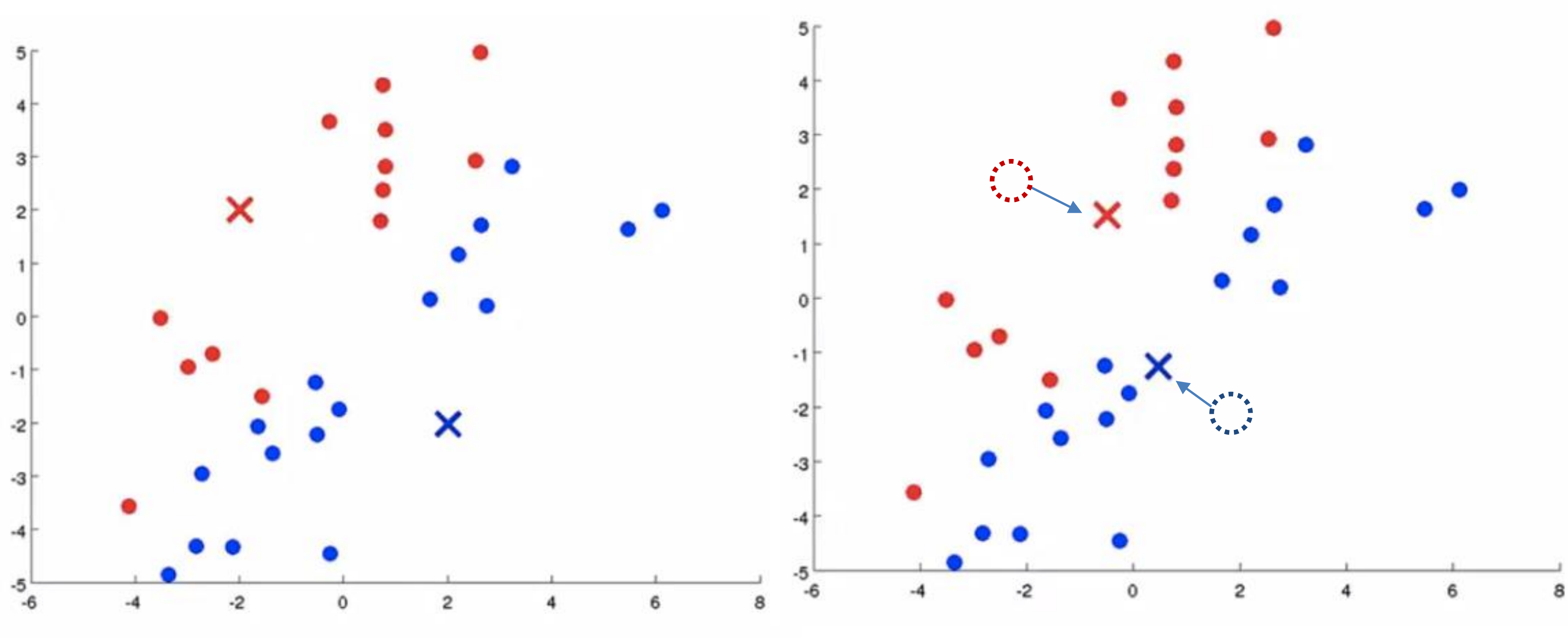
- **Number of cluster k = 2,**
  - Randomly initialize “centroids”



** Slides from Andrew Ng(Stanford Univ.), “Machine Learning”*

K-means clustering

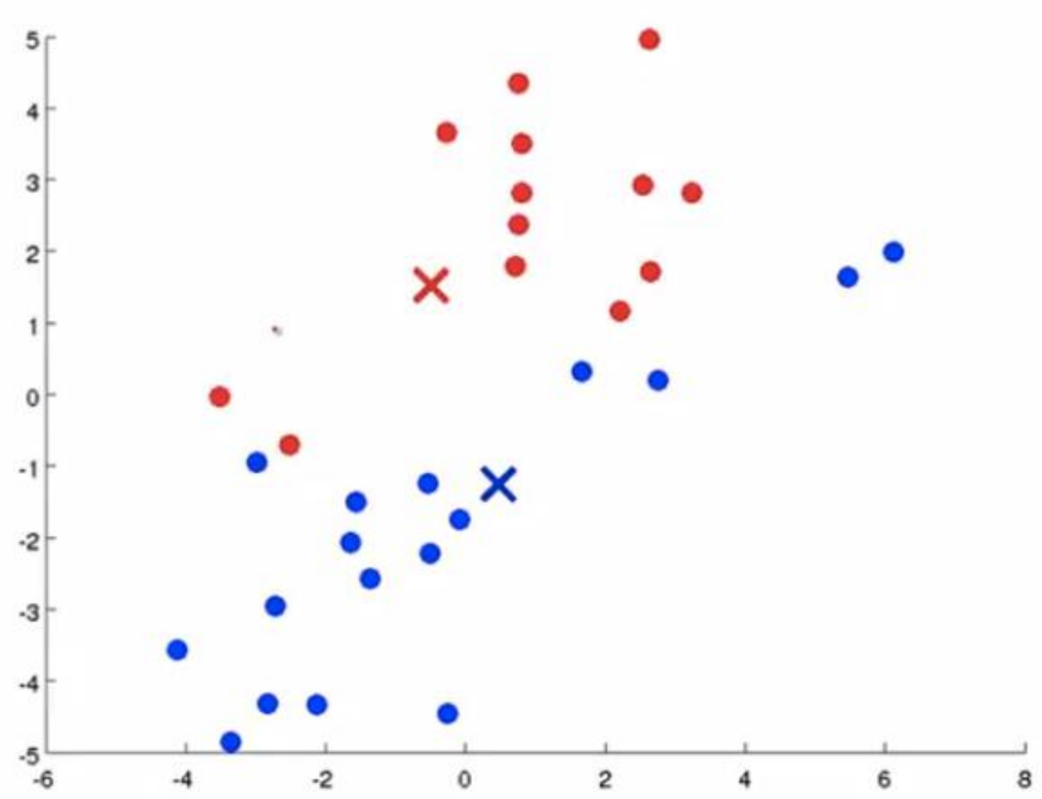
- **Number of cluster k = 2,**
  - Assign cluster membership
  - Update the cluster centroid (average of the data points in each cluster)



** Slides from Andrew Ng(Stanford Univ.), "Machine Learning"*

K-means clustering

- **Number of cluster k = 2,**
  - Update cluster membership
  - Repeat those procedure until no membership update



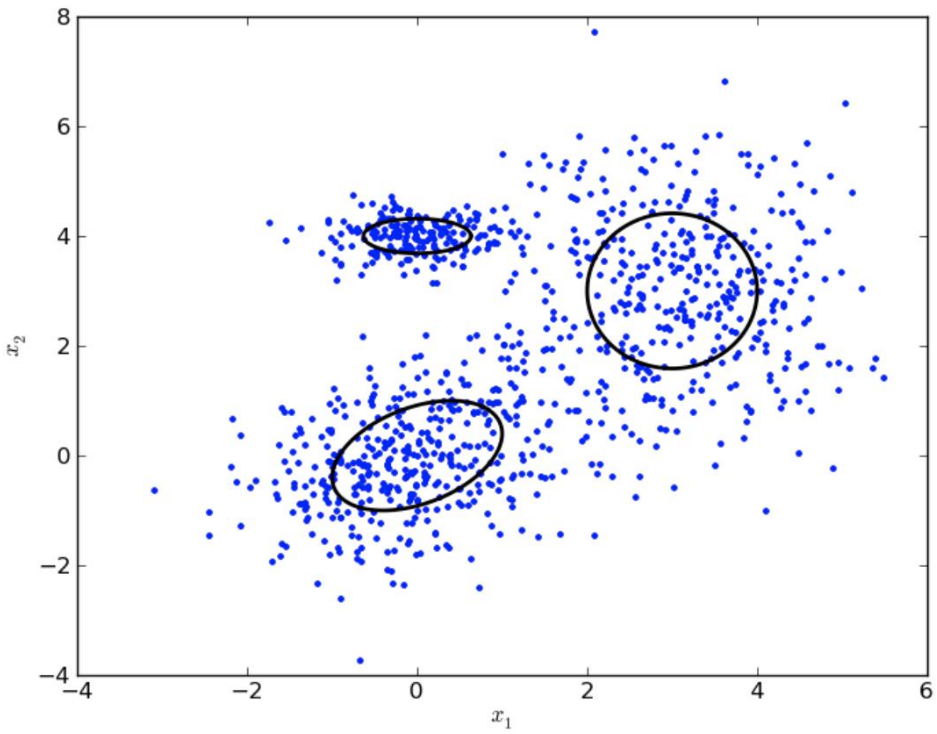
** Slides from Andrew Ng(Stanford Univ.), “Machine Learning”*

Contents

- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Gaussian Mixture Model

- The data is composed of 'Mixture' of 'Gaussian'
- The figure below is the result of finding the gaussian distributions assuming that it consists of three gaussian.



