Mar 01, 2020

Building Deep Learning-based Search and Recommendation systems

http://bit.ly/iith_ps

publicis
sapient

About the Speaker

Abhishek Kumar

Data Science Lead @ **Publicis Sapient**

Masters from **University of California, Berkeley**

Hal R. Varian Awardee

Google Developer Expert – (Machine Learning)

Speaker @ **O'Reilly Strata & AI conferences**

Pluralsight Author

Top 40 under 40 Data Scientist

Session Logistics

1. Access the work environment using the following link [http://35.224.199.4/]
2. Code and presentation available at link : [http://bit.ly/iith_ps]
3. Connecting to the speakers [*Please send introductory note in LinkedIn invite*]

   Abhishek Kumar ( http://bit.ly/kumarabhishek ,  @meabhishekkumar)
4. Don't forget to tweet #PSAtIITH @PubSapientIndia

Audience Profiling

1. Machine Learning?
2. Deep Learning?
3. Search and Recommendation Systems?
4. Tensorflow?

Session Agenda

4 Levels of Learning

FOUNDATION [30 MINS]

1. High Level Overview For Problem Space [Search, Recommendations, Learning to Rank]
2. Deep Learning Primer
3. Why Tensorflow for Deep Learning ?

Search [1 HR]

1. Embedding
2. Demo : Embedding in TF
3. Image Search using CovNet
4. Demo : Image Search in TF

BREAK

Recommendation and LTR [1 HR]

1. Embedding in RecSys
2. Demo : Build DL based RecSys with Explicit Feedback using TF
3. Demo : Build hybrid RecSys using TF
4. Learning on Rank
5. Demo : Build DL based RecSys with Implicit Feedback using TF

PRODUCTION [30 Mins]

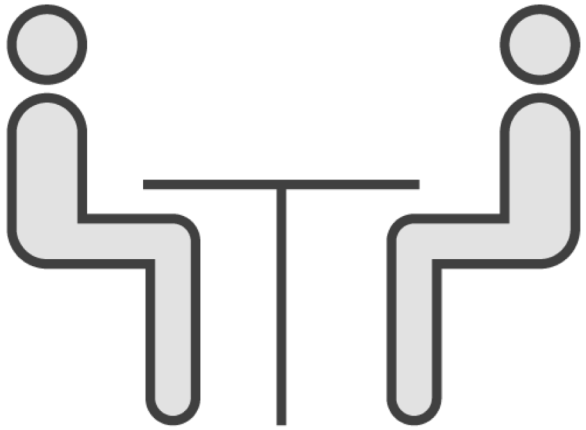
1. TF in Production : Training & Inference
2. Tensorflow Serving
3. RecSys Architecture

By the end of this session...

1. You will have basic foundation of Deep Learning.
2. You will have good understanding of Recommendation Systems, Search and Ranking Systems
3. You will be able to transform the concepts and build DL models using Tensorflow
    1. Deep learning based Image retrieval system
    2. Deep learning based hybrid RecSys on explicit feedback
    3. Deep learning based
4. You will have high level idea to take the lab scale solution to a production ready system

Quick Chat with Your Neighbor

1. Introduce yourself to your neighbor
2. What they are looking to learn from the tutorial



Problem Space : Search and Recommendation

We now live in the connected age

1900

Manufacturing Economy

Mass manufacturing makes industrial powerhouses successful

"A customer can have a car painted any color he wants as long as it's black"

1960

Distribution Economy

Global connections and transportation systems make distribution key

"Strategy is...globalization, taking your products around the world; be the low-cost producer"

1990

Information Economy

Connected PCs and supply chains mean those that control information flow dominate

"The great challenge... is to make productive the tremendous new resource, the knowledge worker. "

2008+

Connected Economy

iPhone and Facebook launch in 2007/8 heralding a new era in transparency, empowerment, and experimentation.

"The customer is the center of your universe."

**Adapted from Forrester Research "Age of the Customer" graphic*

The Connected Consumer is in charge

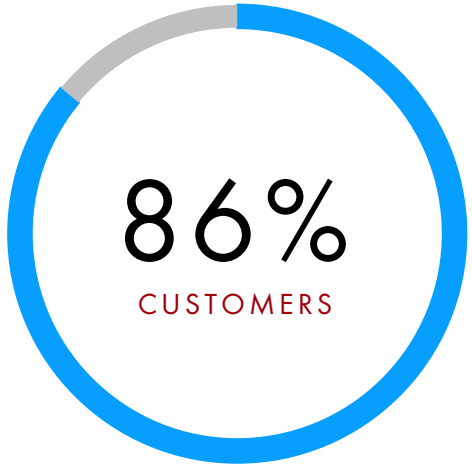
Empowered and demand transparency

Value experiences over most things

Embrace and seek new companies to engage with

Demand personalization and real-time relevancy

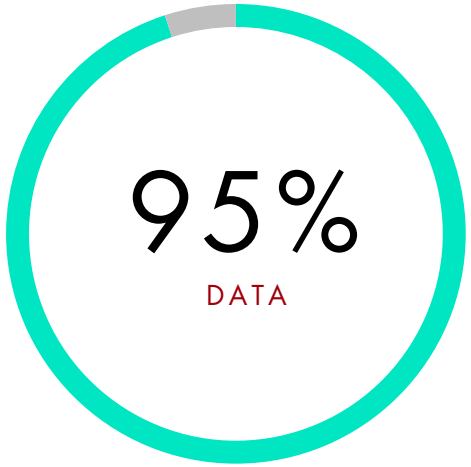
Research proves that consumer experience does matter



