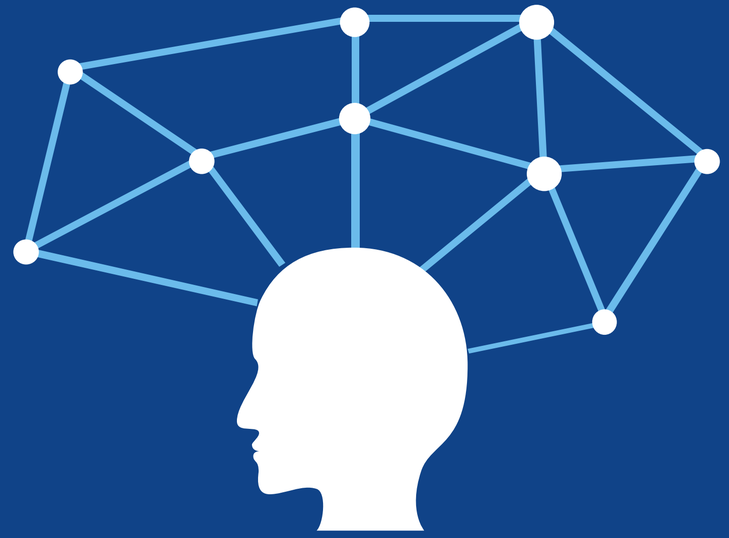
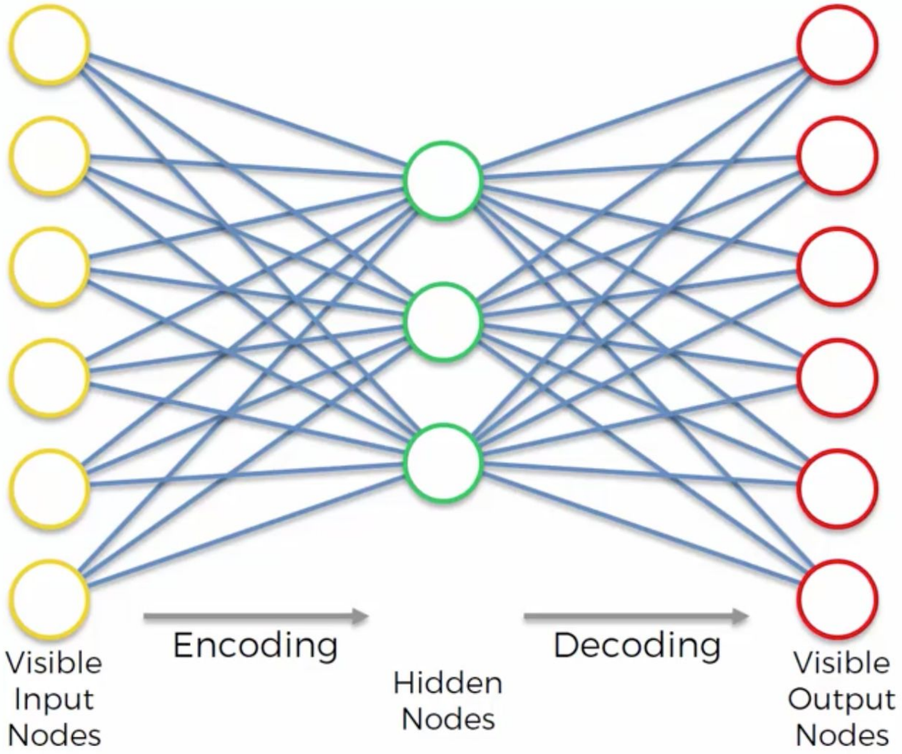
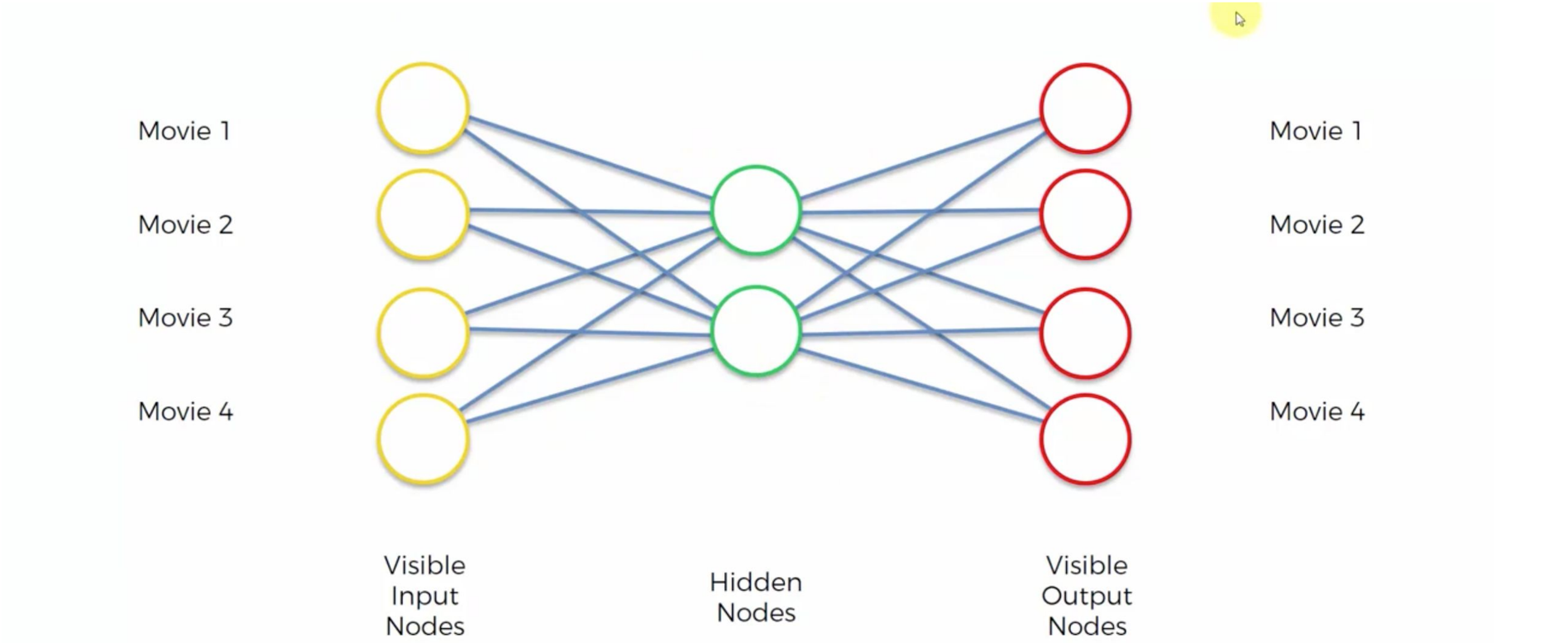
Auto Encoders



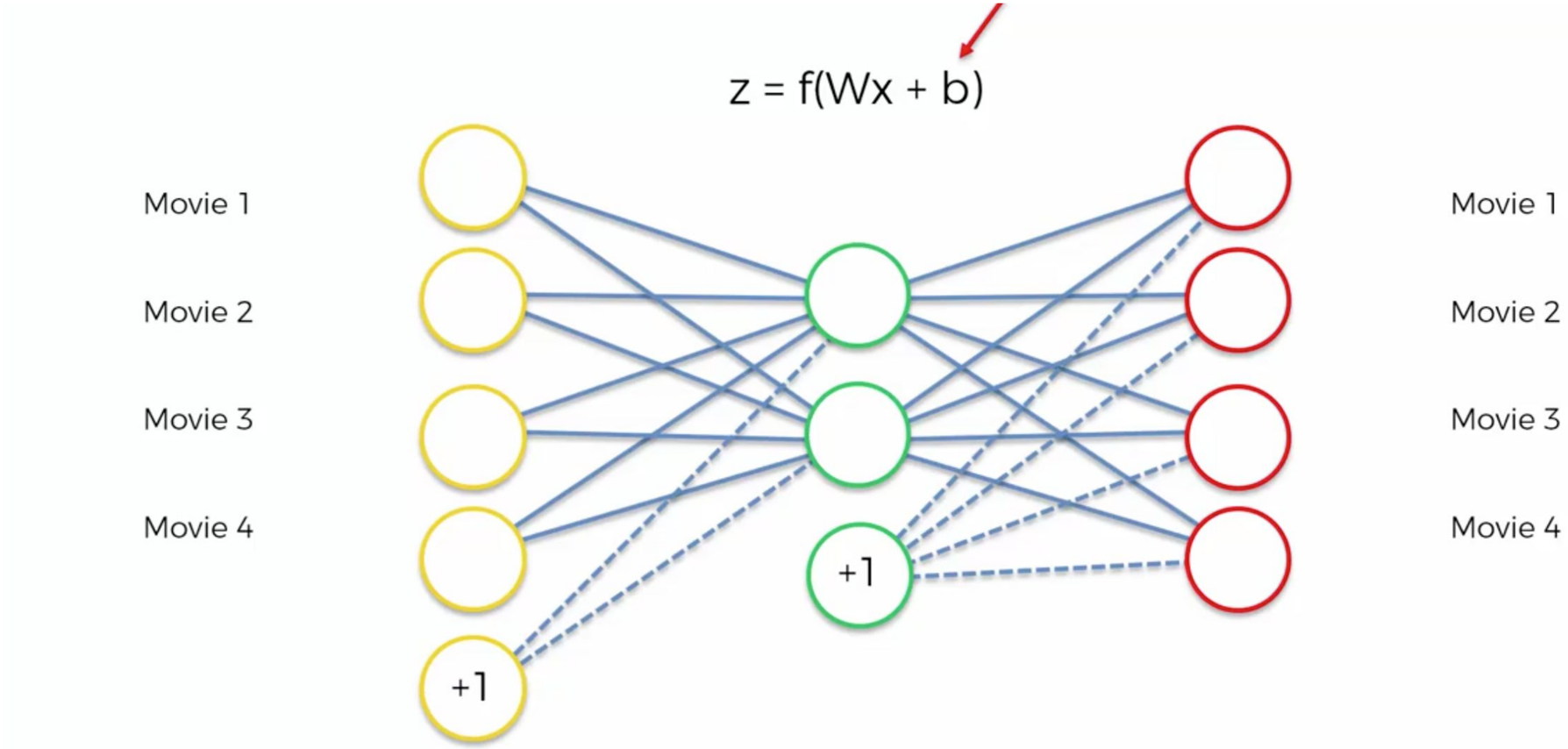
Auto Encoders



Auto Encoders



Auto Encoders - Bias



Auto Encoders - Bias

STEP 1: We start with an array where the lines (the observations) correspond to the users and the columns (the features) correspond to the movies. Each cell (u, i) contains the rating (from 1 to 5, 0 if no rating) of the movie i by the user u.

STEP 2: The first user goes into the network. The input vector x = (r_1, r_2, ..., r_m) contains all its ratings for all the movies.

STEP 3: The input vector x is encoded into a vector z of lower dimensions by a mapping function f (e.g: sigmoid function):

$$z = f(Wx + b)$$ where W is the vector of input weights and b the bias

STEP 4: z is then decoded into the output vector y of same dimensions as x, aiming to replicate the input vector x.

STEP 5: The reconstruction error $d(x, y) = ||x-y||$ is computed. The goal is to minimize it.

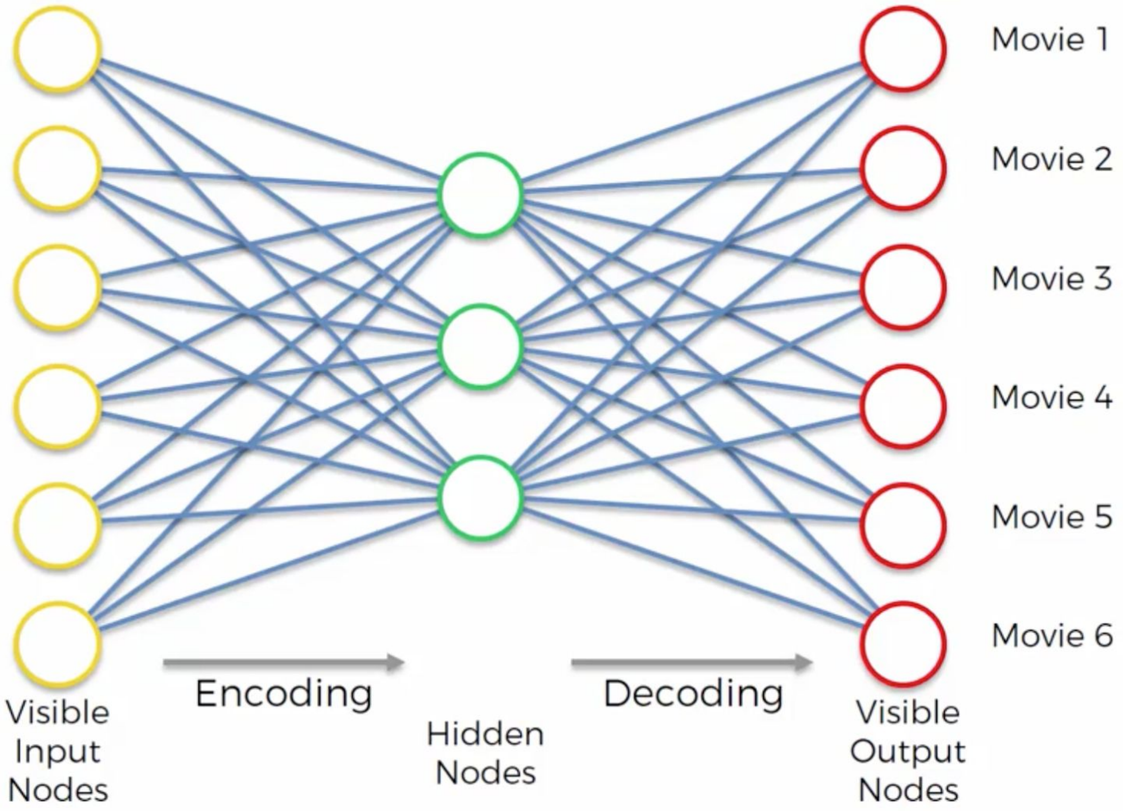
STEP 6: Back-Propagation: from right to left, the error is back-propagated. The weights are updated according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 7: Repeat Steps 1 to 6 and update the weights after each observation (Reinforcement Learning). Or:
Repeat Steps 1 to 6 but update the weights only after a batch of observations (Batch Learning).