Contents

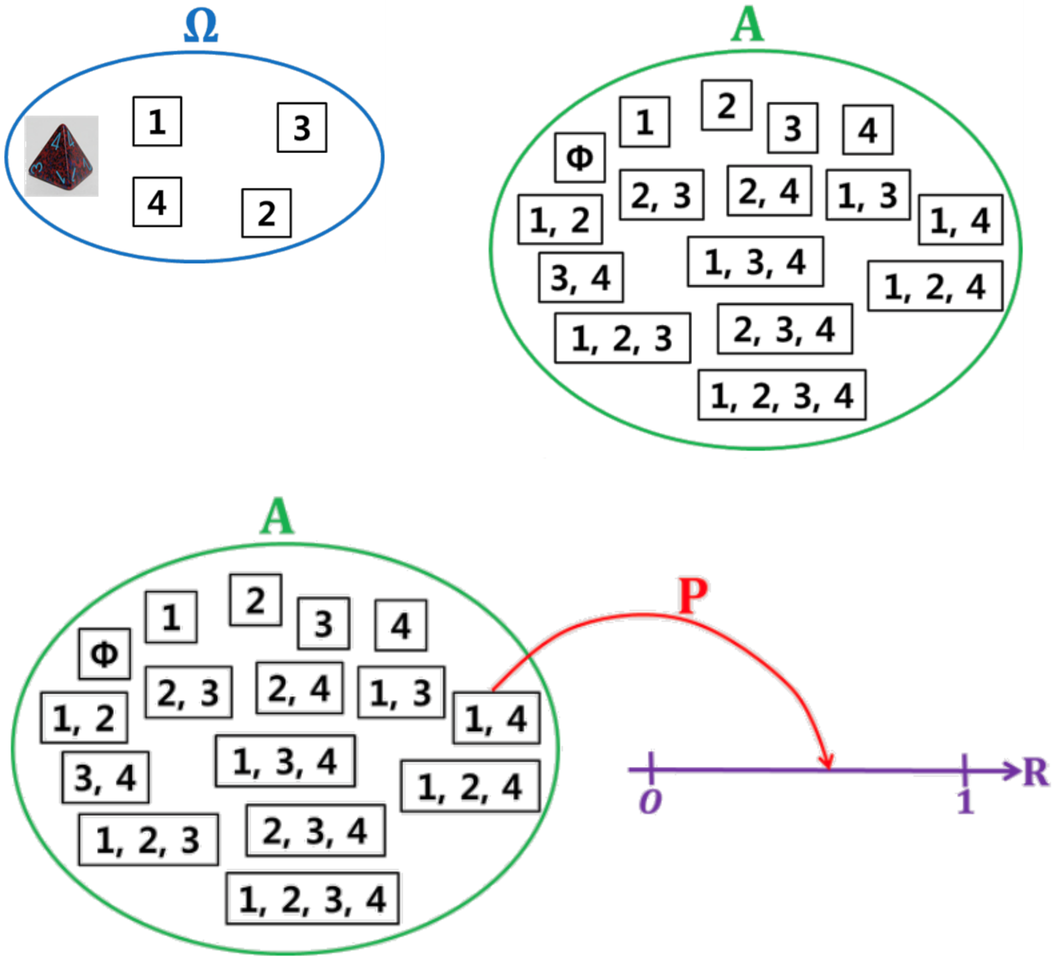
- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

EM Algorithm

- **Probability space**
  - (Ω, A, P)
  - Ω: a sample space
  - A: a set of events
  - P: the assignment of probability to the events

EM Algorithm

- **Probability space** (Ω, A, P)


Ω
1
3
4
2
A
Φ
1
2
3
4
1, 2
2, 3
2, 4
1, 3
1, 4
3, 4
1, 3, 4
1, 2, 4
1, 2, 3
2, 3, 4
1, 2, 3, 4
A
Φ
1
2
3
4
1, 2
2, 3
2, 4
1, 3
1, 4
3, 4
1, 3, 4
1, 2, 4
1, 2, 3
2, 3, 4
1, 2, 3, 4
P
0
1
R

EM Algorithm

X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
θ: Parameter

- **Joint distribution**

$$p(\mathbf{X}, \mathbf{Z}|\theta)$$

- **Marginal probability distribution**

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

EM Algorithm

X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
θ: Parameter

- **Marginal probability distribution**

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

- **Maximum likelyhood estimation**

$$\max_{\theta} p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, Z|\theta)$$

http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

EM Algorithm

X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
$q()$: Probability distribution of Z
θ: Parameter

- **Maximum likelyhood estimation**

$$\max_{\theta} p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, Z|\theta)$$

- **Maximum log likelyhood estimation**

$$\ln p(\mathbf{X}|\theta) = L(q, \theta) + KL(q\|p)$$

$$L(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} \right\}$$

$$KL(q\|p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})} \right\}$$

http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

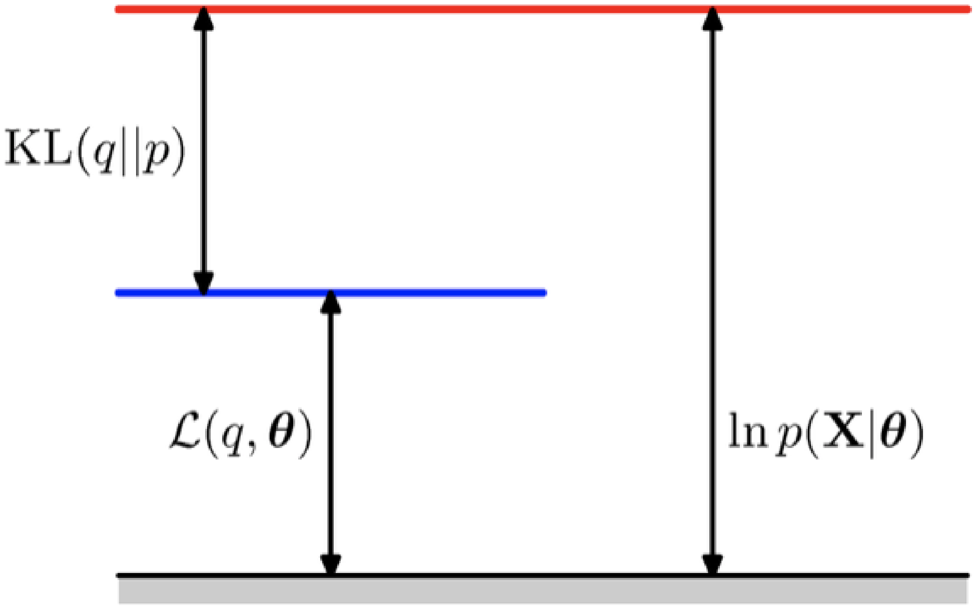
EM Algorithm

X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
$q()$: Probability distribution of Z
θ: Parameter

- **Maximum likelyhood estimation**

$$\ln p(\mathbf{X}|\theta) = L(q, \theta) + KL(q||p)$$

- **Current status**



$\mathrm{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

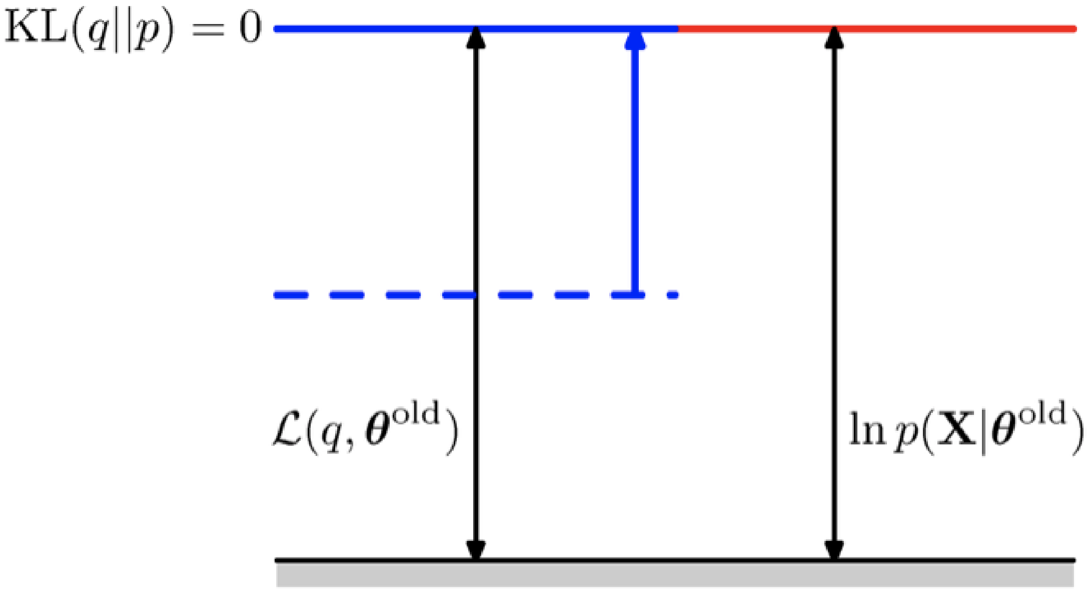
EM Algorithm

X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
$q()$: Probability distribution of Z
θ: Parameter