SAID THAT **PERSONALIZATION** HAS HAD SOME IMPACT ON PURCHASING DECISION

IS SPENT ON PRODUCT **DISCOVERY & RESEARCH** ONLINE BY 50% CUSTOMERS

DEMAND IMPROVED RESPONSE TIME



WITHIN ORGANIZATION REMAINS UNTAPPED

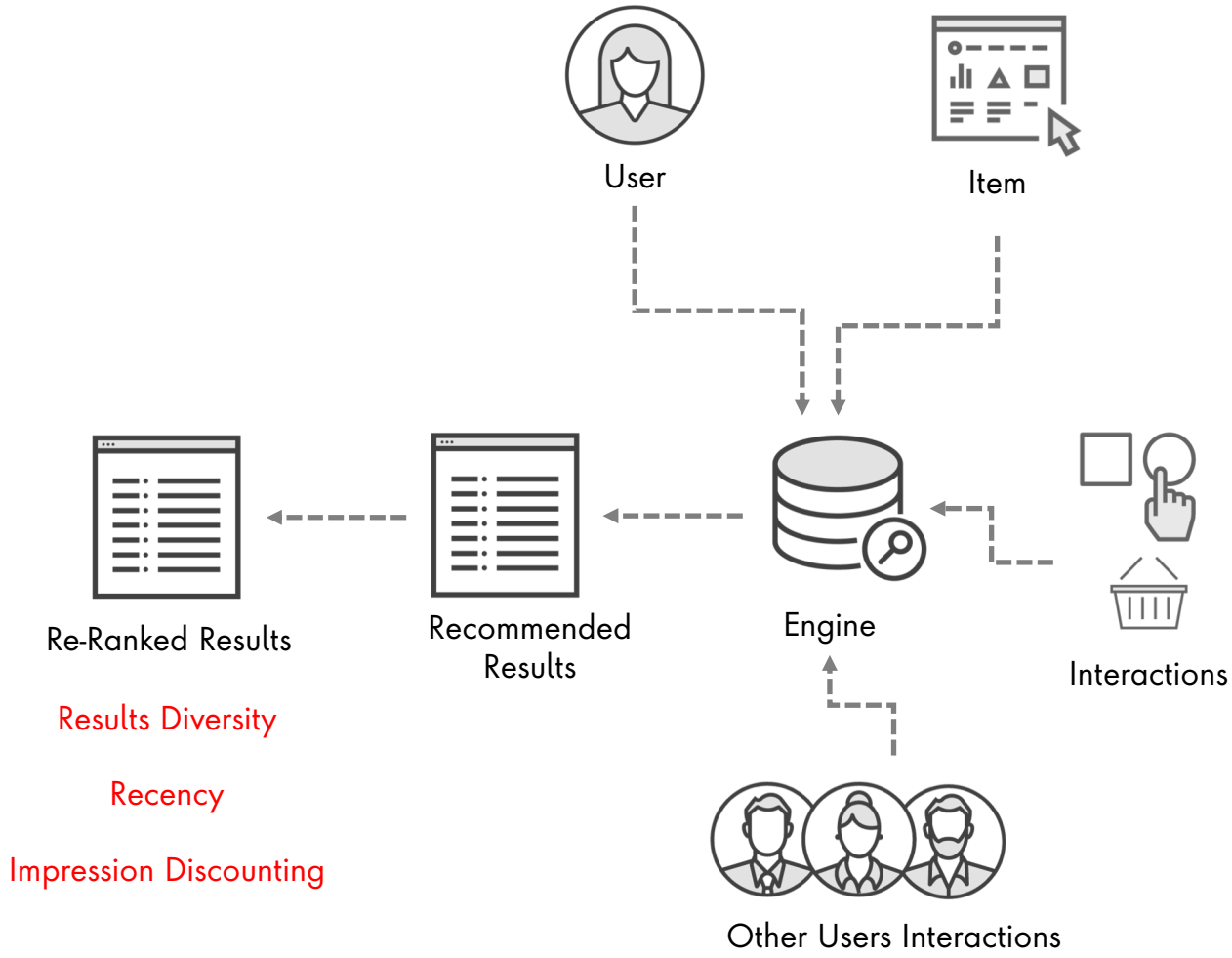
Source : http://www.nextopia.com/wp-content/uploads/2015/01/personalization-ecommerce-infographic.png
https://blog.hubspot.com/blog/tabid/6307/bid/23996/Half-of-Shoppers-Spend-75-of-Time-Conducting-Online-Research-Data.aspx
http://possible.mindtree.com/rs/574-LHH-431/images/Mindtree%20Shopper%20Survey%20Report.pdf
http://www.getelastic.com/using-big-data-for-big-personalization-infographic/

Problem Space : Search

Challenges

- How to represent users and items?
- How to build hybrid systems with both interactions (collaborative) and user/item metadata ?
- How to use dynamic user behaviors?
- How to use implicit (view, share) feedback ?