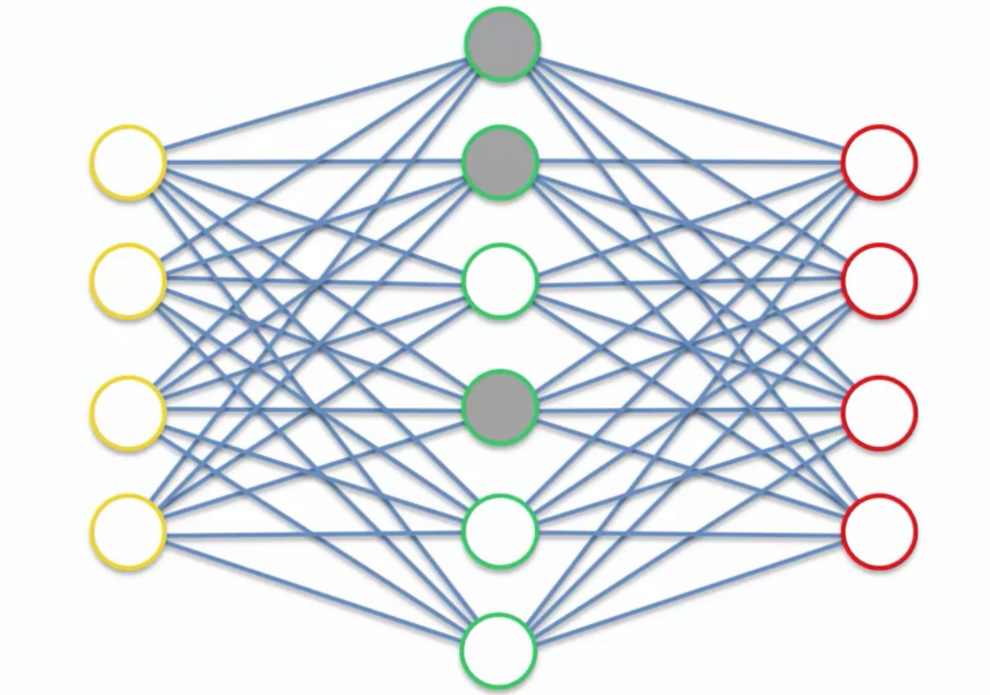
Auto Encoders - Movies Example

STEP 7

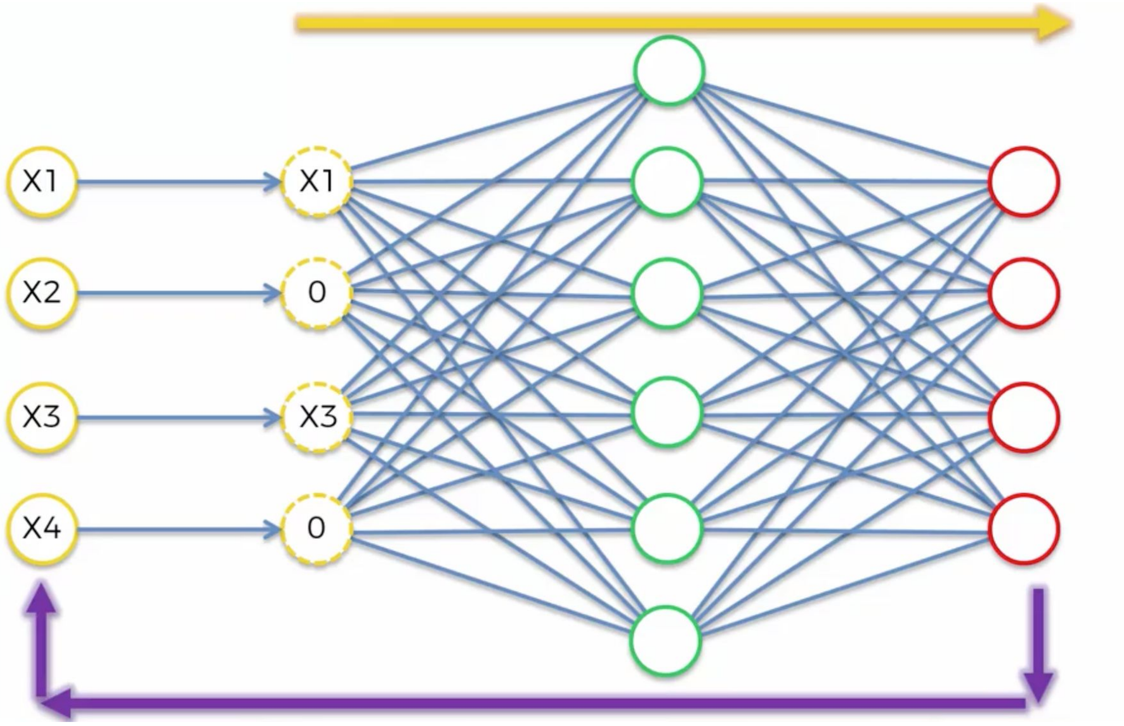
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1



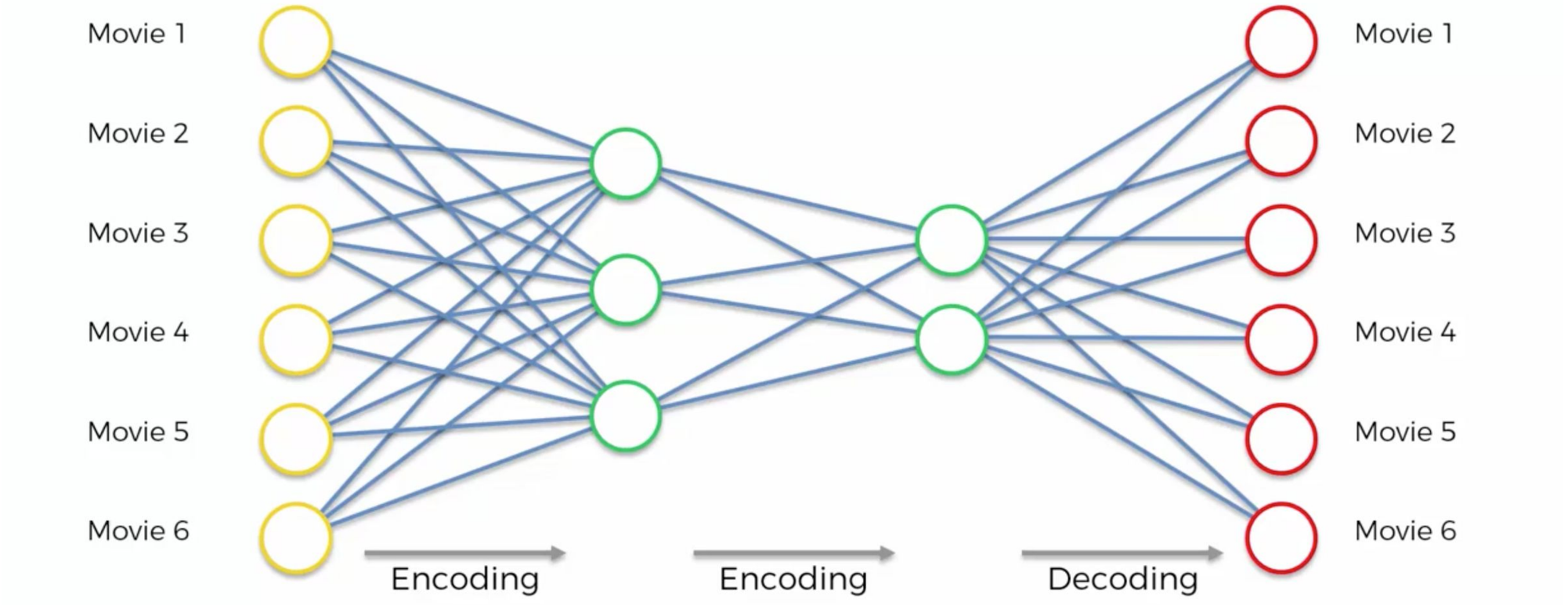
Over Complete Hidden Layers Sparse AutoEncoders



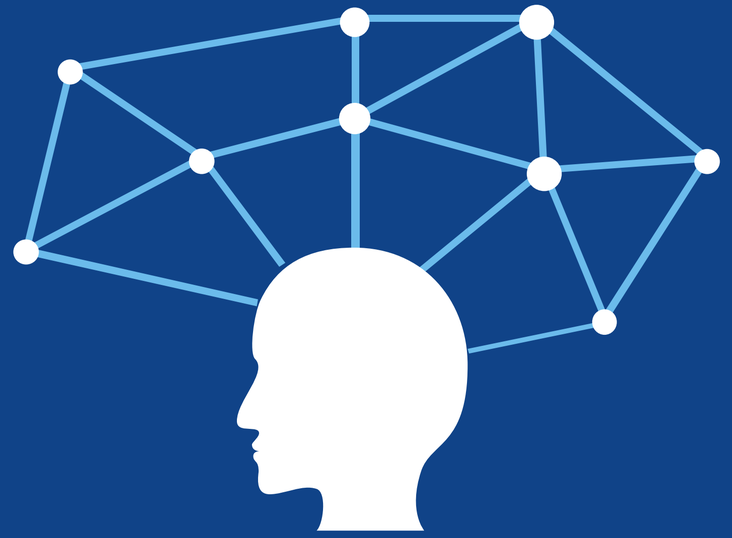
Over Complete Hidden Layers Denoising AutoEncoders



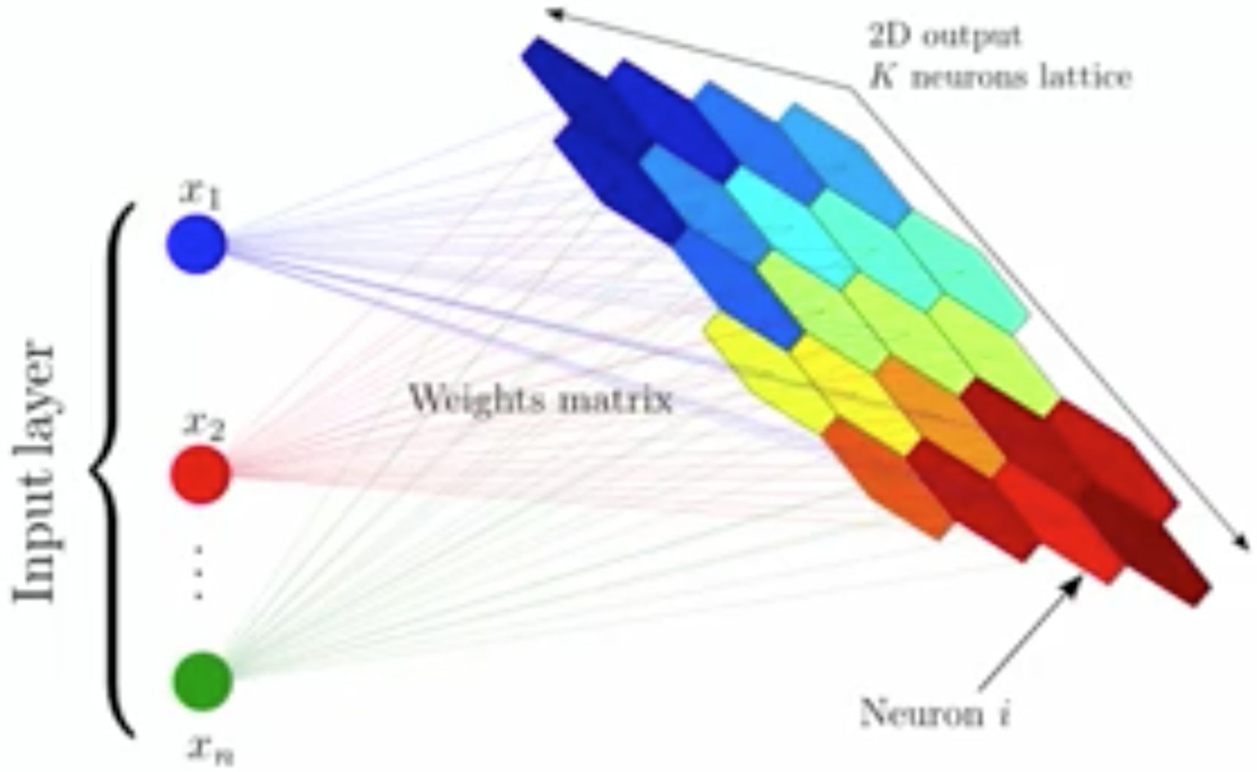
Over Complete Hidden Layers Stacked AutoEncoders