- **Maximum likelyhood estimation**

$$\ln p(\mathbf{X}|\theta) = L(q,\theta) + KL(q\|p)$$

- **E-step**



$\mathrm{KL}(q\|p) = 0$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{old}})$

http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

EM Algorithm

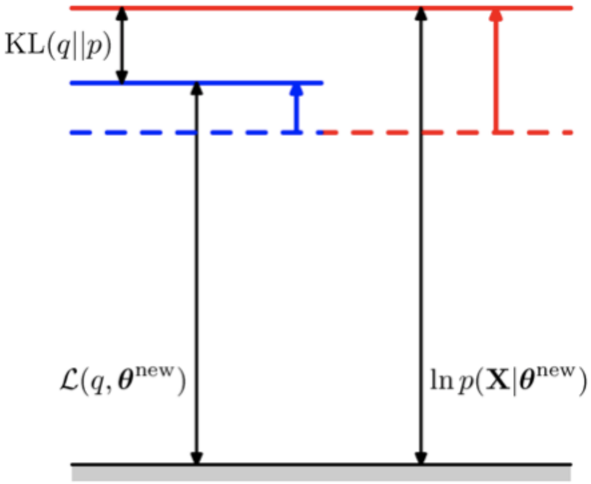
X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
$q()$: Probability distribution of Z
θ: Parameter

- **Maximum likelyhood estimation**

$$\ln p(\mathbf{X}|\theta) = L(q,\theta) + KL(q\|p)$$

$$L(q,\theta) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{old}) \ln p(\mathbf{X},\mathbf{Z}|\theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{old}) \ln p(\mathbf{Z}|\mathbf{X},\theta^{old}) = Q(\theta,\theta^{old}) + const$$

- **M-step**


KL(q||p)
$\mathcal{L}(q,\boldsymbol{\theta}^{\text{new}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{new}})$

http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

EM Algorithm

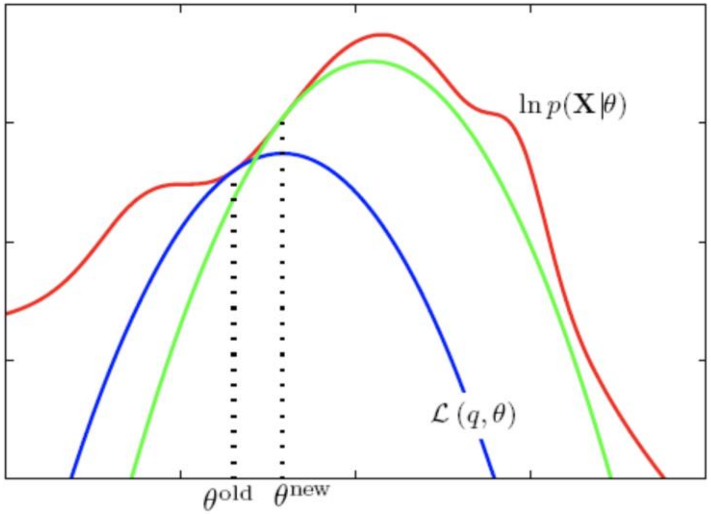
X: Observed data tensor
Z: Hidden data tensor
$p()$: Probability distribution
$q()$: Probability distribution of Z
θ: Parameter

- **Maximum likelyhood estimation**

$$\ln p(\mathbf{X}|\theta) = L(q,\theta) + KL(q\|p)$$

$$L(q,\theta) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{old}) \ln p(\mathbf{X},\mathbf{Z}|\theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{old}) \ln p(\mathbf{Z}|\mathbf{X},\theta^{old}) = Q(\theta,\theta^{old}) + const$$