Recommendation Engines



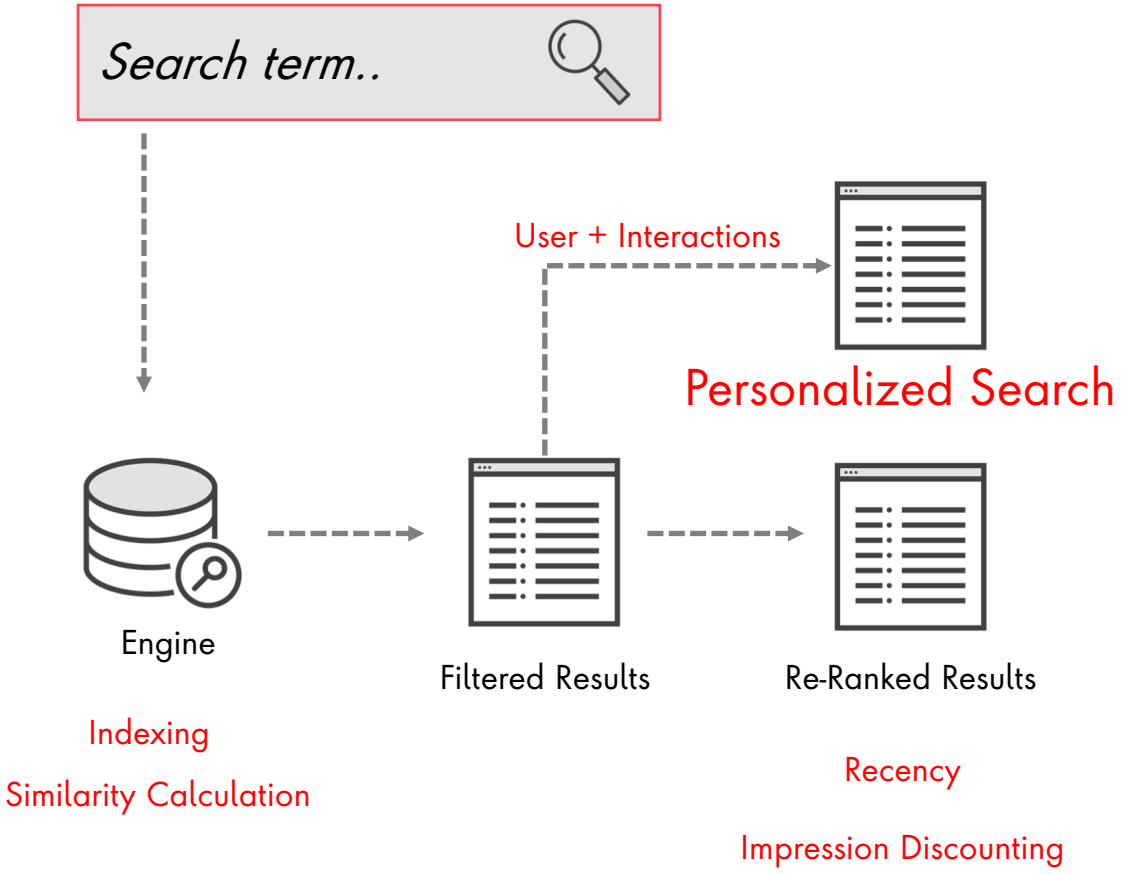
Results Diversity

Recency

Impression Discounting

Problem Space : Search

Search Engines



Challenges

- How to represent text, images, audios
  - TF-IDFs?
  - Metadata for binary?
- Search in other languages?
- Search Quality
  - Well-ranked results
  - *By providing better search results, Netflix estimates that it is avoiding canceled subscriptions that would reduce its revenue by $1B annually. [Link]*

Why Deep Learning for Search and Recommender System?

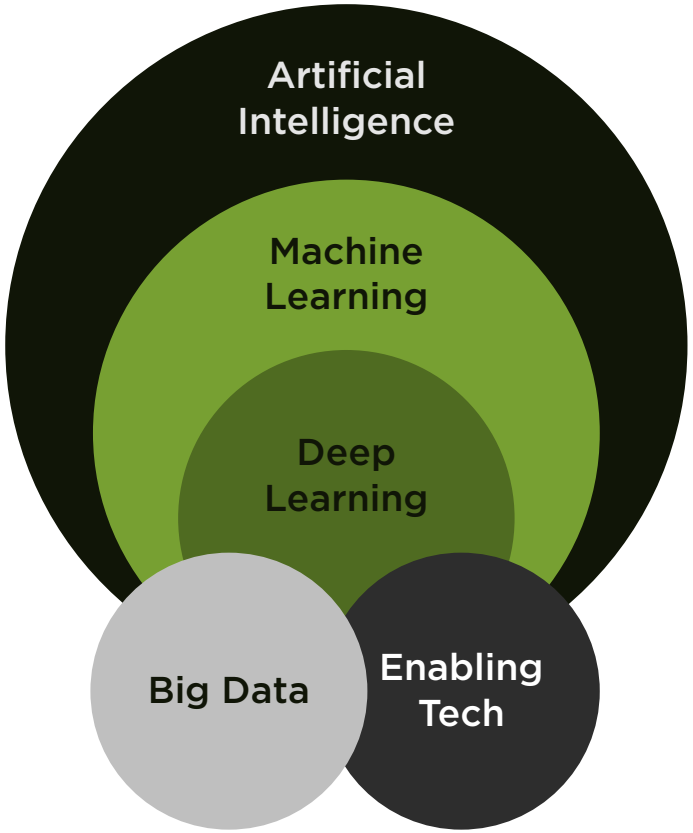
- Direct content Feature extraction instead of metadata
  - Text, Image, Audio
- Better representation of users and items for Recsys
- Hybrid algorithms and heterogeneous data can be used
- Better suited to model dynamic behavioral patterns and complex feature interactions

Deep Learning Primer

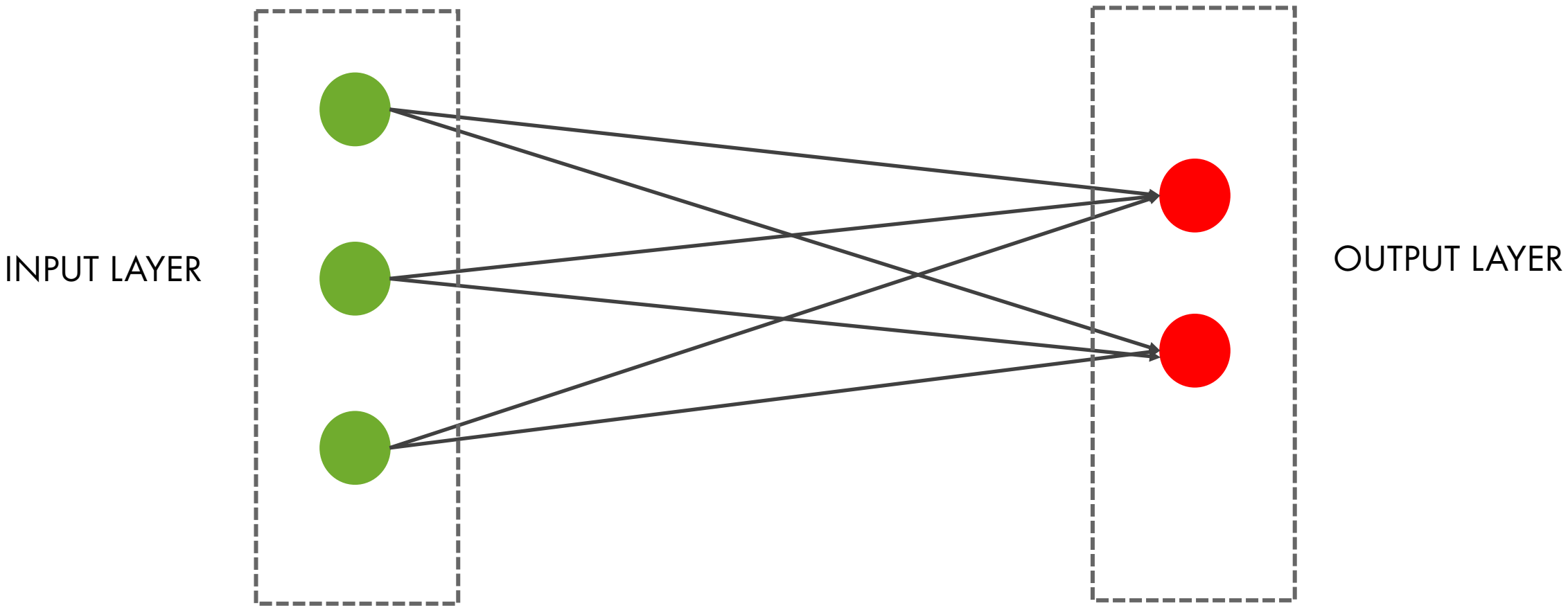
What is Deep Learning ?