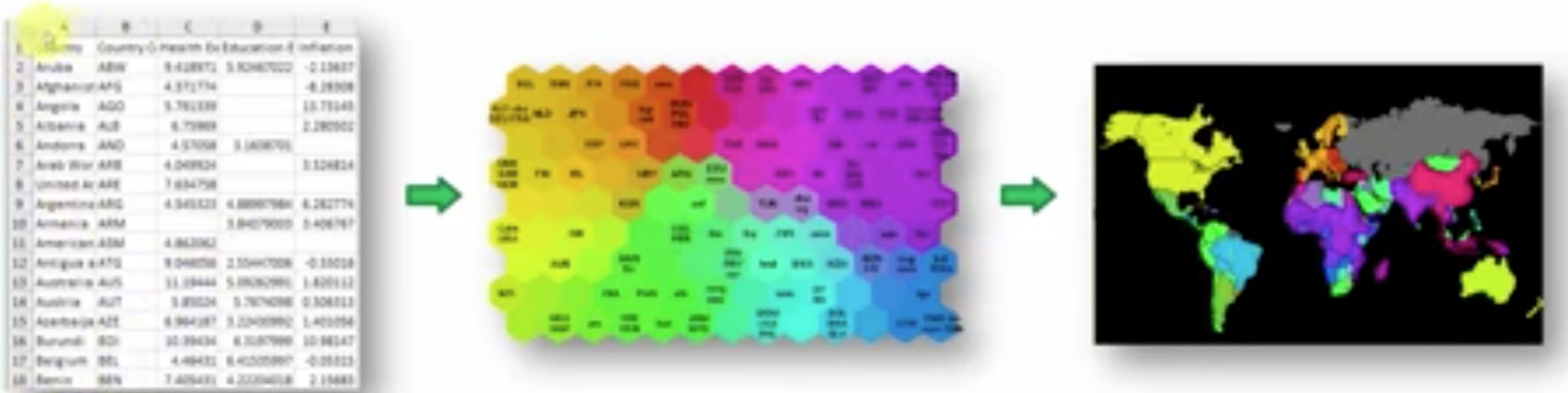
Self Organizing Maps



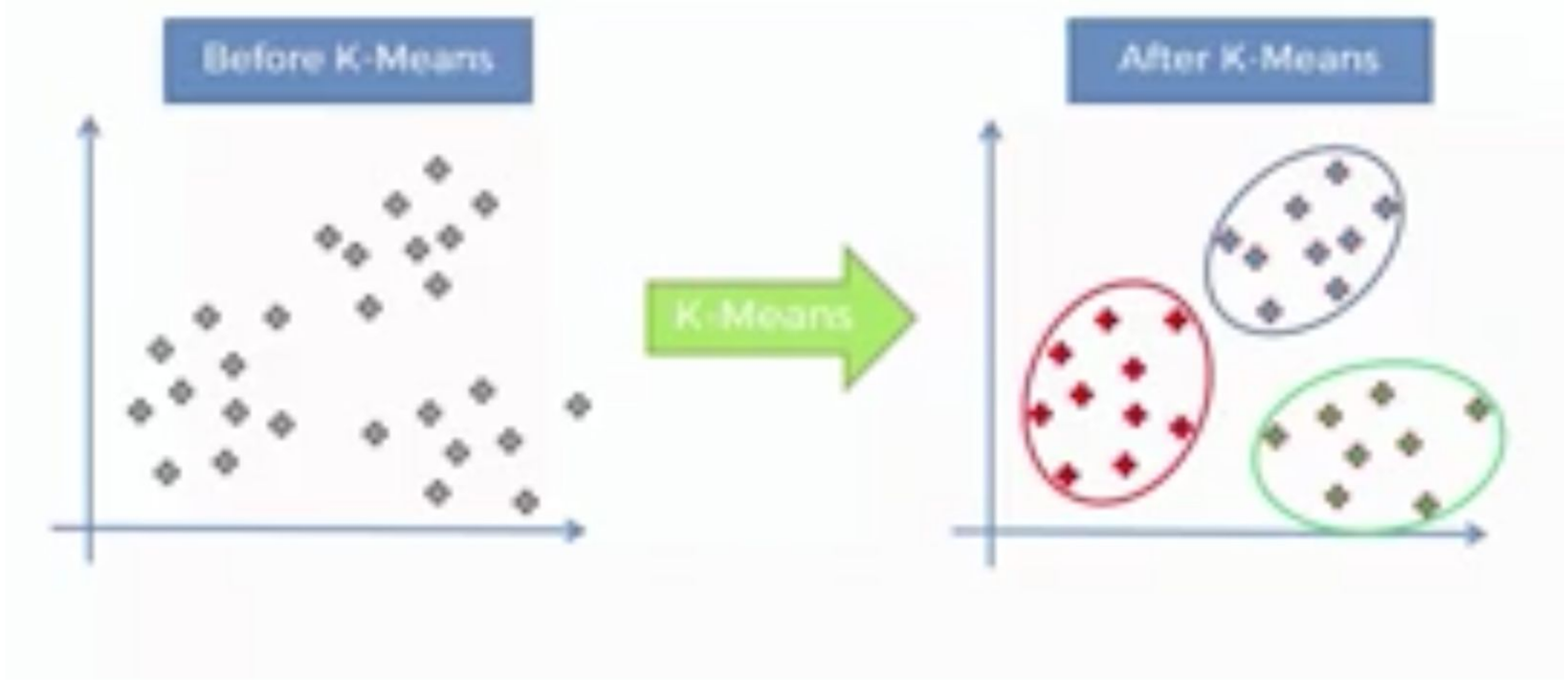
Self Organizing Maps



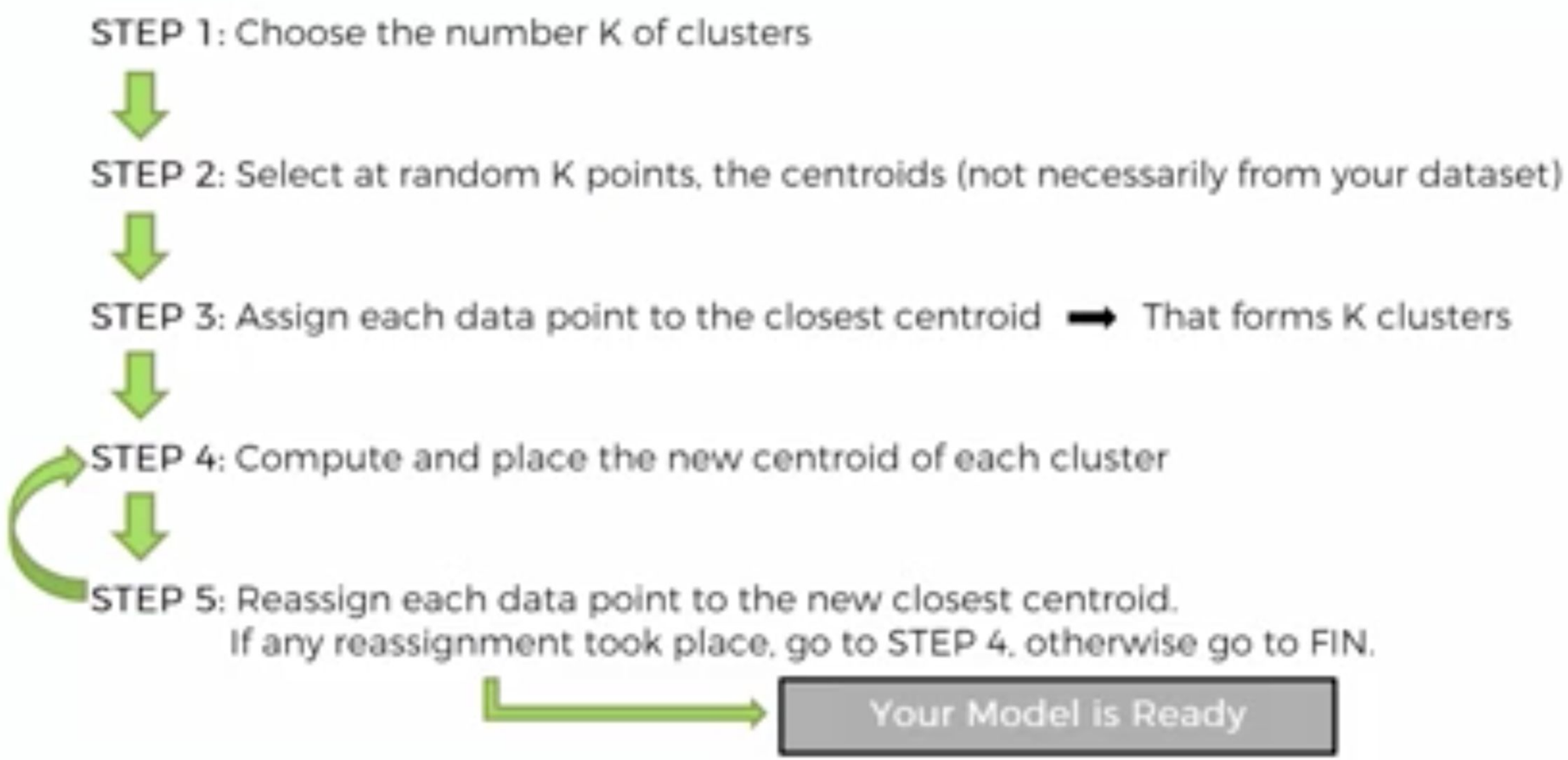
Self Organising Maps



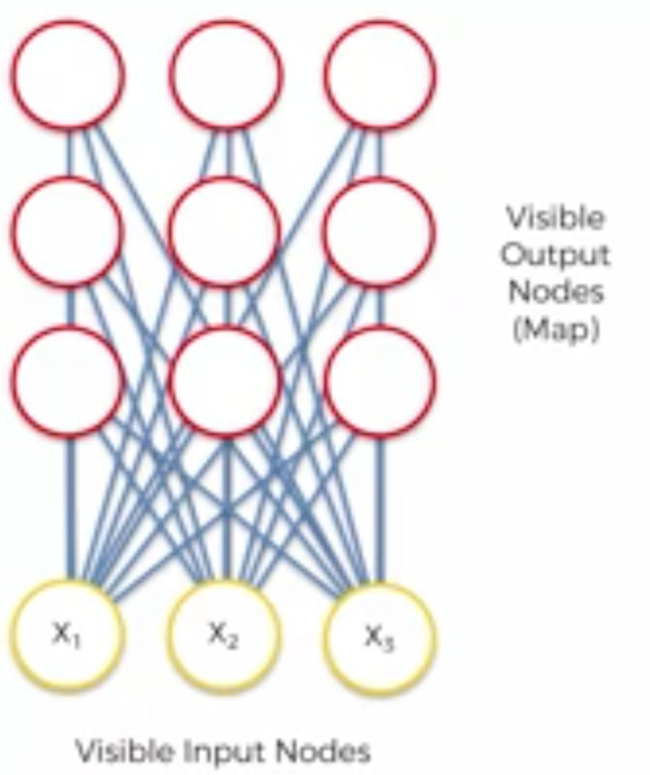
K-Means ~ Self Organising Maps

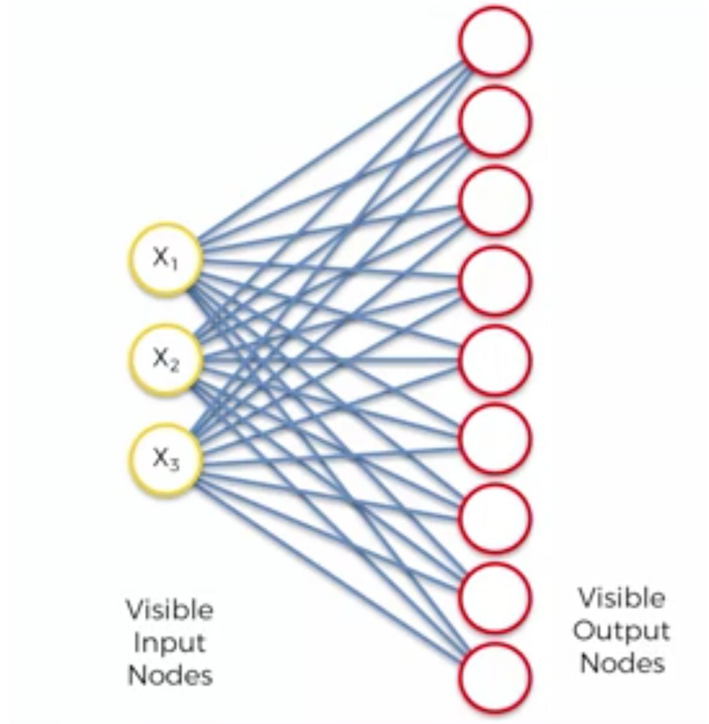


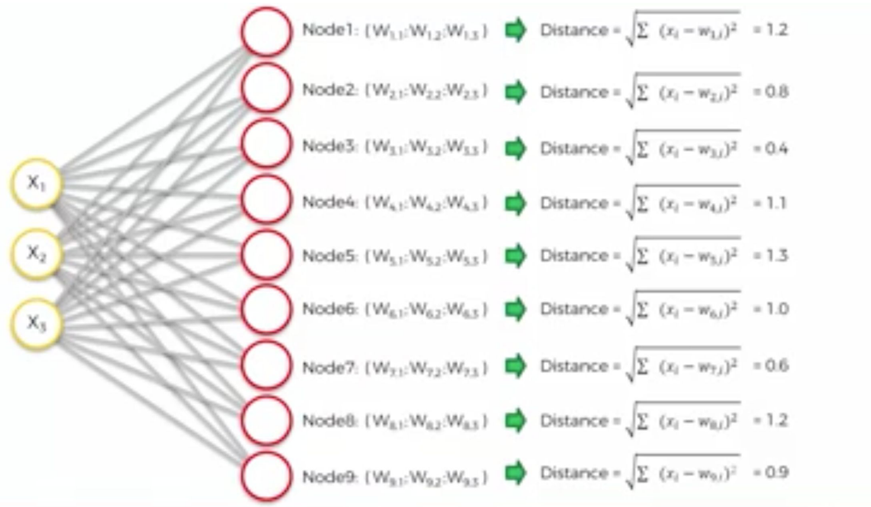
K-Means ~ Self Organising Maps



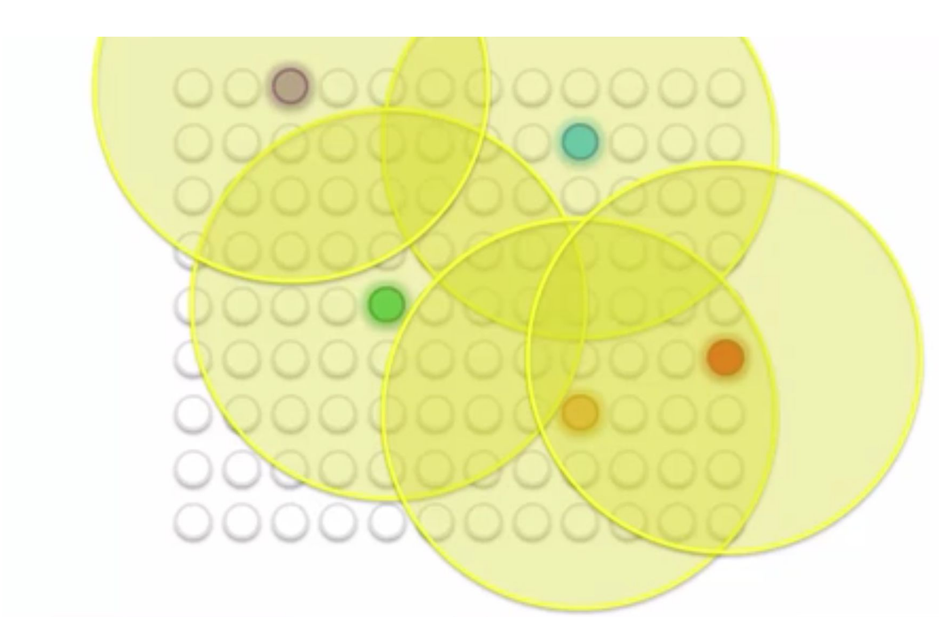
Self Organising Maps

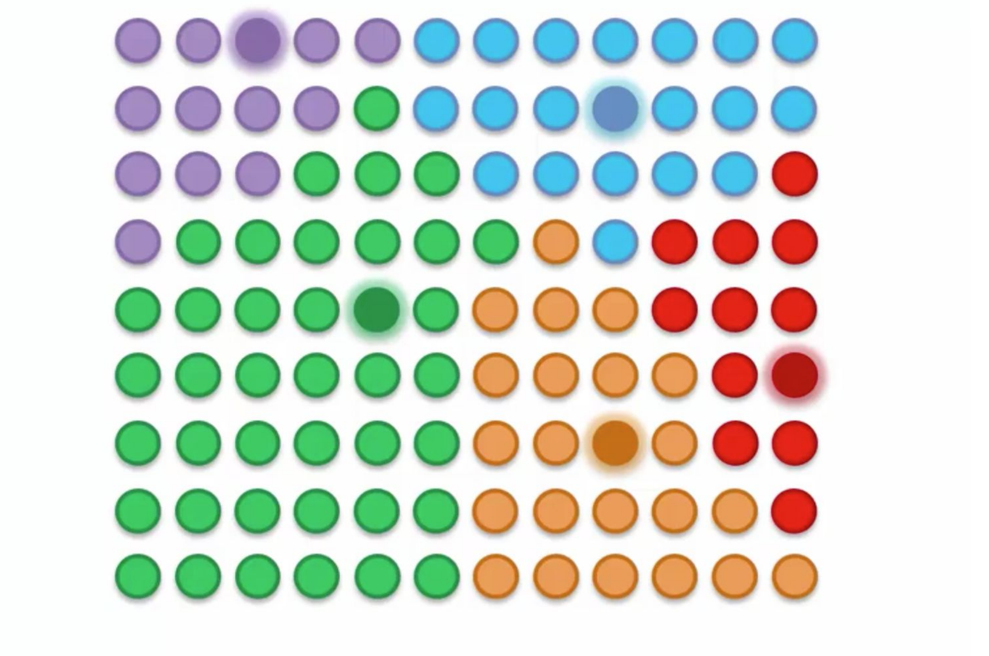




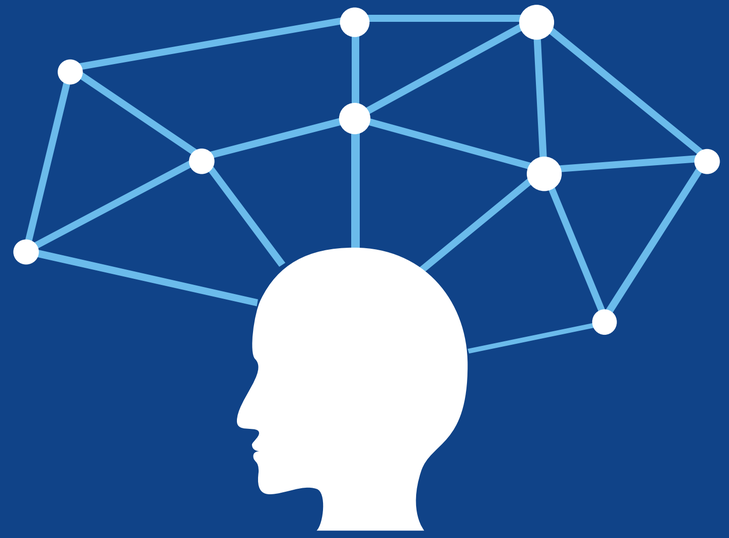


Self Organising Maps





Boltzmann Machines

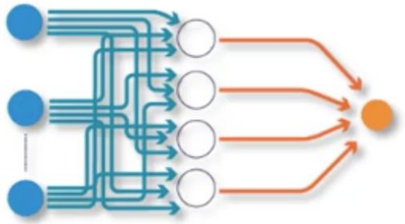


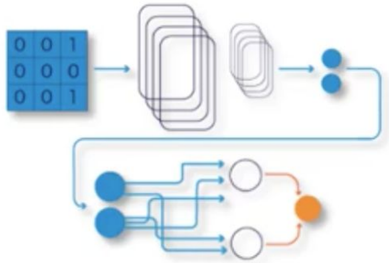
Boltzmann Machines

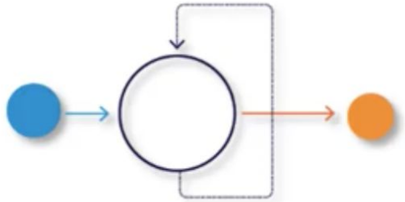
Restricted Boltzmann machines for collaborative filtering
Authors: Ruslan Salakhutdinov University of Toronto, Toronto, Ontario, Canada
Andriy Mnih University of Toronto, Toronto, Ontario, Canada
Geoffrey Hinton University of Toronto, Toronto, Ontario, Canada

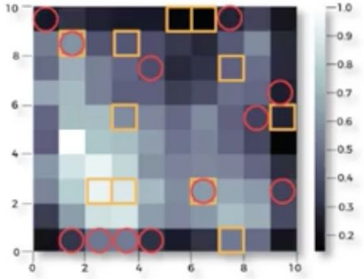
Most of the existing approaches to collaborative filtering cannot handle very large data sets. In this paper we show how a class of two-layer undirected graphical models, called Restricted Boltzmann Machines (RBM's), can be used to model tabular data, such as user's ratings of movies. We present efficient learning and inference procedures for this class of models and demonstrate that RBM's can be successfully applied to the Netflix data set, containing over 100 million user/movie ratings. We also show that RBM's slightly outperform carefully-tuned SVD models. When the predictions of multiple RBM models and multiple SVD models are linearly combined, we achieve an error rate that is well over 6% better than the score of Netflix's own system.