- **M-step**

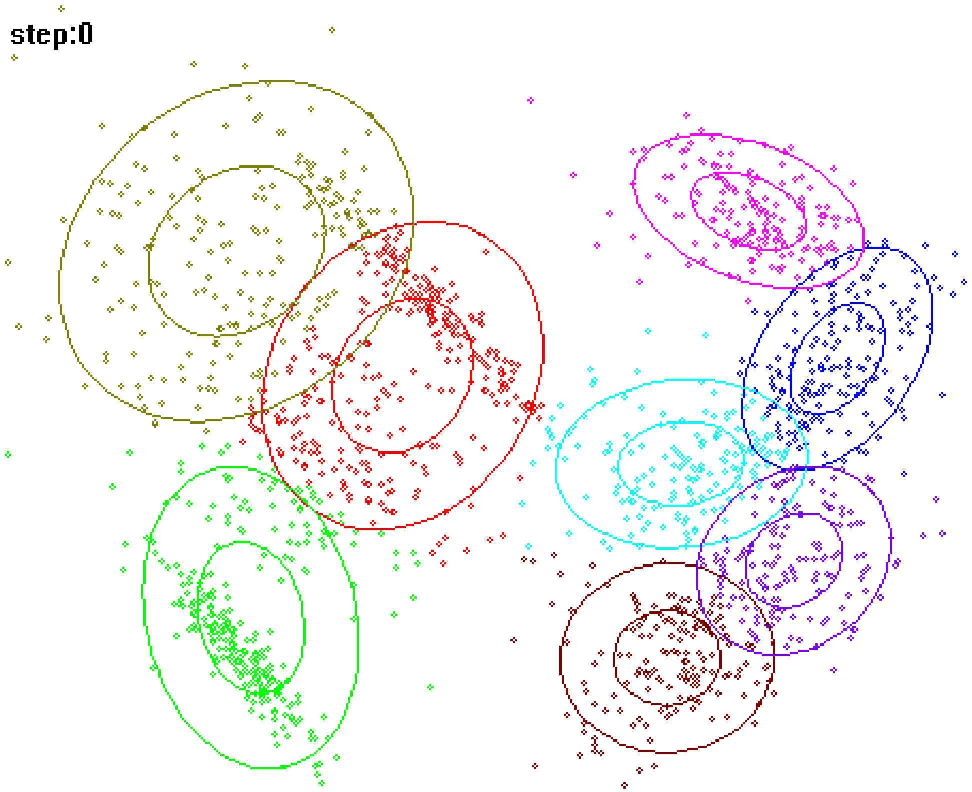


http://norman3.github.io/prml/docs/chapter09/4

http://sanghyukchun.github.io/70/

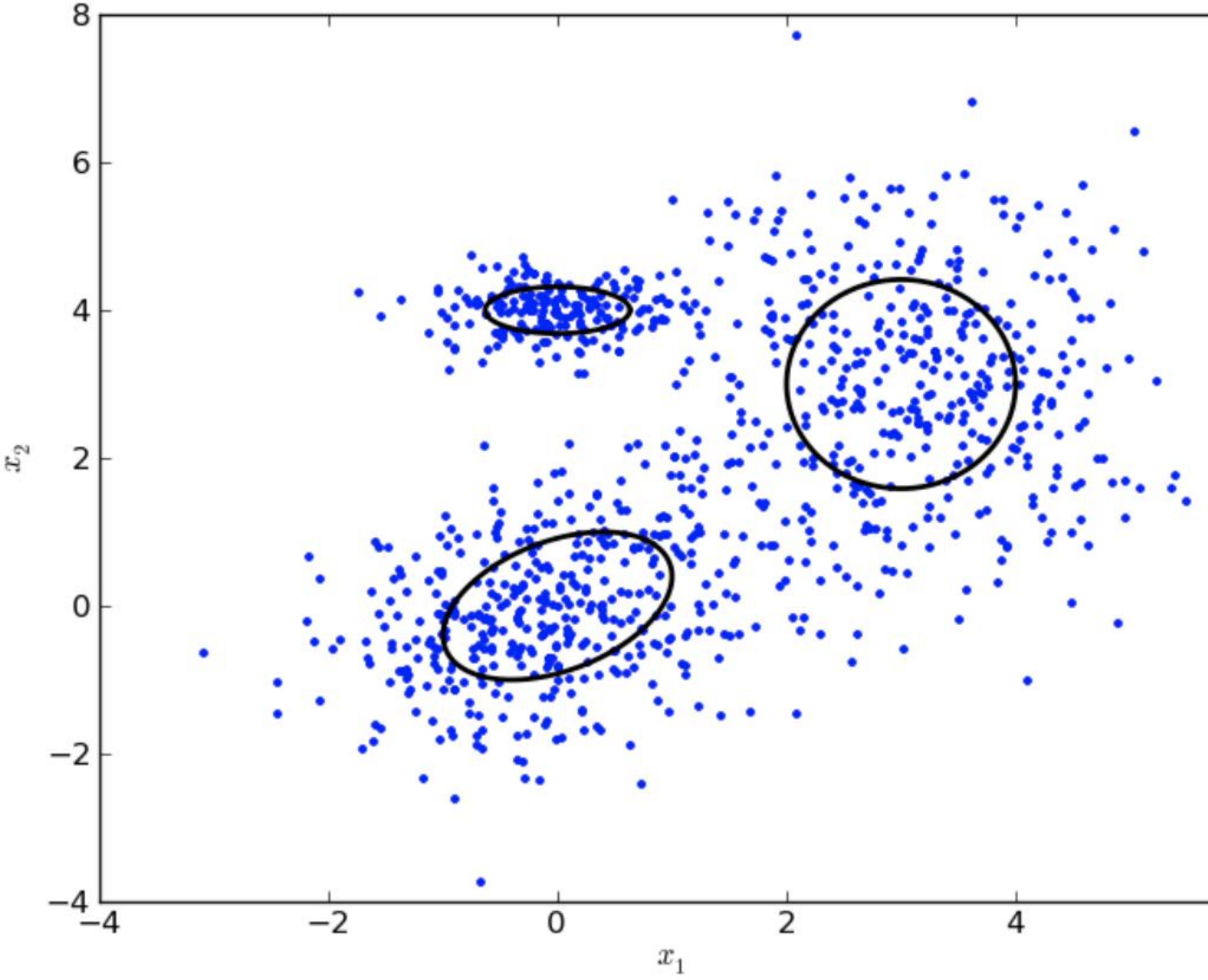
GMM Result

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{i}^{n} \ln \sum_{j}^{k} \pi_j N(x_i|\mu_j, \Sigma_j)$$



http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

Example



http://norman3.github.io/prml/docs/chapter09/4 http://sanghyukchun.github.io/70/

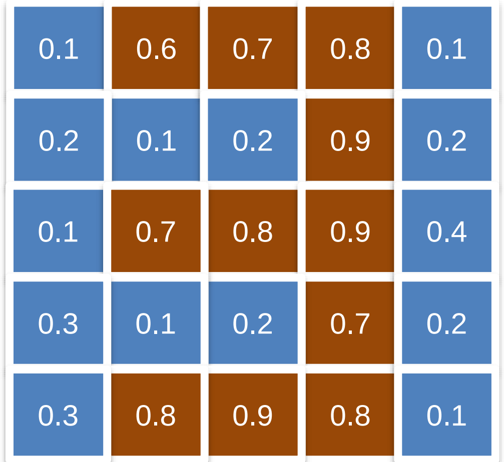
Contents

- Unsupervised pre-training
- Introduction of supervised(SL) and unsupervised learning(UL)
- Representation
- Clustering
- K-means
- Gaussian Mixture Model
- EM algorithm
- Simple representation learning implementation

Simplification MNIST

🟦 Input value 🟫 Larger value

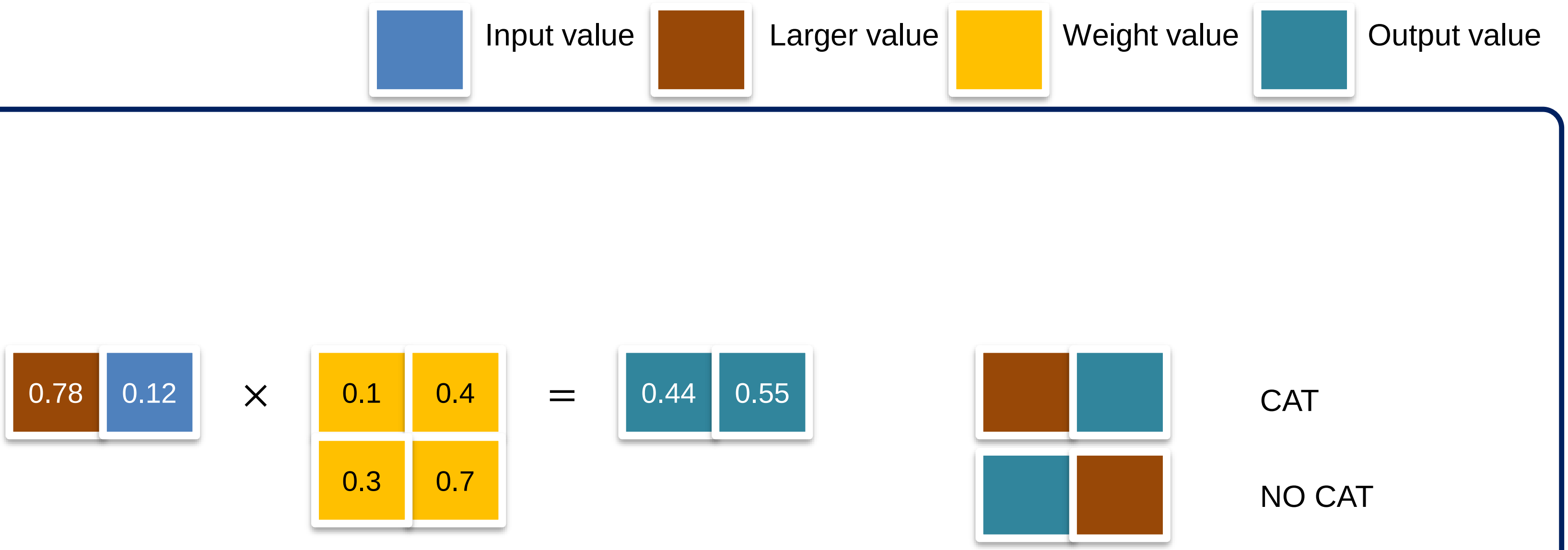
MNIST



0.1	0.6	0.7	0.8	0.1
0.2	0.1	0.2	0.9	0.2
0.1	0.7	0.8	0.9	0.4
0.3	0.1	0.2	0.7	0.2
0.3	0.8	0.9	0.8	0.1

3

Simplification



🟦 Input value 🟫 Larger value 🟨 Weight value 🟩 Output value

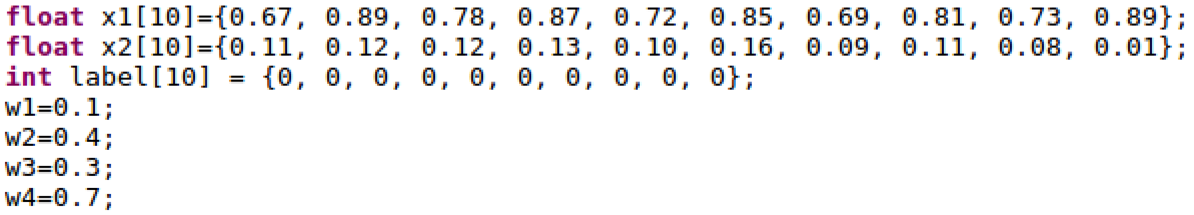
$$\begin{bmatrix} 0.78 & 0.12 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.4 \\ 0.3 & 0.7 \end{bmatrix} = \begin{bmatrix} 0.44 & 0.55 \end{bmatrix}$$

🟫🟩 CAT

🟩🟫 NO CAT

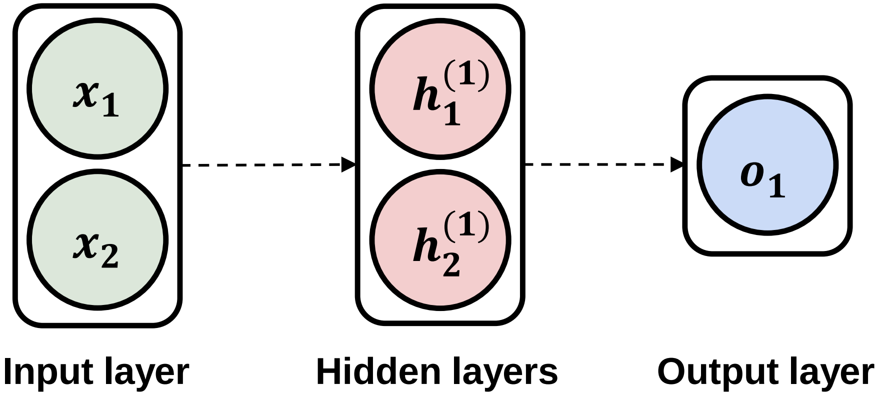
Example

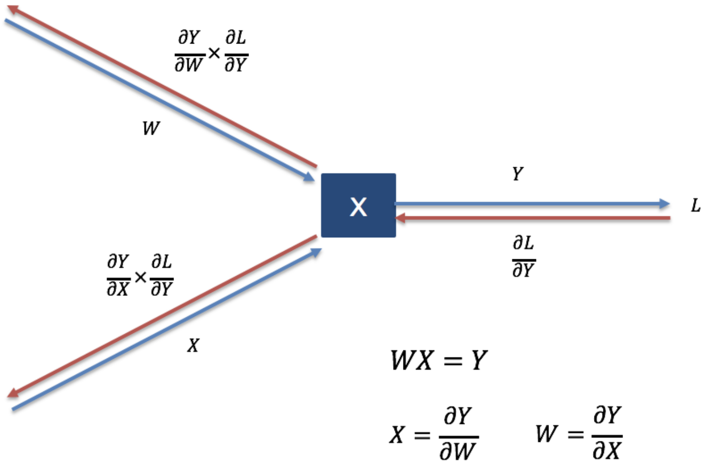
- **Data**



```
float x1[10]={0.67, 0.89, 0.78, 0.87, 0.72, 0.85, 0.69, 0.81, 0.73, 0.89};
float x2[10]={0.11, 0.12, 0.12, 0.13, 0.10, 0.16, 0.09, 0.11, 0.08, 0.01};
int label[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
w1=0.1;
w2=0.4;
w3=0.3;
w4=0.7;
```