Class of machine learning algorithms

- That uses hierarchy of non-linear processing layers and complex model structures
- Layers learn to represent different representation of data
- Higher level features are constructed from lower level abstract features
- Trendy name for “Neural Networks with deep layers”

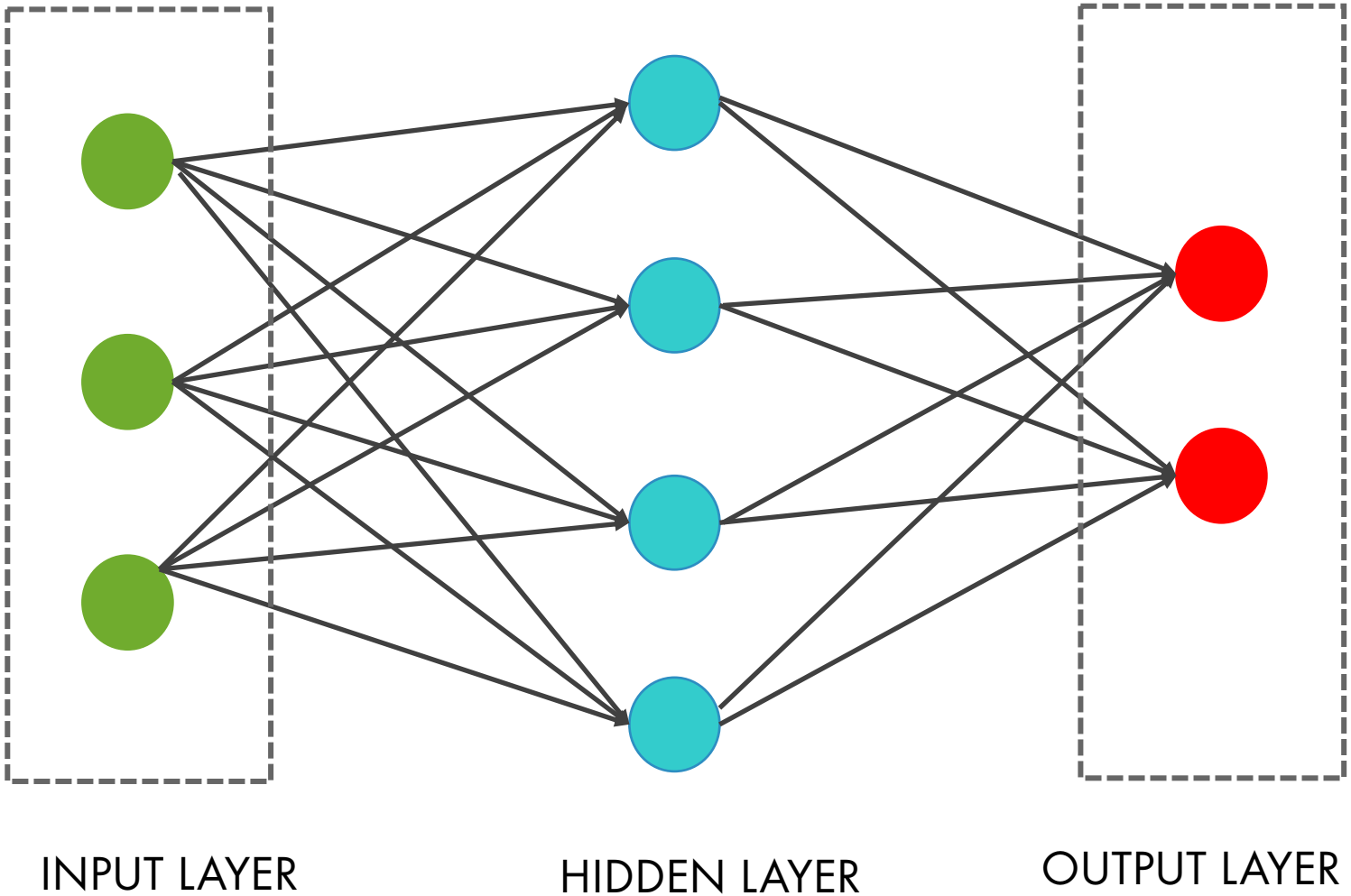