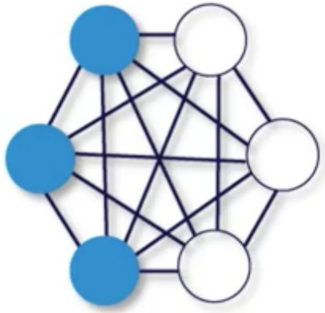
Boltzmann Machines



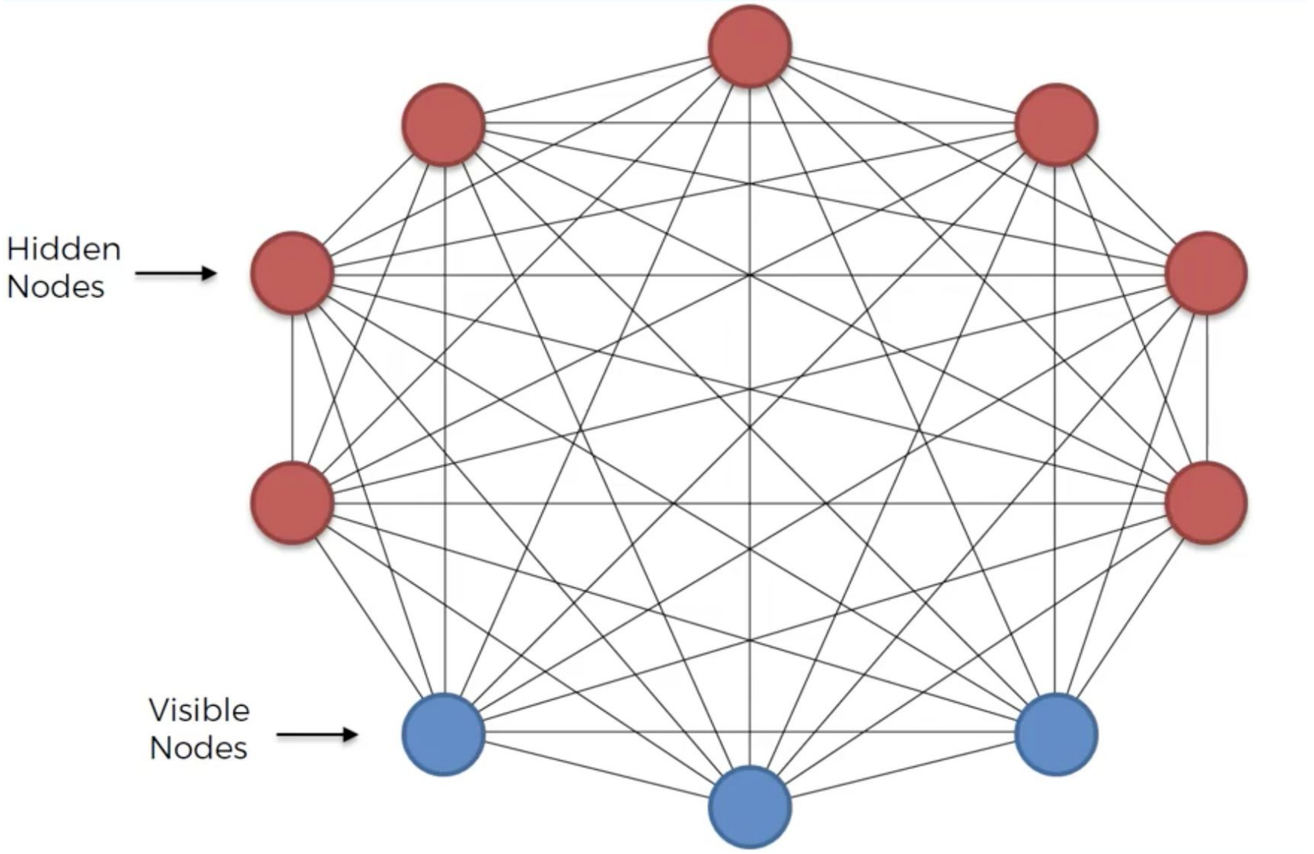




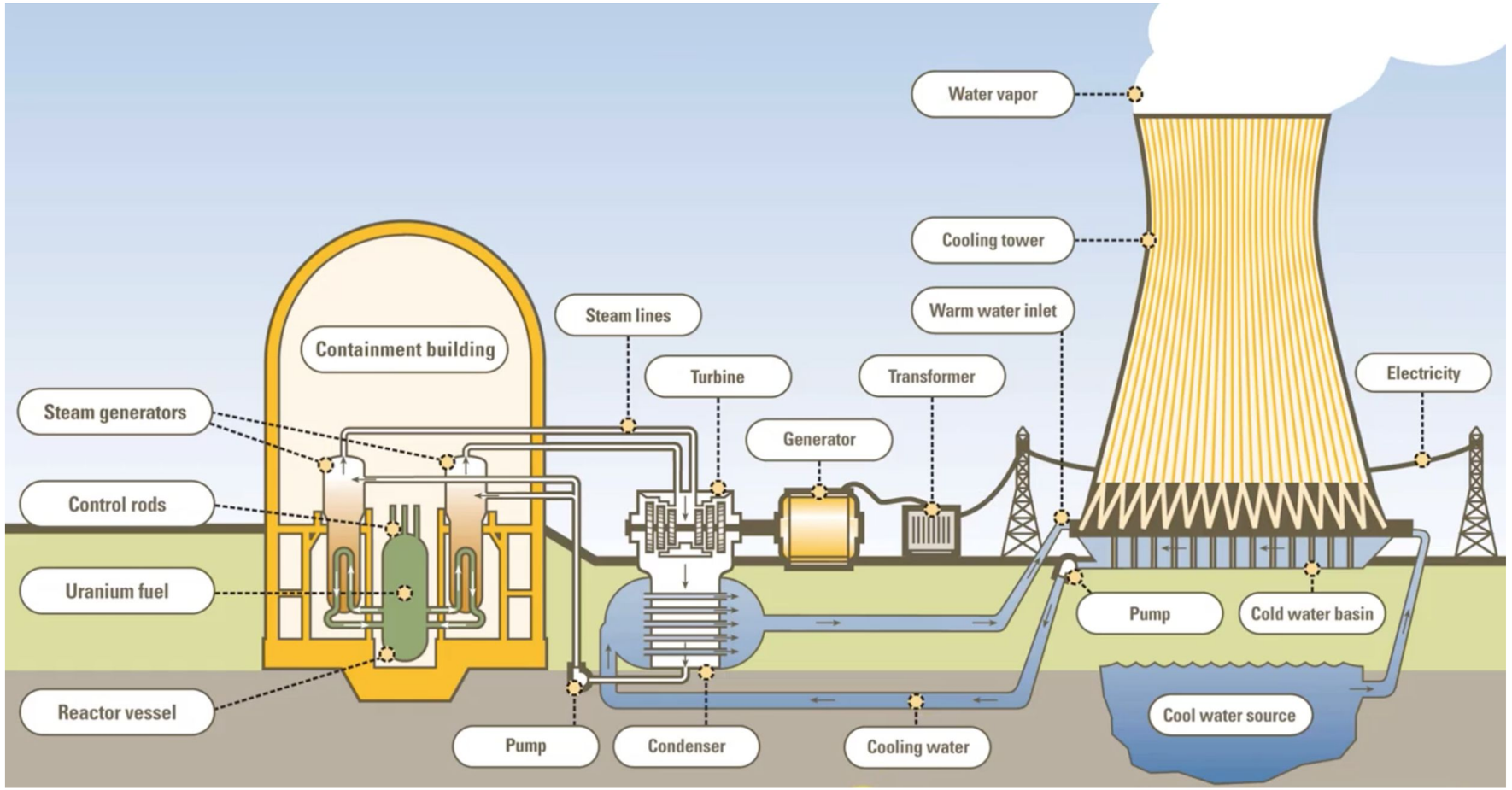




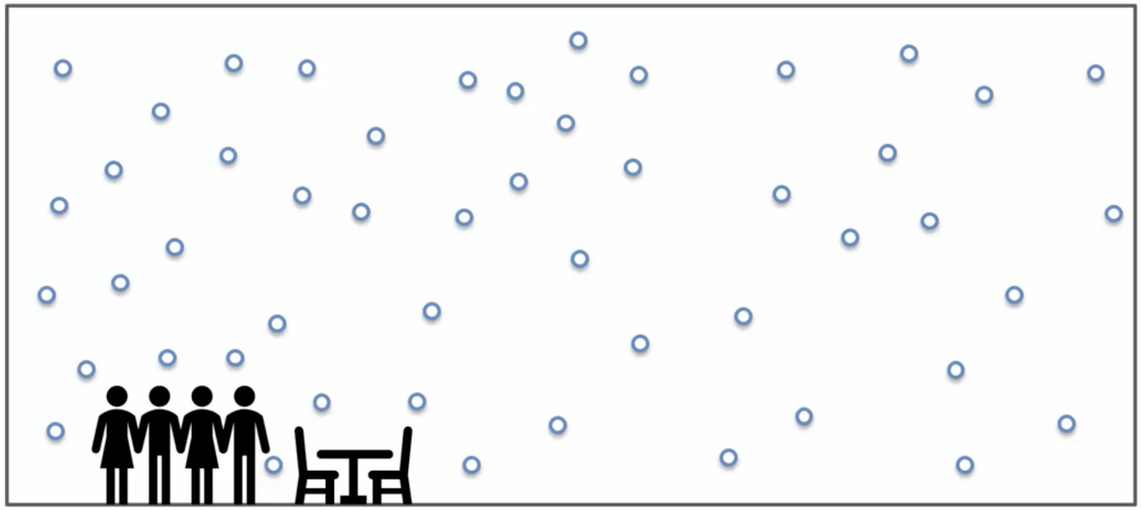
Boltzmann Machines

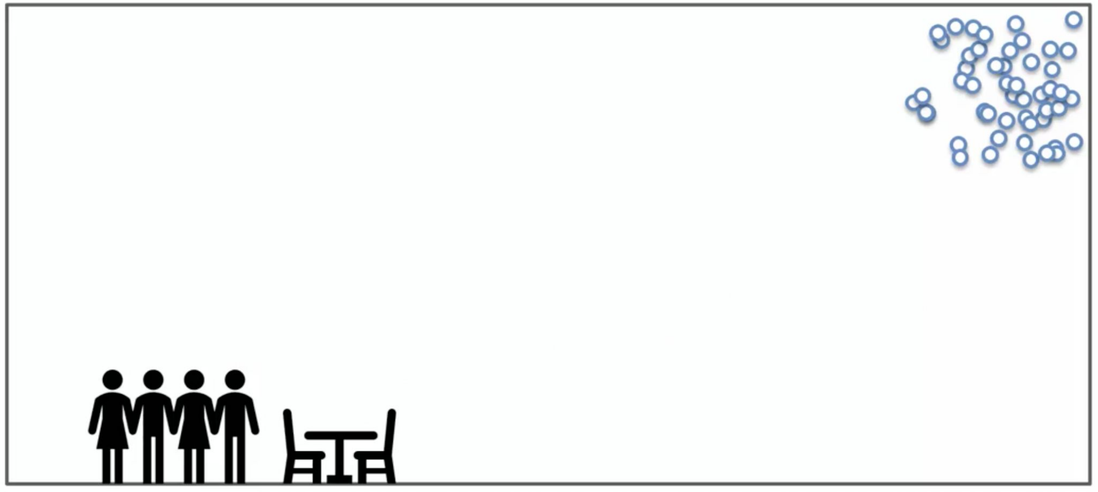


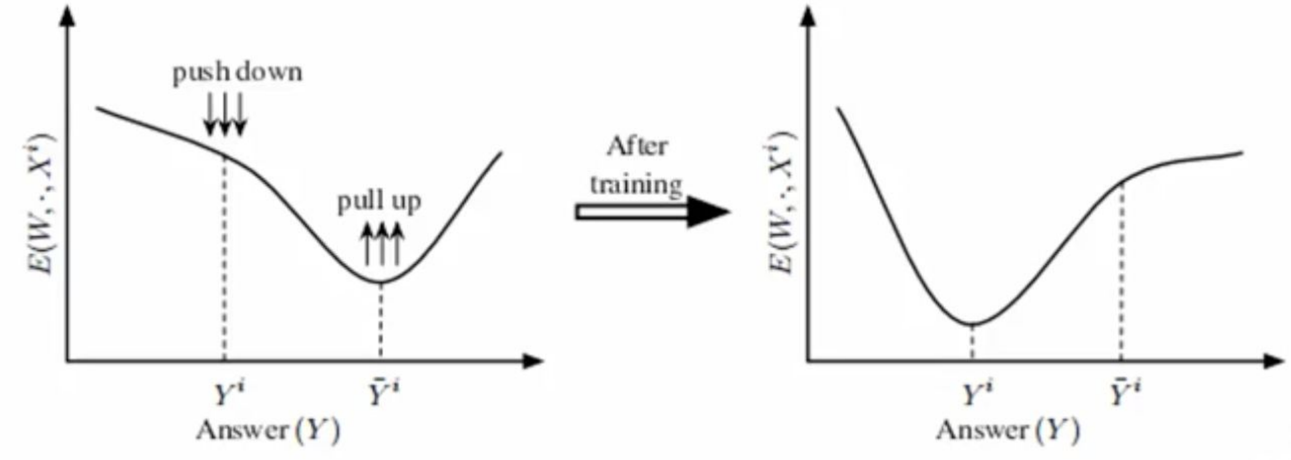
Boltzmann Machines



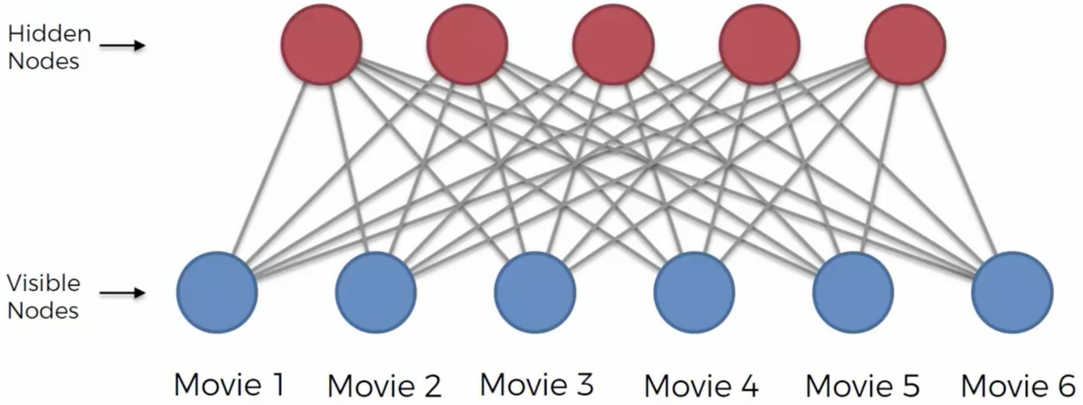
Boltzmann Machines

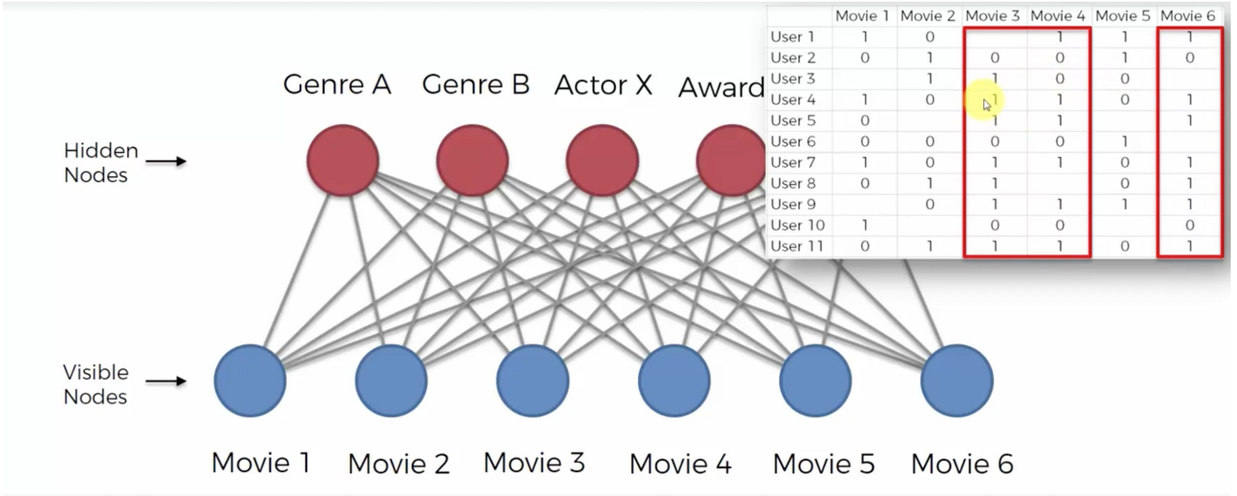






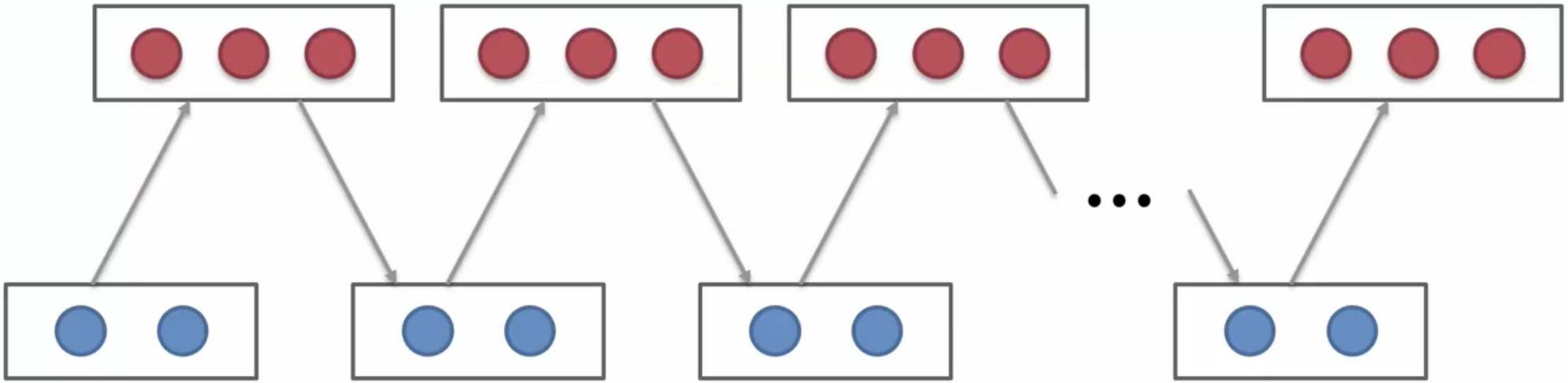
Boltzmann Machines



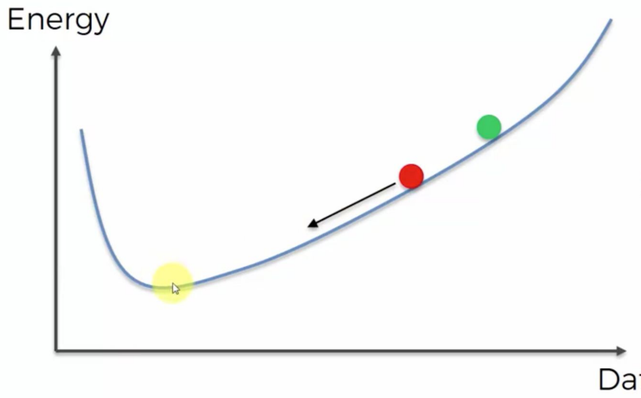


	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

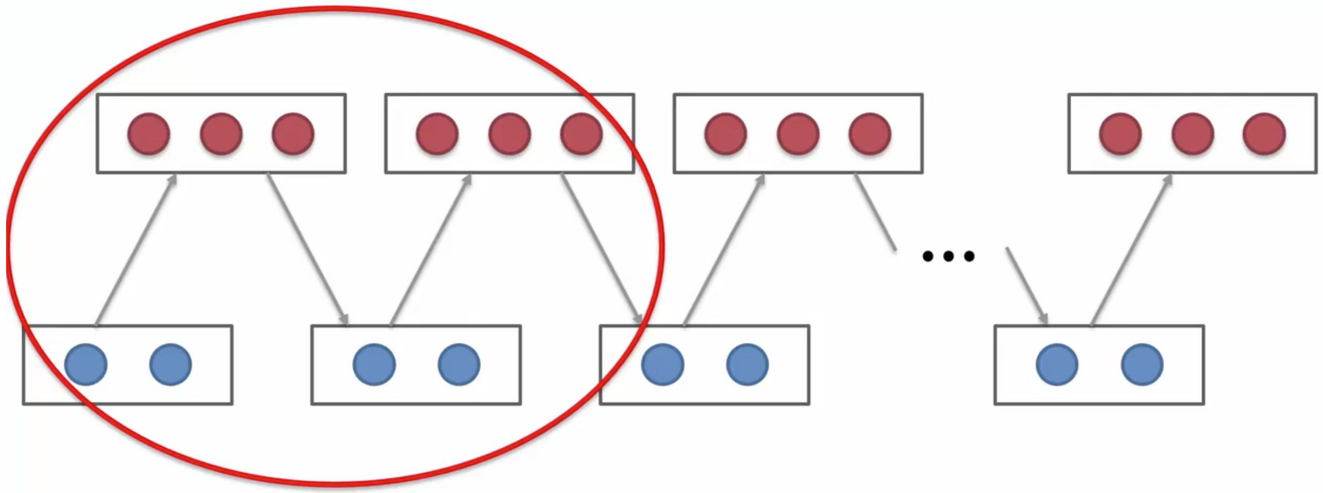
Contrastive Divergence



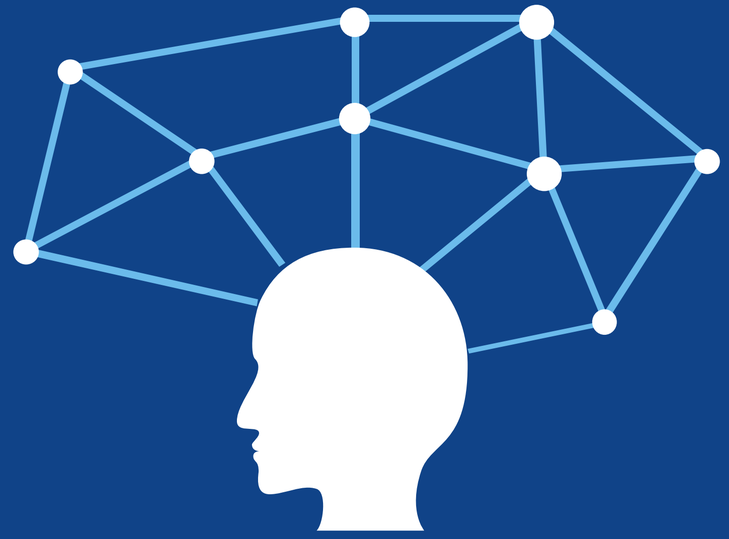
Contrastive Divergence



$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}} = <v_i^0 h_j^0> - <v_i^\infty h_j^\infty>$$



Deep Learning Frameworks



Frameworks

1

Open Source Deep Learning Libraries

LIBRARIES

TensorFlow	Google Brain, 2015 (rewritten DistBelief)
Theano	University of Montréal, 2009
Keras	François Chollet, 2015 (now at Google)
Torch	Facebook AI Research, Twitter, Google DeepMind
Caffe	Berkeley Vision and Learning Center (BVLC), 2013

More Libraries (⅓)

Deeplearning4j

DeepLearning4j (or DL4J) is a popular deep learning framework developed in Java and supports other JVM languages as well. It is very slick, and is very widely used as a commercial, industry-focused distributed deep learning platform. The advantage of using DL4j is that you can bring together the power of the whole Java ecosystem to perform efficient deep learning, as it can be implemented on top of the popular Big Data tools such as Apache Hadoop and Apache Spark.

MXNet

MXNet is one of the most languages-supported deep learning frameworks, with support for languages such as R, Python, C++ and Julia. This is helpful because if you know any of these languages, you won't need to step out of your comfort zone at all, to train your deep learning models. It's backend is written in C++ and cuda, and is able to manage its own memory like Theano. MXNet is also popular because it scales very well and is able to work with multiple GPUs and computers, which makes it very useful for the enterprises. This is also one of the reasons why Amazon made MXNet its reference library for Deep Learning too.