- **Logic**

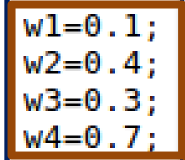

x_1
x_2
$h_1^{(1)}$
$h_2^{(1)}$
o_1
Input layer
Hidden layers
Output layer


$\frac{\partial Y}{\partial W} \times \frac{\partial L}{\partial Y}$
W
Y
x
L
$\frac{\partial L}{\partial Y}$
$\frac{\partial Y}{\partial X} \times \frac{\partial L}{\partial Y}$
X

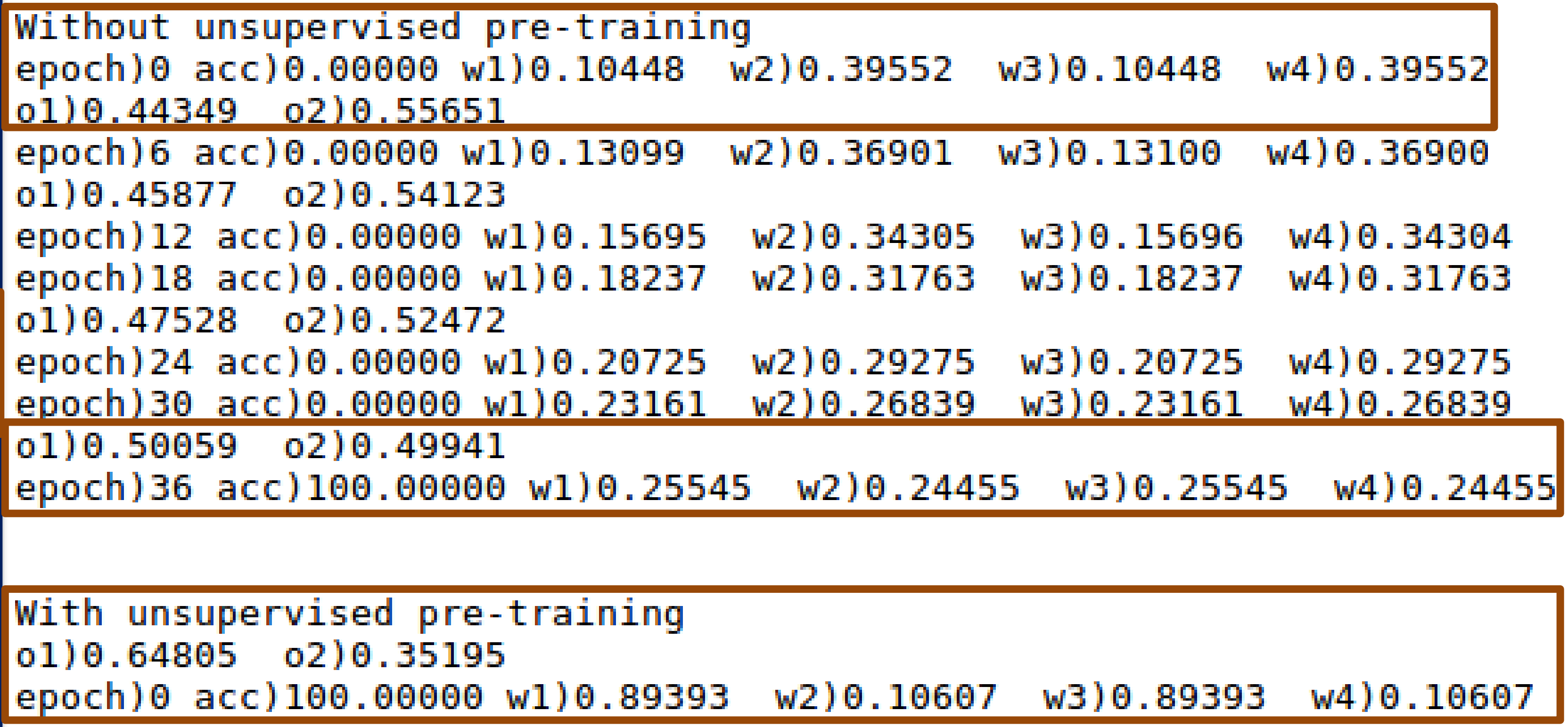
$WX = Y$

$X = \frac{\partial Y}{\partial W} \qquad W = \frac{\partial Y}{\partial X}$

Result

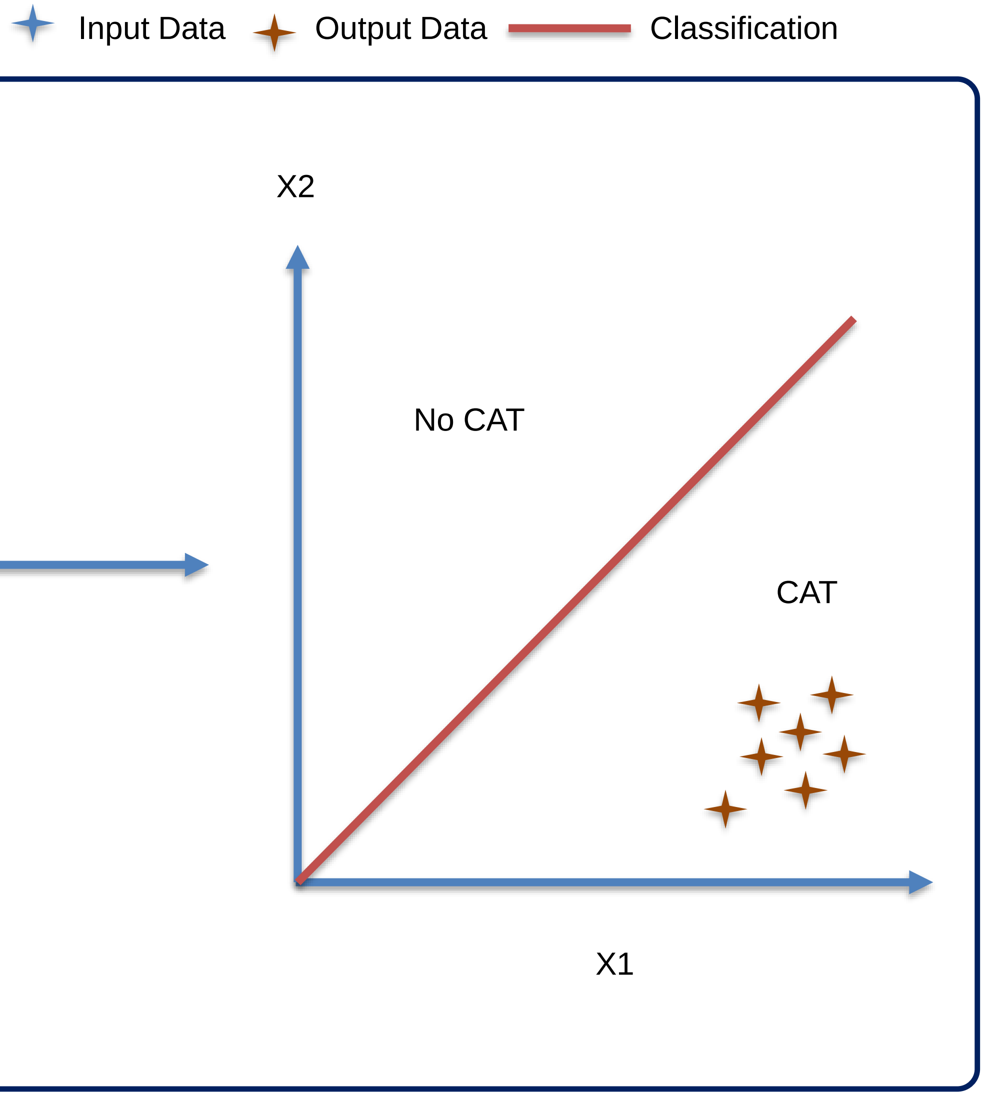


```
w1=0.1;
w2=0.4;
w3=0.3;
w4=0.7;
```