Simple Neural Network With 2 Layers



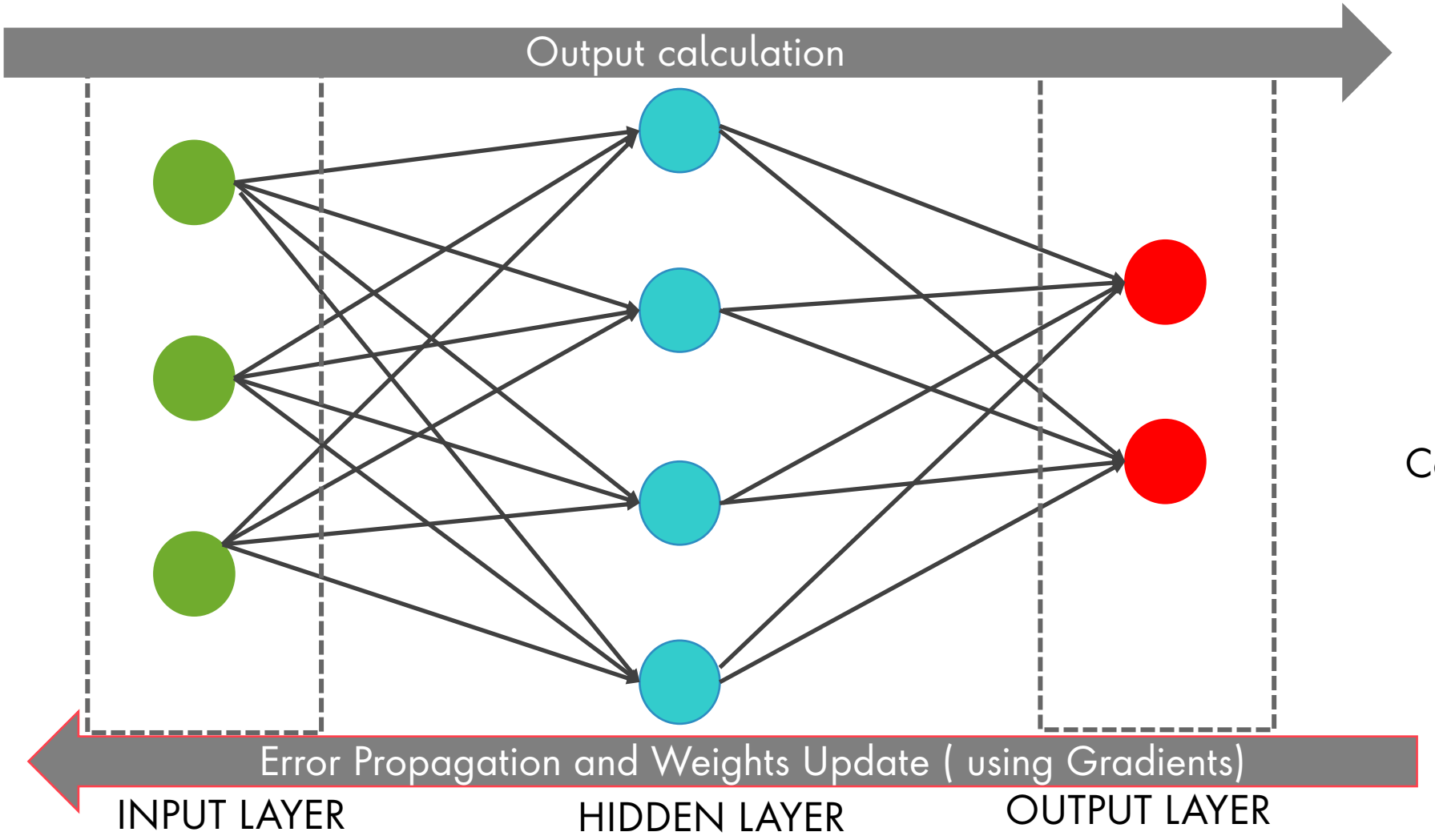
Limitation : Can learn only linear relationship

Simple Neural Network with At Least One Hidden Layer



Universal Approximator

Neural Network Training : Backpropagation



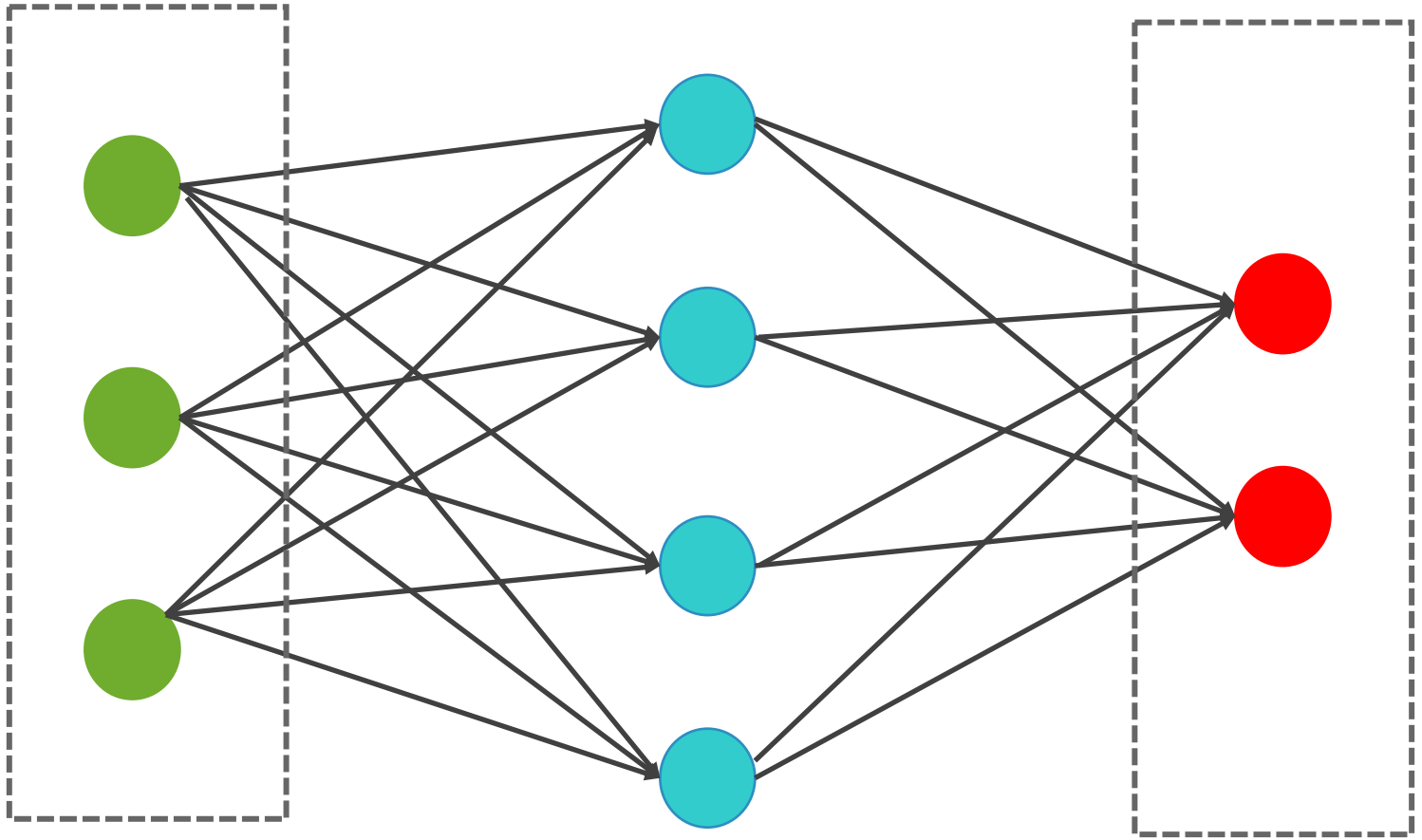
What Changed Now ?