More Libraries (⅔)

Microsoft Cognitive Toolkit

Microsoft Cognitive Toolkit, previously known by its acronym CNTK, is an open-source deep learning toolkit to train deep learning models. It is highly optimized, and has support for languages such as Python and C++. Known for its efficient resource utilization, you can easily implement efficient Reinforcement Learning models or Generative Adversarial Networks (GANs) using the Cognitive Toolkit. It is designed to achieve high scalability and performance, and is known to provide high performance gains when compared to other toolkits like Theano and Tensorflow, when running on multiple machines.

Lasagne

Lasagne is a high-level deep learning library that runs on top of Theano. It has been around for quite some time now, and was developed with the aim of abstracting the complexities of Theano, and provide a more friendly interface to the users to build and train neural networks. It requires Python, and finds many similarities to Keras, which we just saw above. However, if we are to find differences between the two, Keras is faster, and has a better documentation in place.

More Libraries (3/3)

BigDL

BigDL is distributed deep learning library for Apache Spark, and is designed to scale very well. With the help of BigDL, you can run your deep learning applications directly on Spark or Hadoop clusters, by writing them as Spark programs. It has a rich deep learning support, and uses Intel's Math Kernel Library (MKL) to ensure high performance. Using BigDL, you can also load your pre-trained Torch or Caffe models into Spark. If you want to add deep learning functionalities to a massive set of data stored on your cluster, this is a very good library to use.

Many More ...

There are many other deep learning libraries and frameworks available for use today – DSSTNE, Apache Singa, Veles are just a few worth an honourable mention.