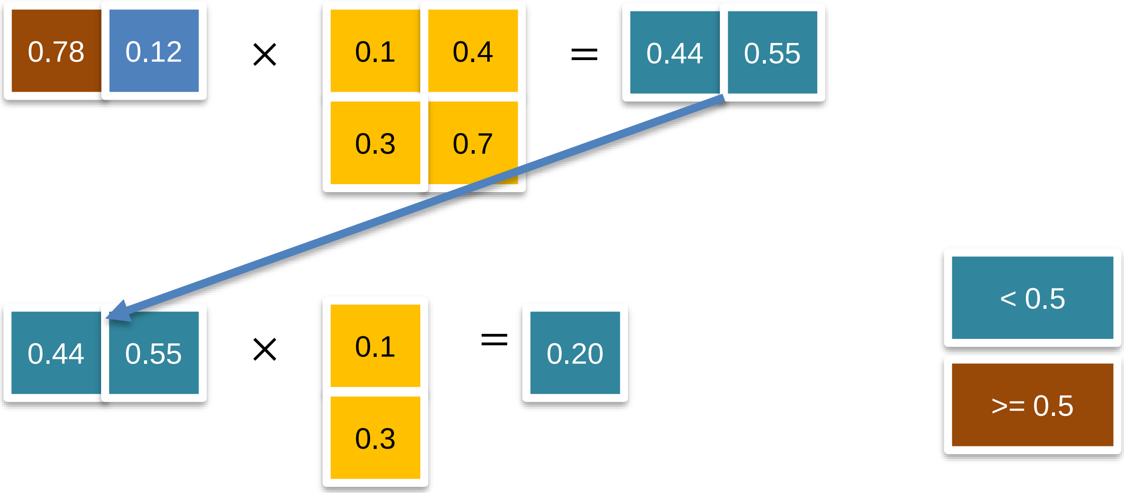
```
Without unsupervised pre-training
epoch)0 acc)0.00000 w1)0.10448  w2)0.39552  w3)0.10448  w4)0.39552
o1)0.44349  o2)0.55651
epoch)6 acc)0.00000 w1)0.13099  w2)0.36901  w3)0.13100  w4)0.36900
o1)0.45877  o2)0.54123
epoch)12 acc)0.00000 w1)0.15695  w2)0.34305  w3)0.15696  w4)0.34304
epoch)18 acc)0.00000 w1)0.18237  w2)0.31763  w3)0.18237  w4)0.31763
o1)0.47528  o2)0.52472
epoch)24 acc)0.00000 w1)0.20725  w2)0.29275  w3)0.20725  w4)0.29275
epoch)30 acc)0.00000 w1)0.23161  w2)0.26839  w3)0.23161  w4)0.26839
o1)0.50059  o2)0.49941
epoch)36 acc)100.00000 w1)0.25545  w2)0.24455  w3)0.25545  w4)0.24455


With unsupervised pre-training
o1)0.64805  o2)0.35195
epoch)0 acc)100.00000 w1)0.89393  w2)0.10607  w3)0.89393  w4)0.10607
```

Representation graph


Input Data
Output Data
Classification
X2
X1
X2
No CAT
CAT
X1

Simplification

🟦 Input value 🟫 Larger value 🟨 Weight value 🟩 Output value



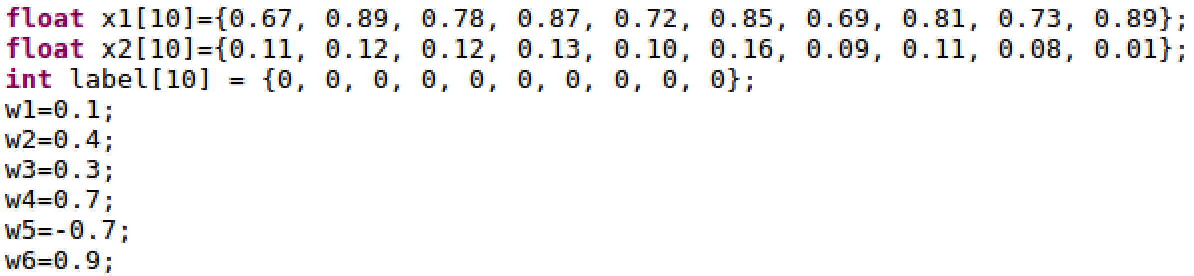
$$\begin{bmatrix} 0.78 & 0.12 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.4 \\ 0.3 & 0.7 \end{bmatrix} = \begin{bmatrix} 0.44 & 0.55 \end{bmatrix}$$

$$\begin{bmatrix} 0.44 & 0.55 \end{bmatrix} \times \begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.20 \end{bmatrix}$$

< 0.5	CAT
>= 0.5	NO CAT

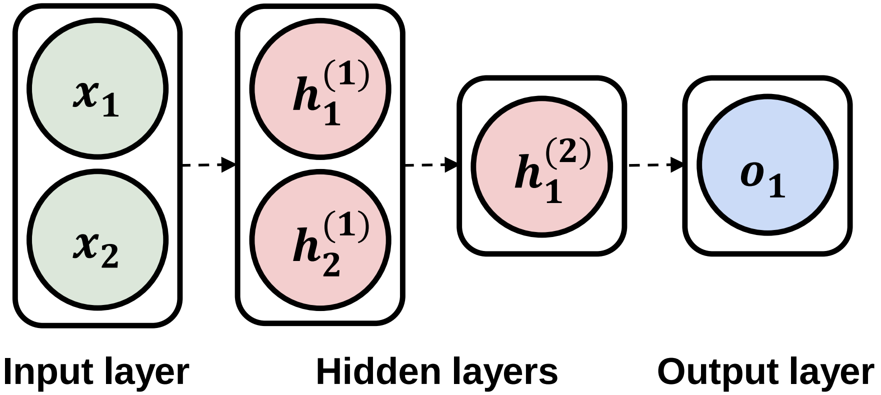
Example

Data

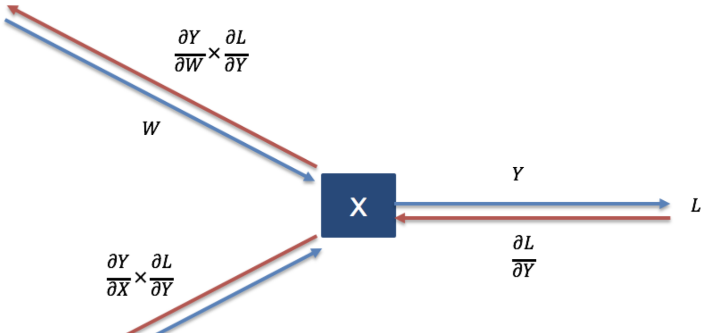


```
float x1[10]={0.67, 0.89, 0.78, 0.87, 0.72, 0.85, 0.69, 0.81, 0.73, 0.89};
float x2[10]={0.11, 0.12, 0.12, 0.13, 0.10, 0.16, 0.09, 0.11, 0.08, 0.01};
int label[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
w1=0.1;
w2=0.4;
w3=0.3;
w4=0.7;
w5=-0.7;
w6=0.9;
```

Logic



x_1
x_2
$h_1^{(1)}$
$h_2^{(1)}$
$h_1^{(2)}$
o_1
Input layer
Hidden layers
Output layer



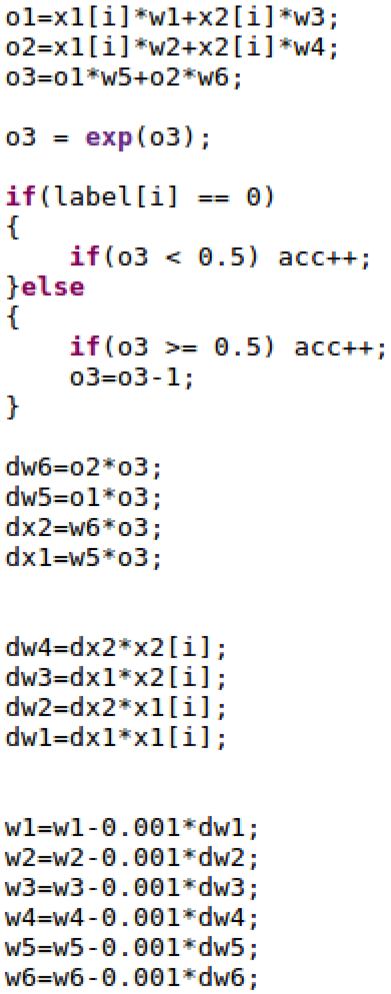
$\frac{\partial Y}{\partial W}\times\frac{\partial L}{\partial Y}$
W
X
Y
L
$\frac{\partial L}{\partial Y}$
$\frac{\partial Y}{\partial X}\times\frac{\partial L}{\partial Y}$
X