- More data
  - More complex models need more data to avoid overfitting
  - Deep learning models have higher VC dimension

- Computing Power
  - Computing power have increased significantly
  - Specialized hardware such as GPUs and TPUs

- Research Breakthrough
  - Hinton's work on layerwise training led a new paradigm to train deep networks
  - Non-saturating activation functions (variation of ReLUs)
  - Dropouts helped to achieve regularization easily
  - Adaptive learning rate helped to avoid problems of local minima and led to better convergence

Popular Neural Network Architectures : Deep Feed forward



INPUT LAYER

MULTIPLE HIDDEN LAYERS

OUTPUT LAYER

Popular Neural Network Architectures : Convolution Neural Network (CovNet)

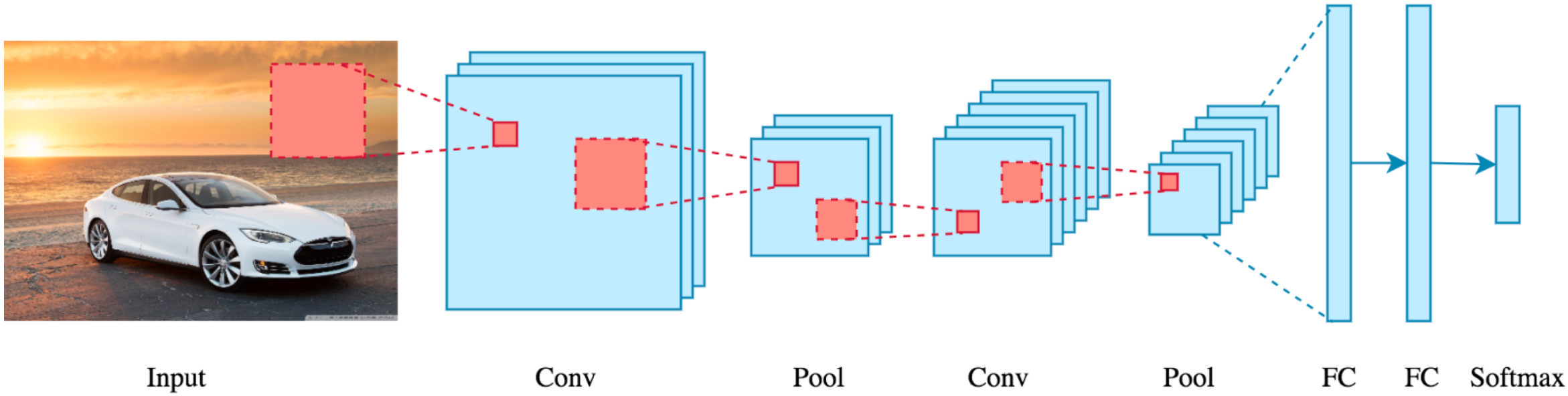


Image Credit : https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolution Neural Network (CovNet) : Components