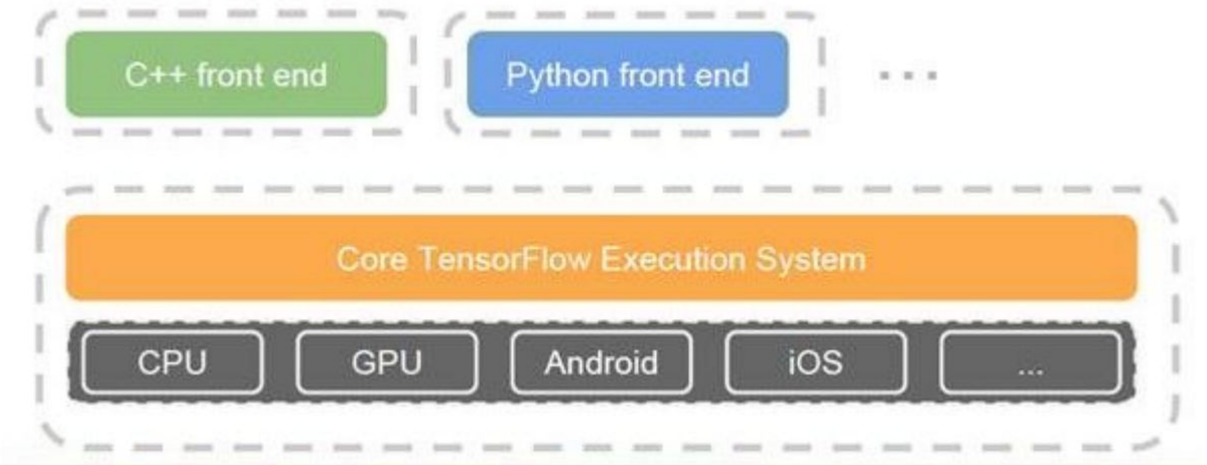
TensorFlow, Theano, Keras, Torch, Caffe

2

Open Source Deep Learning Libraries

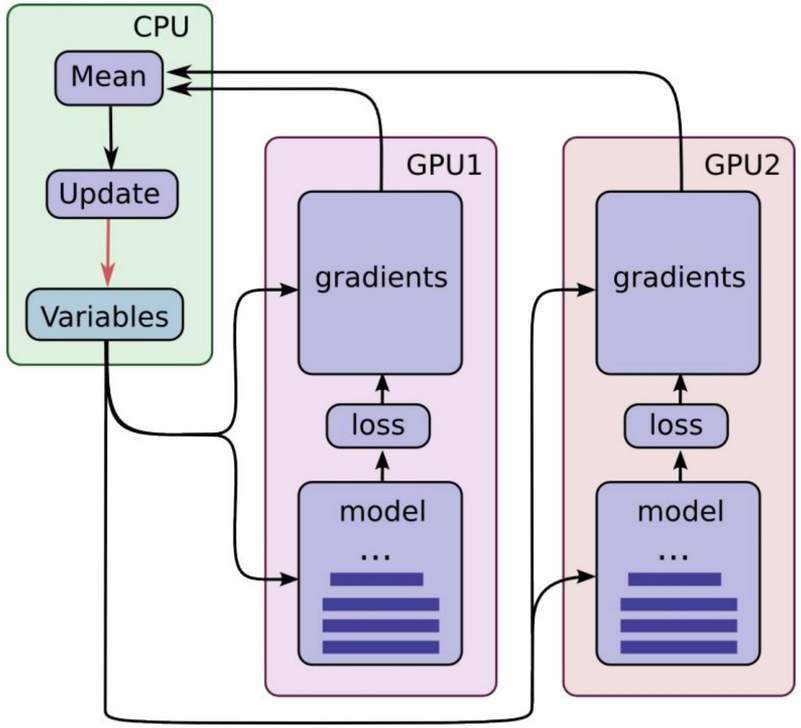
TensorFlow Architecture

1) Low-level core (C++/CUDA)
2) Simple Python API to define the computational graph
3) High-level API (TF-Learn, TF-Slim, Keras…)

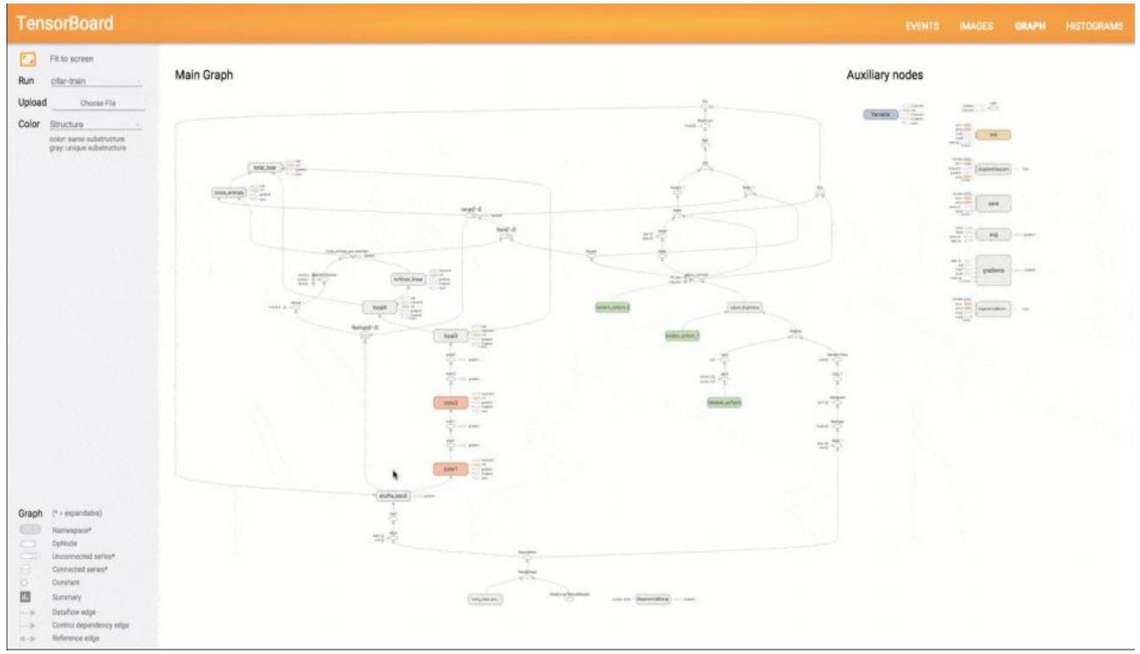


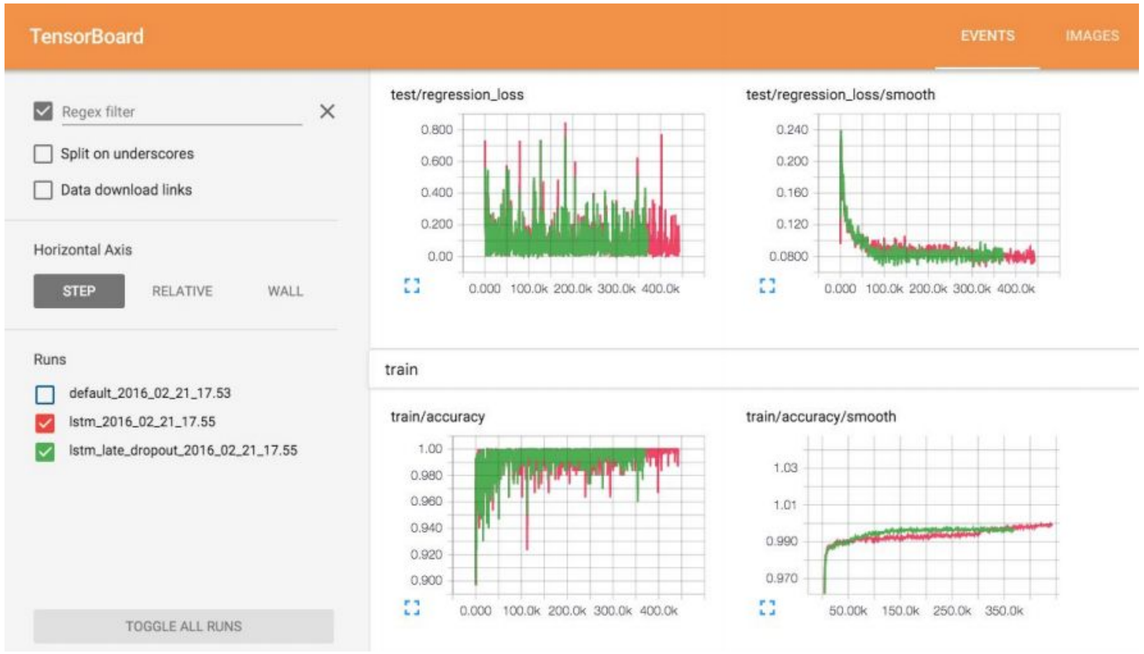
TensorFlow computational graph

- auto-differentiation!
- easy multi-GPU/multi-node
- native C++ multithreading
- device-efficient
- implementation for most ops
- whole pipeline in the graph:
- data loading, preprocessing,
- prefetching...



TensorBoard





TensorFlow Development

+ bleeding edge (GitHub yay!)
+ division in core and contrib => very quick merging of new hotness
+ a lot of new related API: CRF, BayesFlow, SparseTensor, audio IO, CTC,seq2seq
+ so it can easily handle images, videos, audio, text...
+ if you really need a new native op, you can load a dynamic lib
- sometimes contrib stuff disappears or moves
- recently introduced bells and whistles are barely documented

Code + Models, Performance, Model deployment

3

Open Source Deep Learning Libraries

Code and Models

TensorFlow	Theano	Keras	Torch	Caffe
More code and models available (hype), including a lot provided by Google, some of open sourced code is already unusable due to very fast development, caffe models loaded in semi-manual way	- Lots of code to get started (tutorials and older models). - Loss of momentum and so recent models appear more slowly. - State of the art models can always be found, pretrained models can be slow to appear. - Community based: new code can be unstable.	Models from TensorFlow and Theano + converted model from caffe	State of the art classification models available in the ModelZoo. Significant proportion of research projects in torch. Loadcaffe convert caffe models to torch (often involves 0.5 to 2 days of work when non-sequential architecture or “non-standard layers”)	- Extensive code available online (even from the 1st year: >1K forked + significant changes since then) - heaven: state of the art pre-trained models available for a variety of domains - great for fine-tuning networks even without writing code

Community and Documentation

Community : (Github, groups, discussions...)

- Caffe had the largest community
- TensorFlow's is already large and growing.
- Keras' community is growing, while Theano's and Lasagne's are declining

Documentation

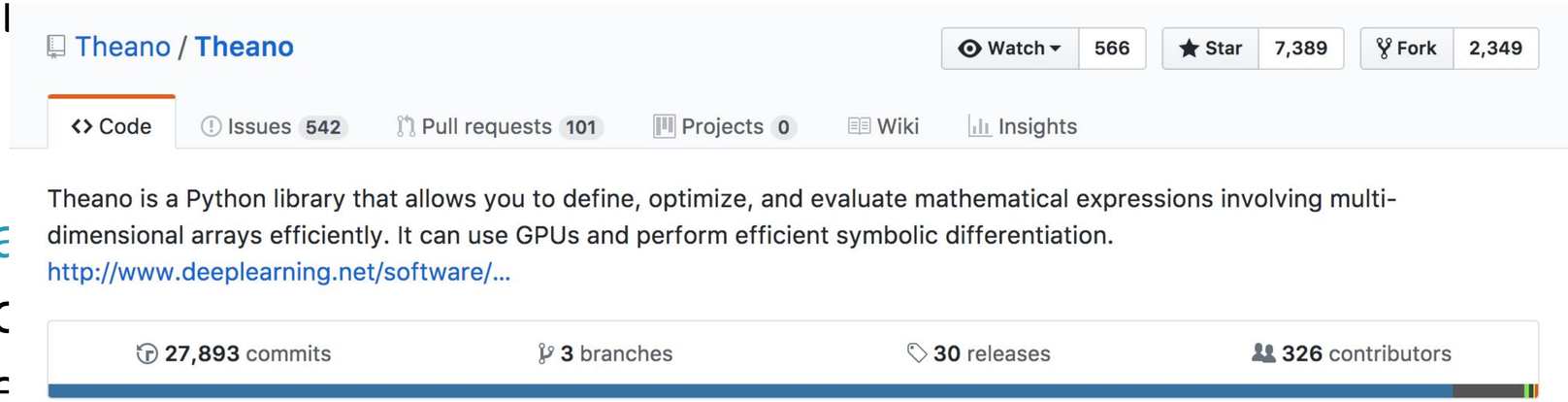
- Great documentation for Theano, Lasagne, Keras and Torch
- Most recent API is not documented for TensorFlow. Tutorials are often outdated.

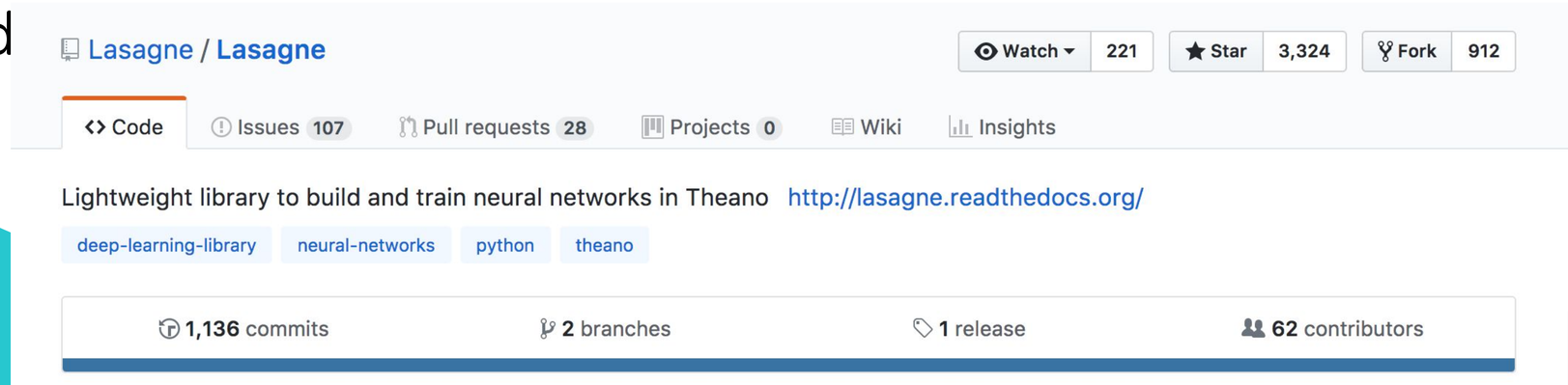
Communit

- Caffe ha
- TensorFl
- Keras' co
  declining

Documenta

- Great doc
- Most rece
  outdated





Performance

TensorFlow	Theano	Keras	Torch	Caffe
Optimized for big models, can be memory hungry, but as fast as others (cuDNN).	- Run-time and memory competitive (efficient RNNs), - Compile time can be a huge pain (Strong -). - Multi-GPU support, not multi-machines.	Cf Theano and TensorFlow	All rely on cuDNN. Advantage : no compilation of models, which saves a lot of time during debugging. Memory : some layers are not very efficient because of inner buffers (fixed with OptNet or Pytorch.)	- Quite fast, eg. NVIDIA TitanX GPU: • Training: ~20 secs/20 iterations (5K images) • Testing: ~70 secs /validation set (50K images) - quick compilation - multi-GPU support but not with python layers - no distributed training

Model Deployment

TensorFlow	Theano	Keras	Torch	Caffe
TF Serving for self-hosted web, Google Cloud Platform for easy web	Although compiled in C++/CUDA, can't be deployed without Python	Cf Theano and TensorFlow	Require LuaJIT to run models. (Can be problematic for integration more than performance)	+ C++ based + stable library - many forks + can be compiled in variety of devices

Extra Features

TensorFlow	Theano	Keras	Torch	Caffe
+Written in C++ +Bindings for Python, C++, Java…	+Python	Python	+ Written in Lua and C/CUDA	Written in C++ Python and matlab interface
+Multi-GPU +Distributed +Windows, Android support	+Multi-GPU	+Multi-GPU +Android support	+ multi-GPU + distributed learning (torch-distlearn) + Stability : testing libraries	+ allows cross-platform (including windows) + very stable

Which Framework to Choose When?