$$WX = Y$$

$$X = \frac{\partial Y}{\partial W} \qquad W = \frac{\partial Y}{\partial X}$$

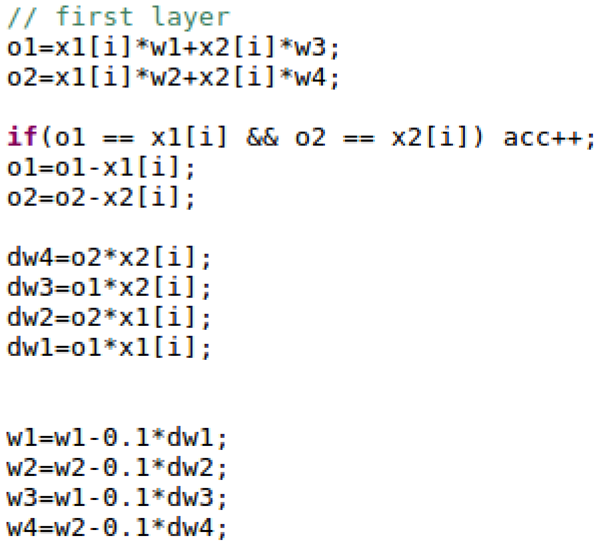
Code

Supervised learning

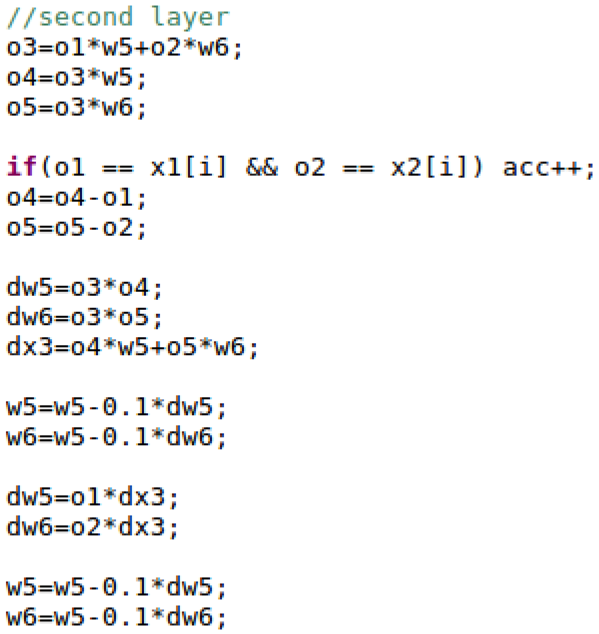


```
o1=x1[i]*w1+x2[i]*w3;
o2=x1[i]*w2+x2[i]*w4;
o3=o1*w5+o2*w6;

o3 = exp(o3);

if(label[i] == 0)
{
    if(o3 < 0.5) acc++;
}else
{
    if(o3 >= 0.5) acc++;
    o3=o3-1;
}

dw6=o2*o3;
dw5=o1*o3;
dx2=w6*o3;
dx1=w5*o3;


dw4=dx2*x2[i];
dw3=dx1*x2[i];
dw2=dx2*x1[i];
dw1=dx1*x1[i];


w1=w1-0.001*dw1;
w2=w2-0.001*dw2;
w3=w3-0.001*dw3;
w4=w4-0.001*dw4;
w5=w5-0.001*dw5;
w6=w6-0.001*dw6;
```

Unsupervised learning



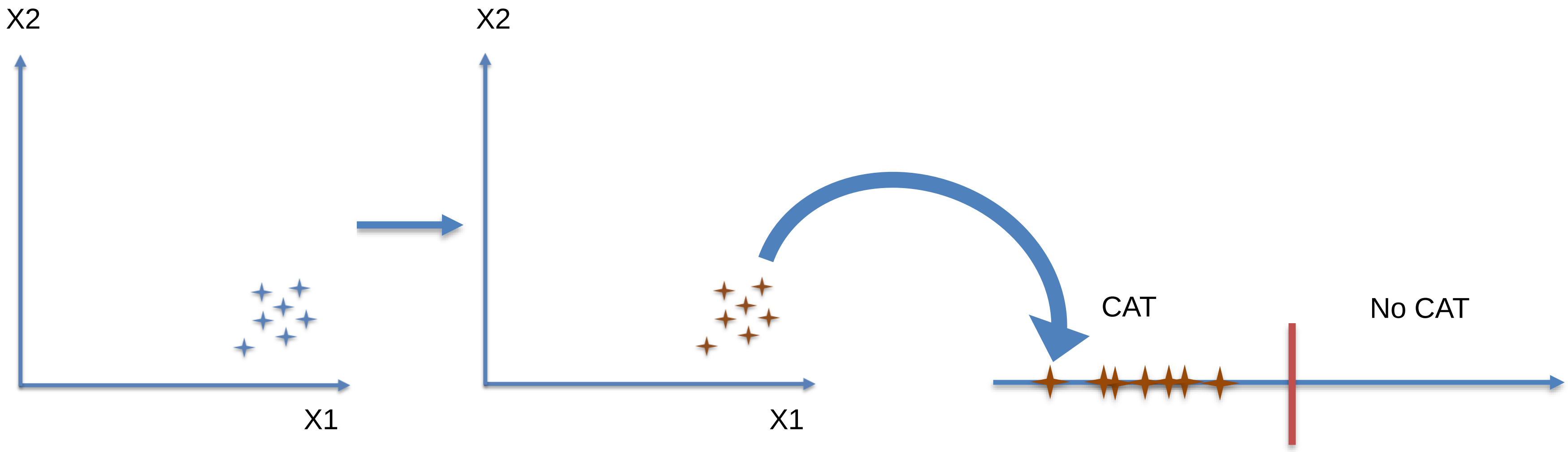
```
// first layer
o1=x1[i]*w1+x2[i]*w3;
o2=x1[i]*w2+x2[i]*w4;

if(o1 == x1[i] && o2 == x2[i]) acc++;
o1=o1-x1[i];
o2=o2-x2[i];

dw4=o2*x2[i];
dw3=o1*x2[i];
dw2=o2*x1[i];
dw1=o1*x1[i];


w1=w1-0.1*dw1;
w2=w2-0.1*dw2;
w3=w1-0.1*dw3;
w4=w2-0.1*dw4;
```



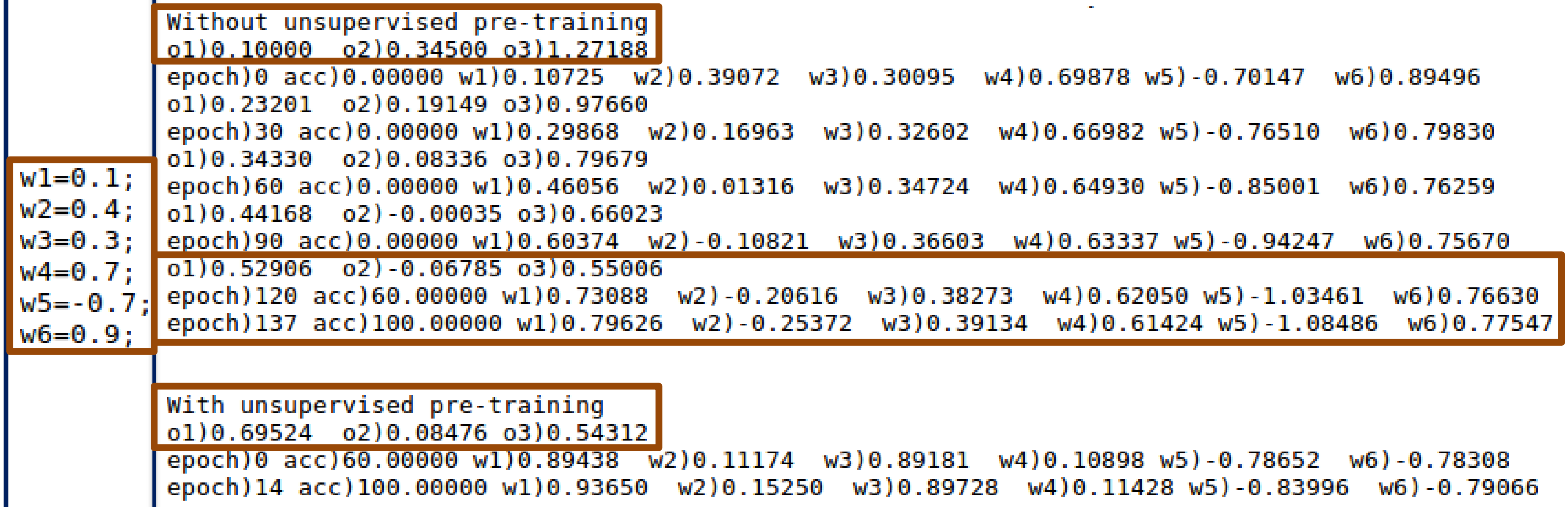
```
//second layer
o3=o1*w5+o2*w6;
o4=o3*w5;
o5=o3*w6;

if(o1 == x1[i] && o2 == x2[i]) acc++;
o4=o4-o1;
o5=o5-o2;

dw5=o3*o4;
dw6=o3*o5;
dx3=o4*w5+o5*w6;

w5=w5-0.1*dw5;
w6=w5-0.1*dw6;

dw5=o1*dx3;
dw6=o2*dx3;

w5=w5-0.1*dw5;
w6=w5-0.1*dw6;
```