CONVOLUTION

Mathematical Operation on two sets of information

NON-LINEARITY

POOLING

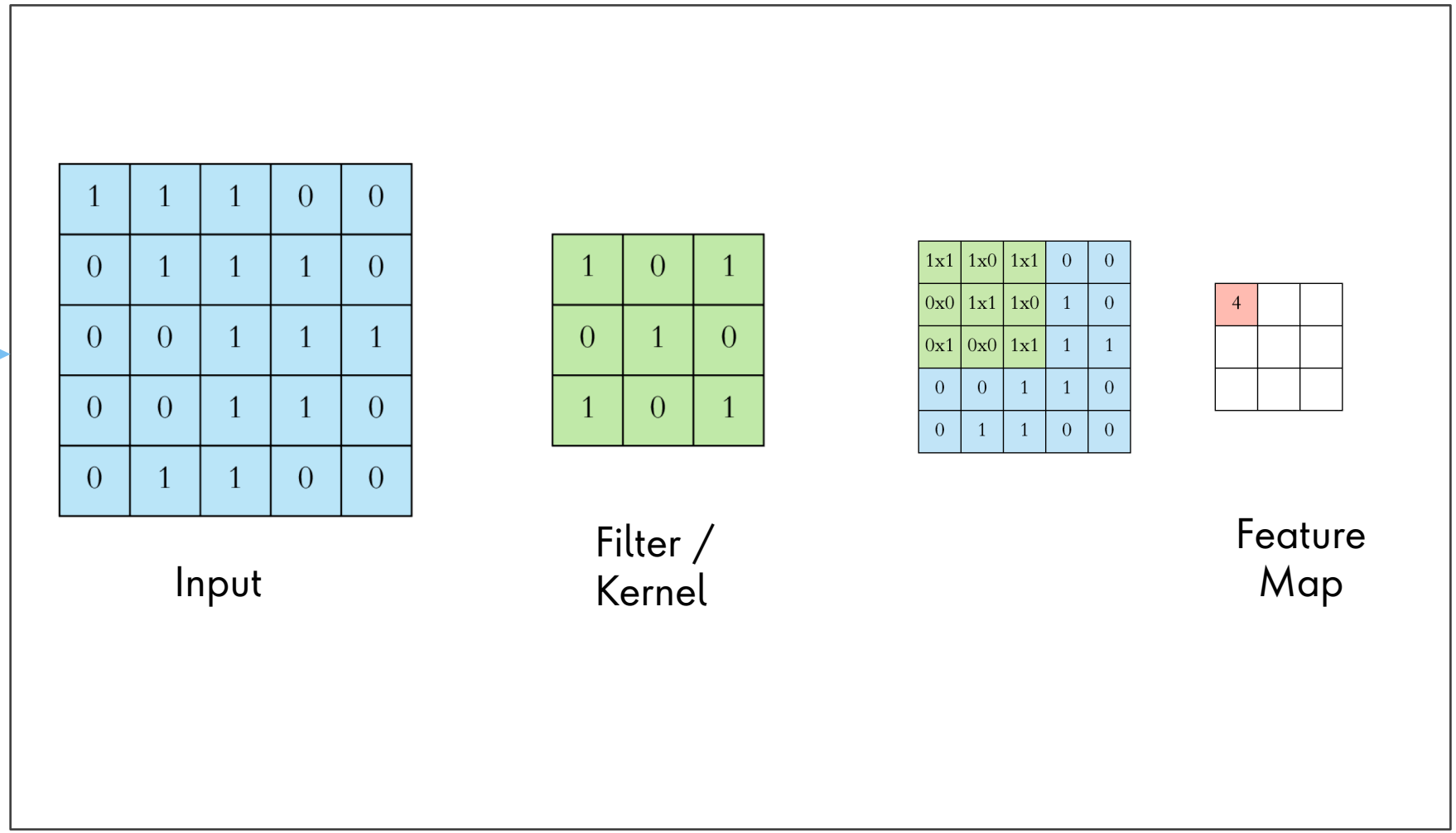


Image Credit : https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Convolution Neural Network (CovNet) : Components

CONVOLUTION

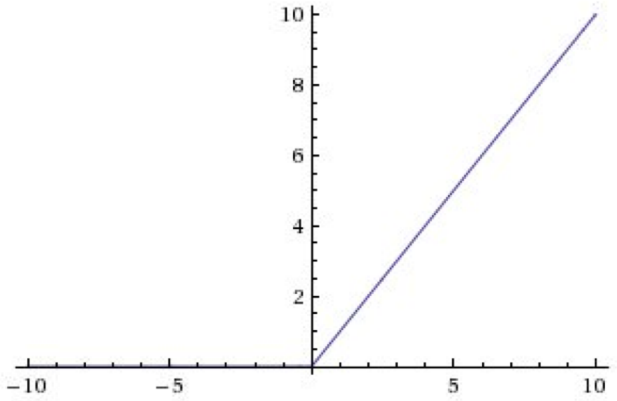
NON-LINEARITY

Non-linear operation on input

POOLING

ReLU (Rectified Linear Unit)

- Capture Interaction
  - E.g Input : 3 * x1 + 4 * x2, Output : f(Input)
- Introduce Non-Linearity
  - Slope is not constant (zero for negative value, 1 for positive)
- Reduce the chances of vanishing gradient
  - Average derivative rarely become 0 (some data points have positive derivative)

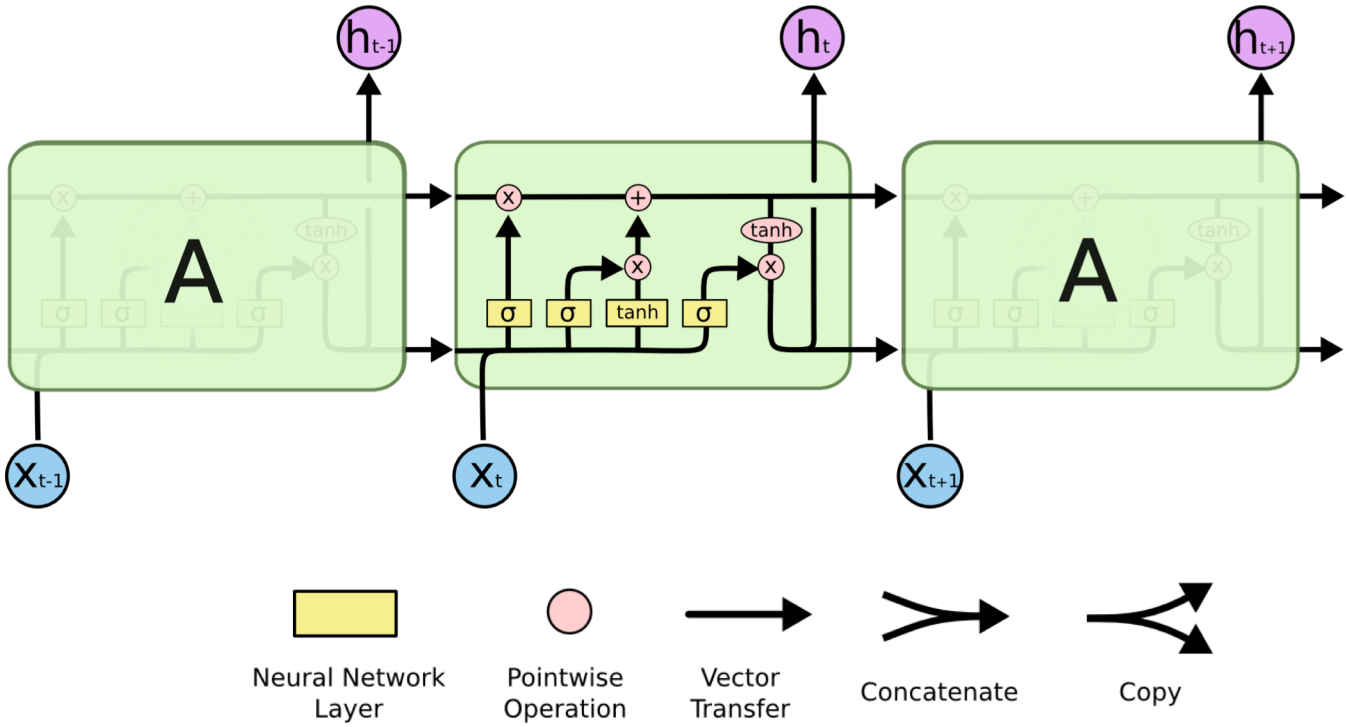


f(x)=max(0,x).