1. You are a student on DL itself?
2. You want to use DL only to get features?
3. You work in industry?
4. You started your internship or POC?
5. You want to solve assignments?
6. You are curious about deep learning?
7. You don’t even know python?

Which Framework to Choose When?

1. You are a student on DL itself? TensorFlow, Theano, Torch
2. You want to use DL only to get features? Keras, Caffe
3. You work in industry? TensorFlow, Caffe
4. You started your internship or POC? Keras, Caffe
5. You want to solve assignments? Keras, Caffe
6. You are curious about deep learning? Caffe
7. You don’t even know python? Keras, Torch

References

https://www.nature.com/articles/nature14539 -- Must read with best papers on Deep Learning (Yann LeCun, Yoshua Bengio & Geoffrey Hinton)

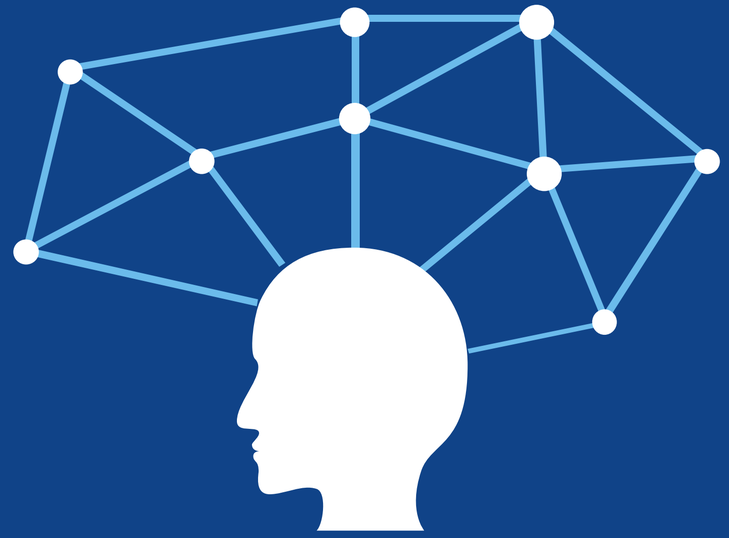
http://karpathy.github.io/ -- Very Good Blogs on Deep Learning Architectures

http://colah.github.io/ -- Theoretical understanding of various algorithms (RNN)

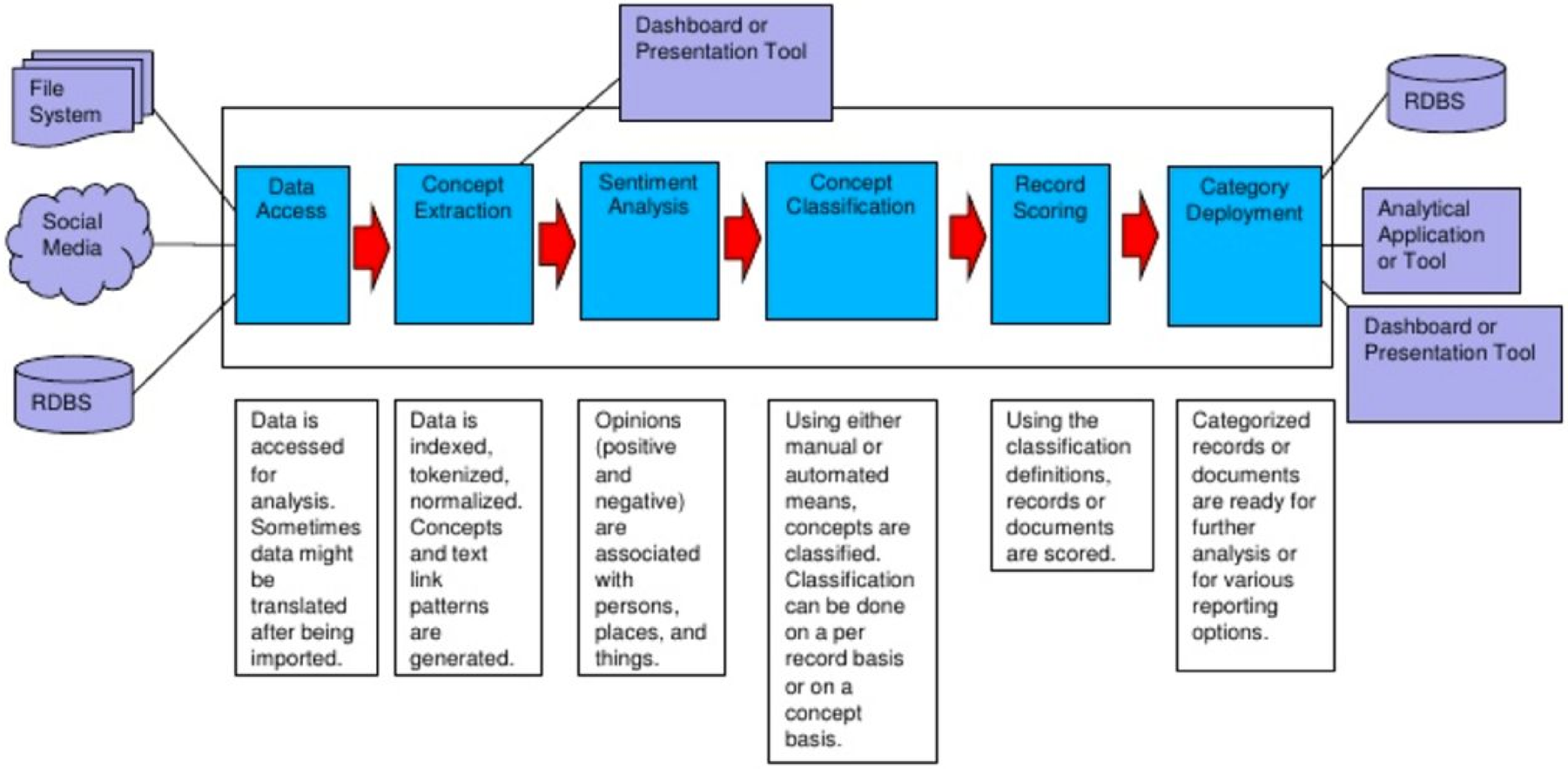
http://deeplearning.net/reading-list/ -- Good Reading List

https://github.com/ChristosChristofidis/awesome-deep-learning -- Everything on Deep Learning

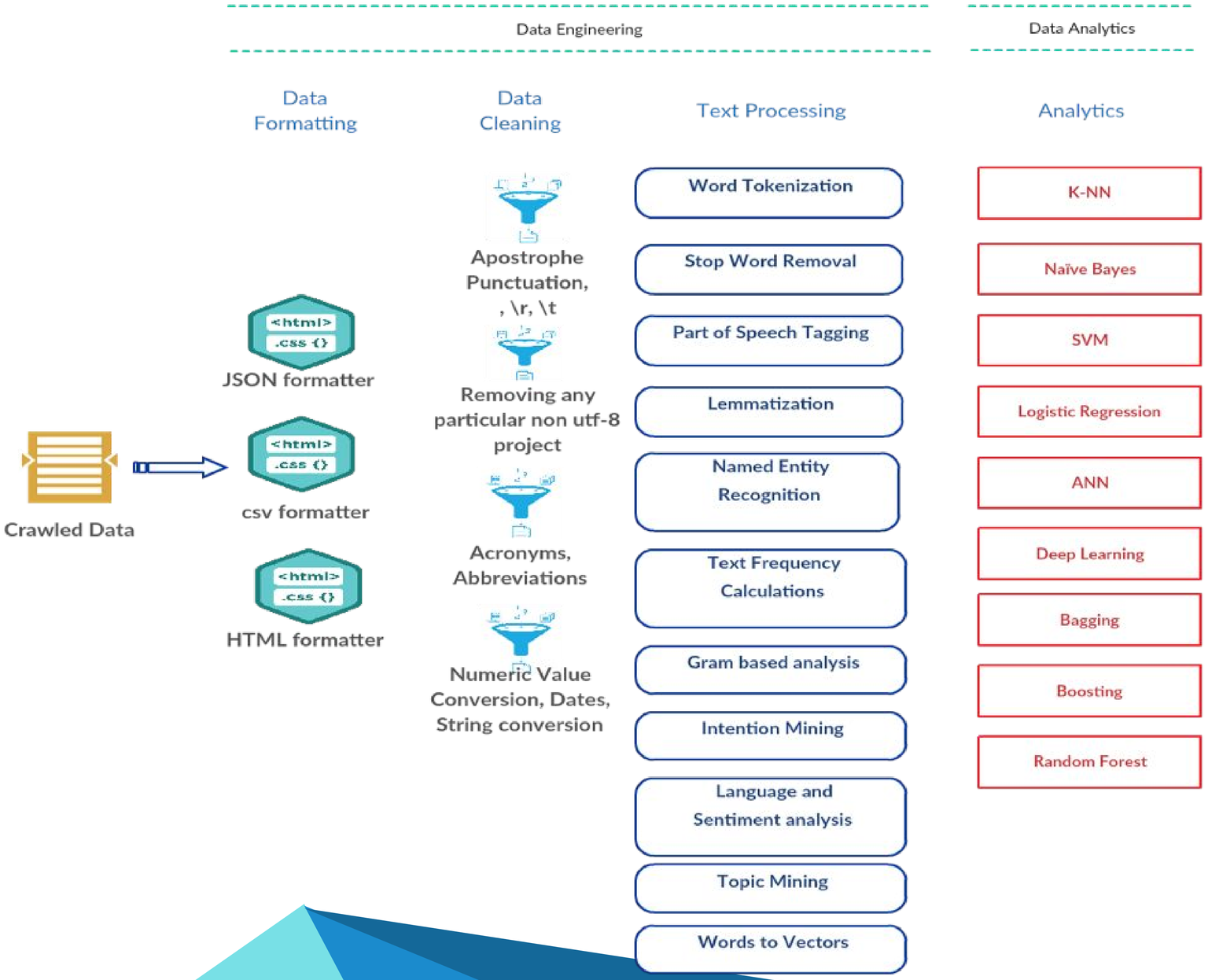
Architectures



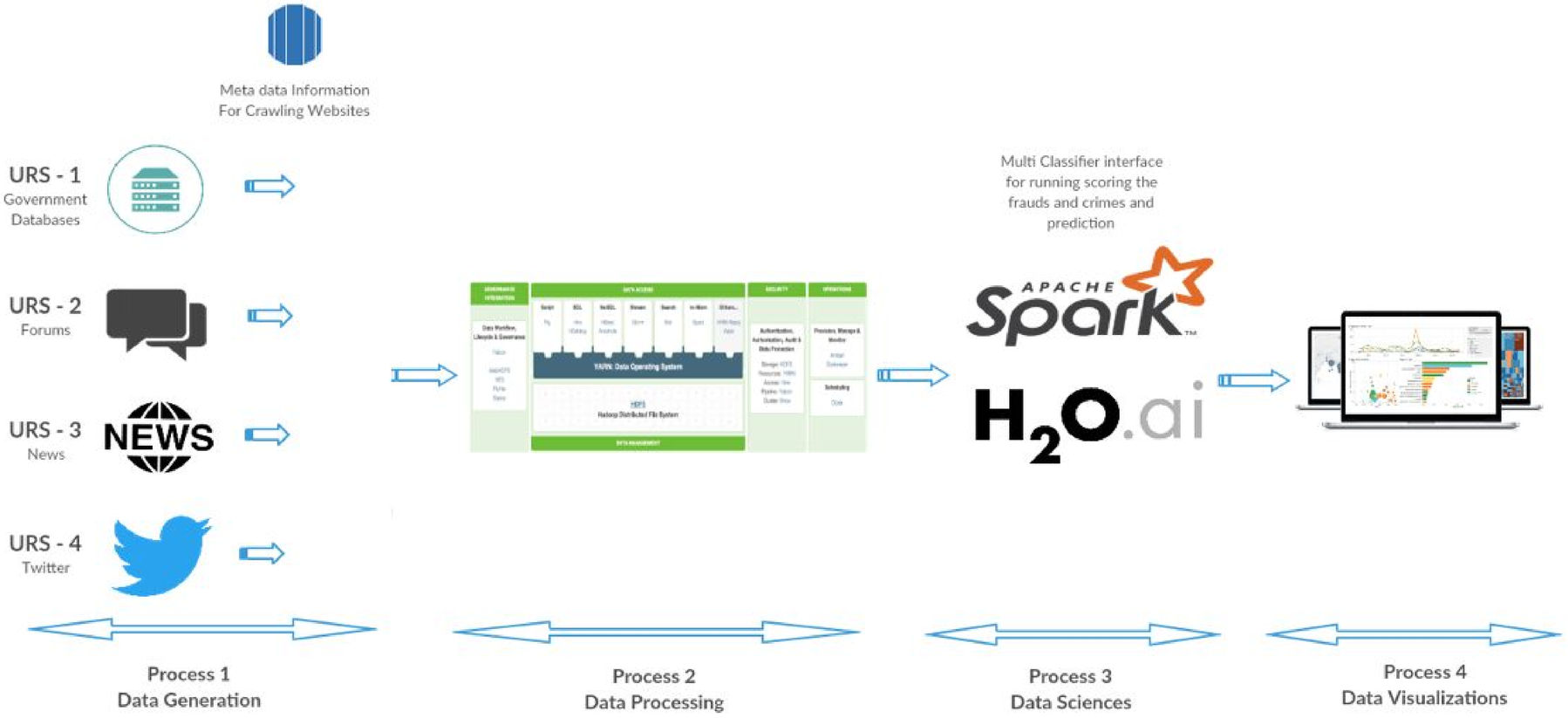
NLP


Dashboard or Presentation Tool
File System
Social Media
RDBS
Data Access
Concept Extraction
Sentiment Analysis
Concept Classification
Record Scoring
Category Deployment
RDBS
Analytical Application or Tool
Dashboard or Presentation Tool
Data is accessed for analysis. Sometimes data might be translated after being imported.
Data is indexed, tokenized, normalized. Concepts and text link patterns are generated.
Opinions (positive and negative) are associated with persons, places, and things.
Using either manual or automated means, concepts are classified. Classification can be done on a per record basis or on a concept basis.
Using the classification definitions, records or documents are scored.
Categorized records or documents are ready for further analysis or for various reporting options.