Representation graph

Input Data
Output Data
Classification


X2
X1
X2
X1
CAT
No CAT

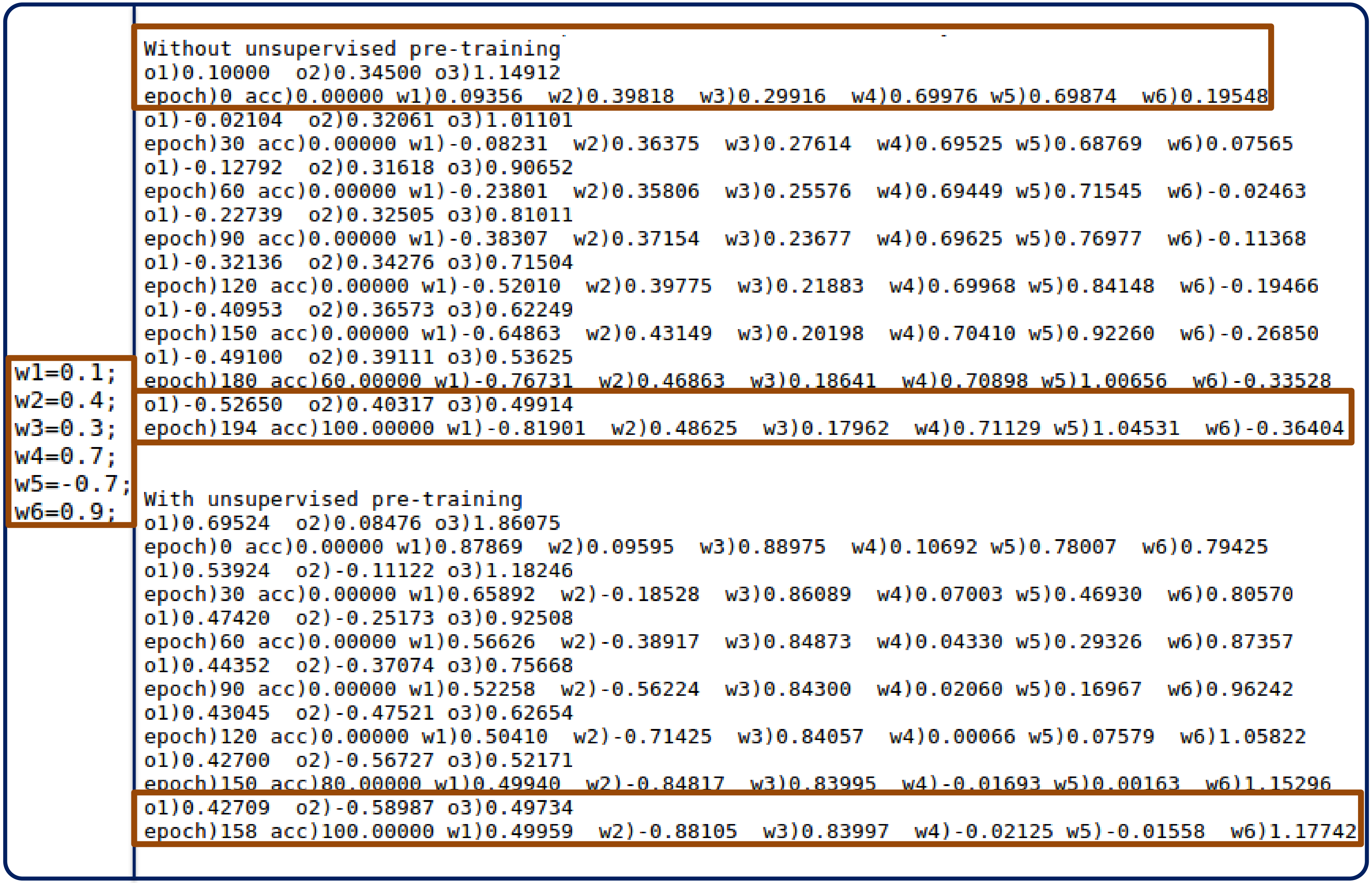
Result



```
w1=0.1;
w2=0.4;
w3=0.3;
w4=0.7;
w5=-0.7;
w6=0.9;
```

```
Without unsupervised pre-training
o1)0.10000  o2)0.34500 o3)1.27188
epoch)0 acc)0.00000 w1)0.10725  w2)0.39072  w3)0.30095  w4)0.69878 w5)-0.70147  w6)0.89496
o1)0.23201  o2)0.19149 o3)0.97660
epoch)30 acc)0.00000 w1)0.29868  w2)0.16963  w3)0.32602  w4)0.66982 w5)-0.76510  w6)0.79830
o1)0.34330  o2)0.08336 o3)0.79679
epoch)60 acc)0.00000 w1)0.46056  w2)0.01316  w3)0.34724  w4)0.64930 w5)-0.85001  w6)0.76259
o1)0.44168  o2)-0.00035 o3)0.66023
epoch)90 acc)0.00000 w1)0.60374  w2)-0.10821  w3)0.36603  w4)0.63337 w5)-0.94247  w6)0.75670
o1)0.52906  o2)-0.06785 o3)0.55006
epoch)120 acc)60.00000 w1)0.73088  w2)-0.20616  w3)0.38273  w4)0.62050 w5)-1.03461  w6)0.76630
epoch)137 acc)100.00000 w1)0.79626  w2)-0.25372  w3)0.39134  w4)0.61424 w5)-1.08486  w6)0.77547

With unsupervised pre-training
o1)0.69524  o2)0.08476 o3)0.54312
epoch)0 acc)60.00000 w1)0.89438  w2)0.11174  w3)0.89181  w4)0.10898 w5)-0.78652  w6)-0.78308
epoch)14 acc)100.00000 w1)0.93650  w2)0.15250  w3)0.89728  w4)0.11428 w5)-0.83996  w6)-0.79066
```

Result



```
w1=0.1;
w2=0.4;
w3=0.3;
w4=0.7;
w5=-0.7;
w6=0.9;
```

```
Without unsupervised pre-training
o1)0.10000  o2)0.34500 o3)1.14912
epoch)0 acc)0.00000 w1)0.09356  w2)0.39818  w3)0.29916  w4)0.69976 w5)0.69874  w6)0.19548
o1)-0.02104  o2)0.32061 o3)1.01101
epoch)30 acc)0.00000 w1)-0.08231  w2)0.36375  w3)0.27614  w4)0.69525 w5)0.68769  w6)0.07565
o1)-0.12792  o2)0.31618 o3)0.90652
epoch)60 acc)0.00000 w1)-0.23801  w2)0.35806  w3)0.25576  w4)0.69449 w5)0.71545  w6)-0.02463
o1)-0.22739  o2)0.32505 o3)0.81011
epoch)90 acc)0.00000 w1)-0.38307  w2)0.37154  w3)0.23677  w4)0.69625 w5)0.76977  w6)-0.11368
o1)-0.32136  o2)0.34276 o3)0.71504
epoch)120 acc)0.00000 w1)-0.52010  w2)0.39775  w3)0.21883  w4)0.69968 w5)0.84148  w6)-0.19466
o1)-0.40953  o2)0.36573 o3)0.62249
epoch)150 acc)0.00000 w1)-0.64863  w2)0.43149  w3)0.20198  w4)0.70410 w5)0.92260  w6)-0.26850
o1)-0.49100  o2)0.39111 o3)0.53625
epoch)180 acc)60.00000 w1)-0.76731  w2)0.46863  w3)0.18641  w4)0.70898 w5)1.00656  w6)-0.33528
o1)-0.52650  o2)0.40317 o3)0.49914
epoch)194 acc)100.00000 w1)-0.81901  w2)0.48625  w3)0.17962  w4)0.71129 w5)1.04531  w6)-0.36404


With unsupervised pre-training
o1)0.69524  o2)0.08476 o3)1.86075
epoch)0 acc)0.00000 w1)0.87869  w2)0.09595  w3)0.88975  w4)0.10692 w5)0.78007  w6)0.79425
o1)0.53924  o2)-0.11122 o3)1.18246
epoch)30 acc)0.00000 w1)0.65892  w2)-0.18528  w3)0.86089  w4)0.07003 w5)0.46930  w6)0.80570
o1)0.47420  o2)-0.25173 o3)0.92508
epoch)60 acc)0.00000 w1)0.56626  w2)-0.38917  w3)0.84873  w4)0.04330 w5)0.29326  w6)0.87357
o1)0.44352  o2)-0.37074 o3)0.75668
epoch)90 acc)0.00000 w1)0.52258  w2)-0.56224  w3)0.84300  w4)0.02060 w5)0.16967  w6)0.96242
o1)0.43045  o2)-0.47521 o3)0.62654
epoch)120 acc)0.00000 w1)0.50410  w2)-0.71425  w3)0.84057  w4)0.00066 w5)0.07579  w6)1.05822
o1)0.42700  o2)-0.56727 o3)0.52171
epoch)150 acc)80.00000 w1)0.49940  w2)-0.84817  w3)0.83995  w4)-0.01693 w5)0.00163  w6)1.15296
o1)0.42709  o2)-0.58987 o3)0.49734
epoch)158 acc)100.00000 w1)0.49959  w2)-0.88105  w3)0.83997  w4)-0.02125 w5)-0.01558  w6)1.17742
```

Thank you