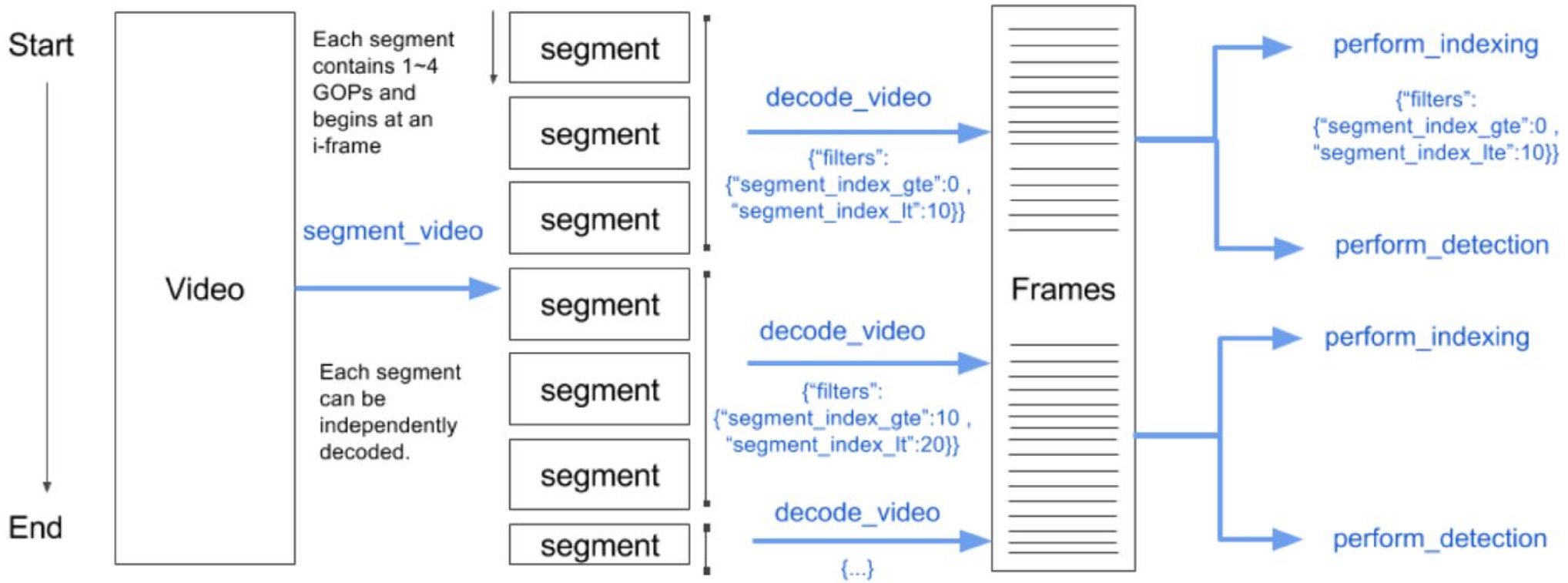
NLP


Data Engineering
Data Analytics
Data Formatting
Data Cleaning
Text Processing
Analytics
Crawled Data
JSON formatter
csv formatter
HTML formatter
Apostrophe Punctuation, , \r, \t
Removing any particular non utf-8 project
Acronyms, Abbreviations
Numeric Value Conversion, Dates, String conversion
Word Tokenization
Stop Word Removal
Part of Speech Tagging
Lemmatization
Named Entity Recognition
Text Frequency Calculations
Gram based analysis
Intention Mining
Language and Sentiment analysis
Topic Mining
Words to Vectors
K-NN
Naïve Bayes
SVM
Logistic Regression
ANN
Deep Learning
Bagging
Boosting
Random Forest

NLP


Meta data Information
For Crawling Websites
URS - 1
Government
Databases
URS - 2
Forums
URS - 3
News
NEWS
URS - 4
Twitter
Multi Classifier interface
for running scoring the
frauds and crimes and
prediction
APACHE
Spark
H2O.ai
Process 1
Data Generation
Process 2
Data Processing
Process 3
Data Sciences
Process 4
Data Visualizations

Video Analytics



Video Analytics

Data Collection

Model Development

TensorFlow

python

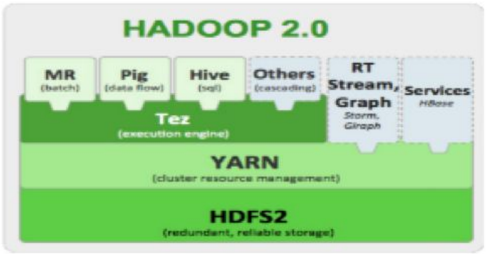


Image Detection and Video Analytics

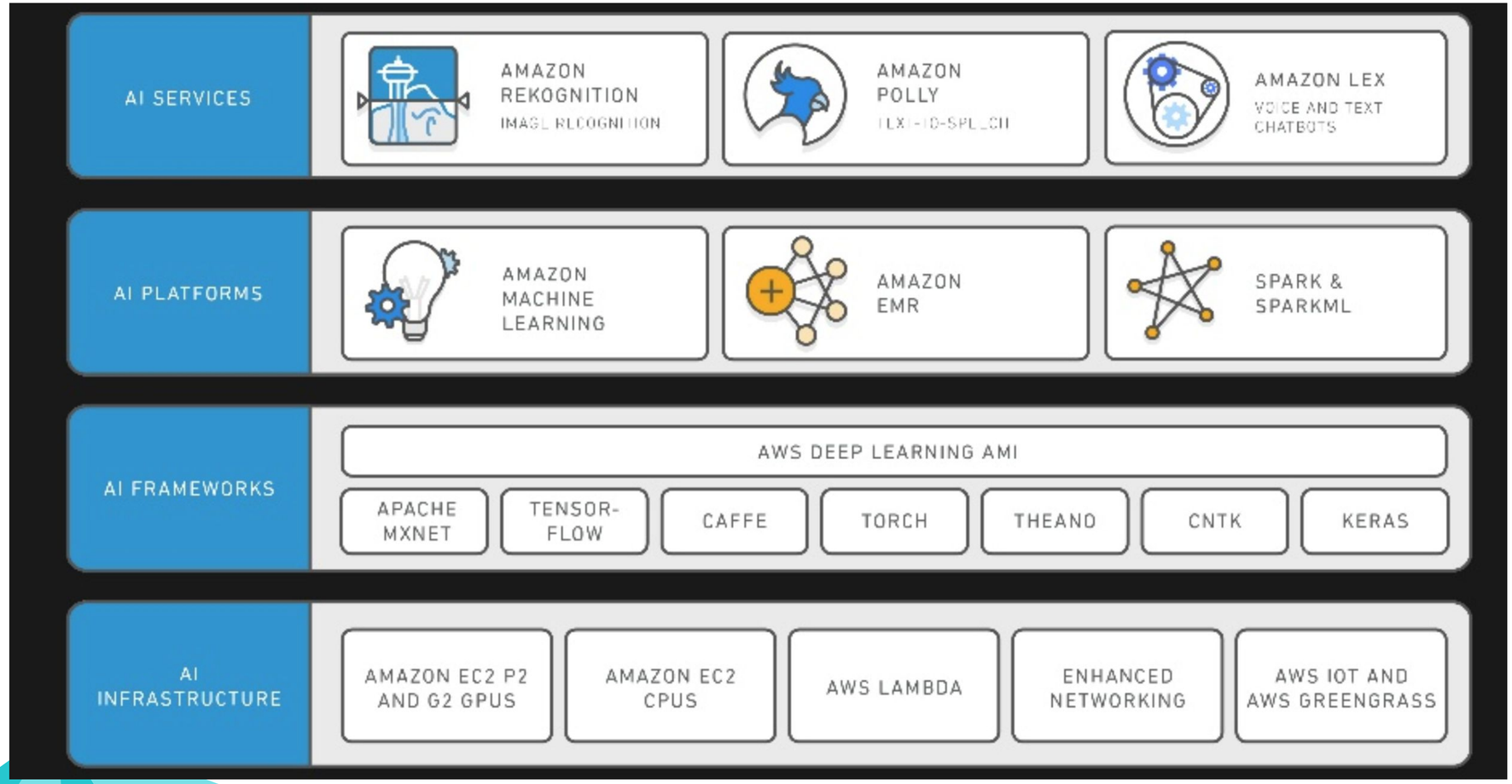
TensorFlow
Model development
+
TensorBoard
Model analysis and tuning

Docker
Virtual container

Flask
Python Web App

Output

AWS and Deep Learning



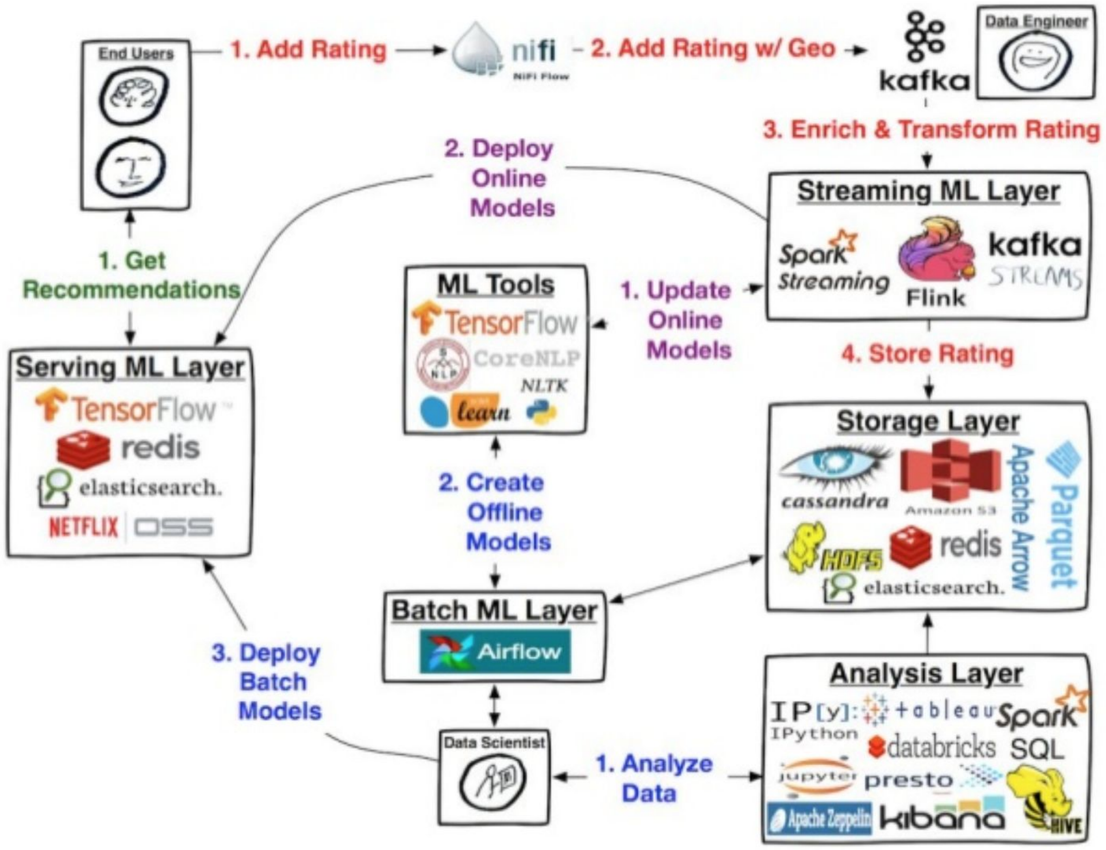
amazon web services

Deep Learning AMI with Source Code (CUDA 8, Amazon Linux)

Sold by: Amazon Web Services

The Deep Learning AMI is a base Ubuntu image provided by Amazon Web Services for use on Amazon Elastic Compute Cloud (Amazon EC2).It is designed to provide a stable, secure, and high performance execution environment for deep learning applications running on Amazon EC2. It has popular deep learning frameworks, including MXNet, Caffe, Caffe2, TensorFlow, PyTorch, Theano, CNTK, Torch and Keras as well as packages that enable easy integration with AWS. It also includes Anaconda Data Science Platform for Python2 and Python3. The Deep Learning AMI is provided at no additional charge to Amazon... Read more

Deployement Plan



Docker

Docker is a virtualization solution (similar to virtual machine). You can download container (or “image”) containing all the frameworks you need.

Why is is useful for DL?

- Installing all the DL frameworks takes time, so download a docker image instead.
- You are sure to have the same running environment on two different machines
- You cannot be root on the cluster.
- Don’t share the code only. Share your docker image also.

References

Neural Networks and Deep Learning

Neural Networks and Deep Learning is a free online book. The book will teach you about:

- Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data
- Deep learning, a powerful set of techniques for learning in neural networks

Neural networks and deep learning currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. This book will teach you many of the core concepts behind neural networks and deep learning.

For more details about the approach taken in the book, see here. Or you can jump directly to Chapter 1 and get started.

http://neuralnetworksanddeeplearning.com/

Deep Learning

An MIT Press book

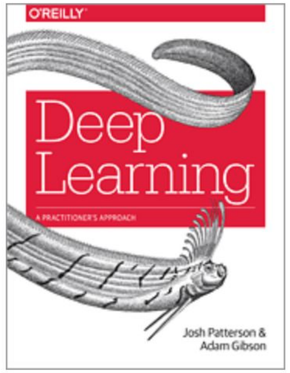
Ian Goodfellow and Yoshua Bengio and Aaron Courville

Exercises Lectures External Links

The Deep Learning textbook is a resource intended to help students and practitioners enter the field of machine learning in general and deep learning in particular. The online version of the book is now complete and will remain available online for free.

The deep learning textbook can now be pre-ordered on Amazon. Pre-orders should ship on December 16, 2016.

For up to date announcements, join our mailing list.



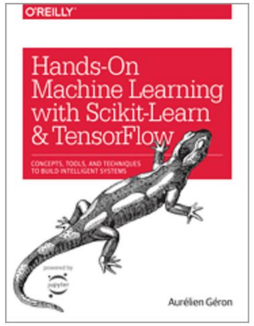
Deep Learning

A Practitioner's Approach

By Adam Gibson, Josh Patterson

Publisher: O'Reilly Media
Release Date: August 2017
Pages: 532

Although interest in machine learning has reached a high point, lofty expectations often scuttle projects before they get very far. How can machine learning—especially deep neural networks—make a real difference in your organization? This hands-on guide not only provides the most practical information available on the subject, but also helps you get started building efficient deep learning networks.



Hands-On Machine Learning with Scikit-Learn and TensorFlow

Concepts, Tools, and Techniques to Build Intelligent Systems

By Aurélien Géron

Publisher: O'Reilly Media
Release Date: March 2017
Pages: 570

Graphics in this book are printed in black and white.

Through a series of recent breakthroughs, deep learning has boosted the entire field of machine learning. Now, even programmers who know close to nothing about this technology can use simple, efficient tools to implement programs capable of learning from data. This practical book shows you how.

Thanks!

Any questions?