Image Credit : https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Popular Neural Network Architectures : LSTM (Long Short Term Memory)

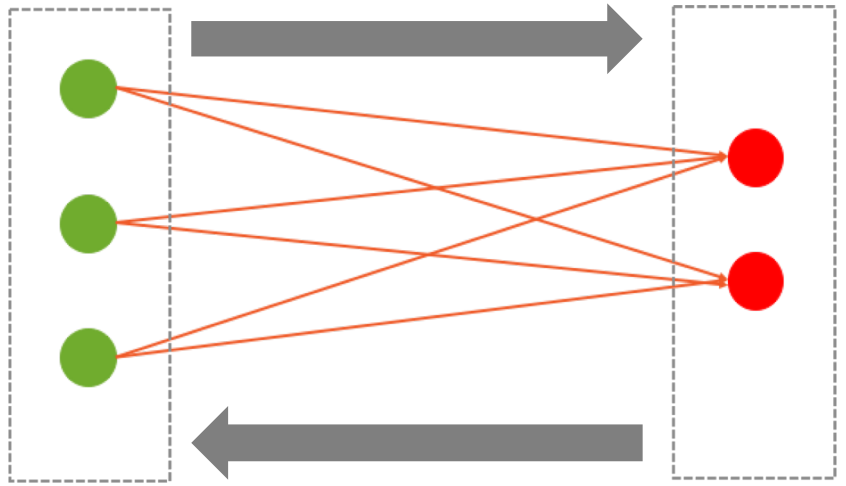
- Special kind of Recurrent Neural Network (RNN)
- Can learn long-term dependencies (as default behavior)
- Use gates
  - Forget gate
  - Input gate
  - Output gate



Source : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

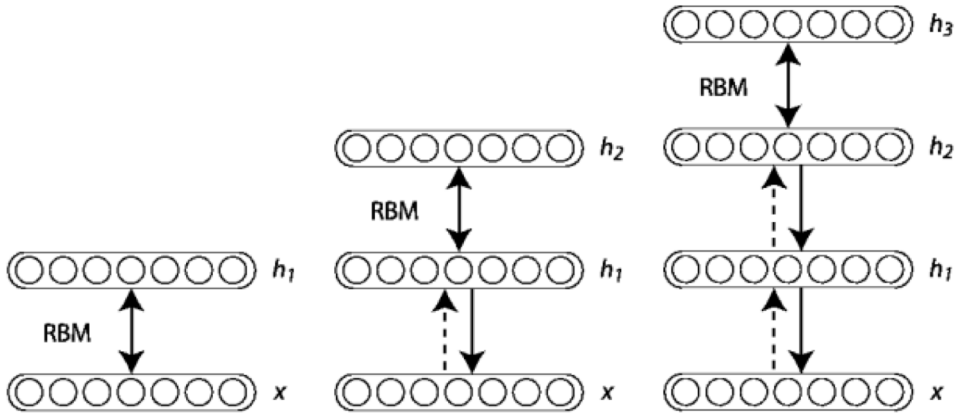
Popular Neural Network Architectures : RBM (Restricted Boltzmann Machine)

- Shallow network
- Can be used for unsupervised learning
- Reconstruct input
- Belongs to auto-encoder family
- Useful in Collaborative filtering, dimensionality reduction



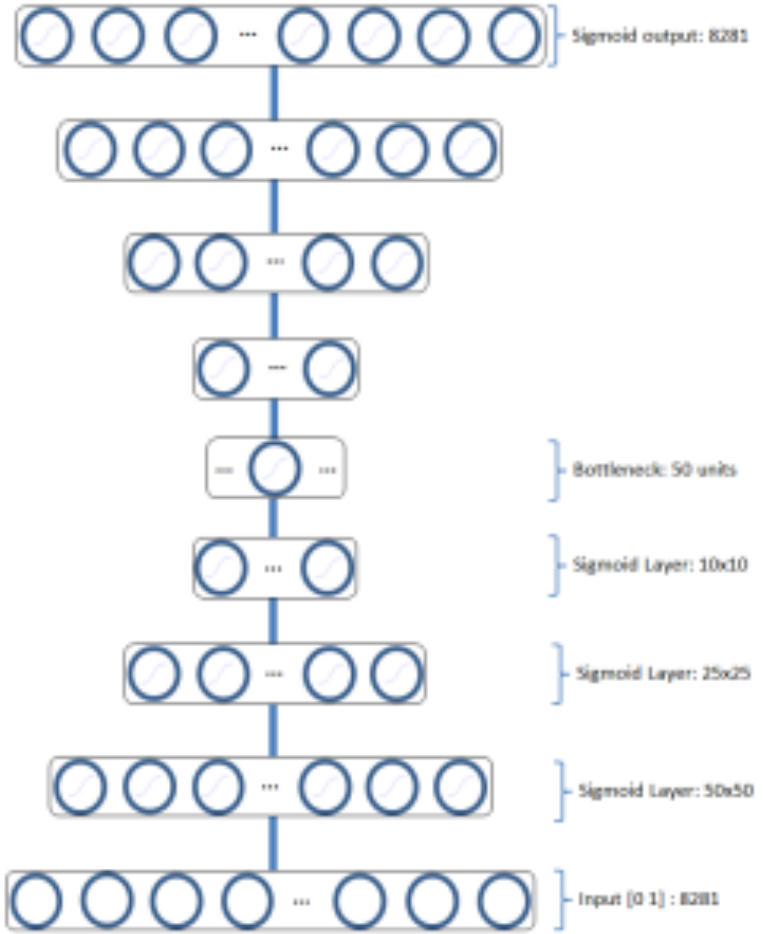
Popular Neural Network Architectures : Deep Belief Networks

- Boltzmann Machine is a specific energy model with linear energy function.
- This is a deep neural network composed of multiple layers of latent variables (hidden units or feature detectors)
- Can be viewed as a stack of RBMs
- Hinton along with his student proposed that these networks can be trained greedily one layer at a time



Popular Neural Network Architectures : Auto-Encoders

- Aim of auto encoders network is to learn a compressed representation for set of data
- Unsupervised learning algorithm that applies back propagation, setting the target values equal to inputs (identity function)
- Denoising auto encoder addresses identity function by randomly corrupting input that the auto encoder must then reconstruct or denoise
- Best applied when there is structure in the data
- Applications : Dimensionality reduction, feature selection



Source : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Why Tensorflow for Deep Learning?

Why Tensorflow for Deep Learning ?

Extensive built-in support

Assemble neural networks

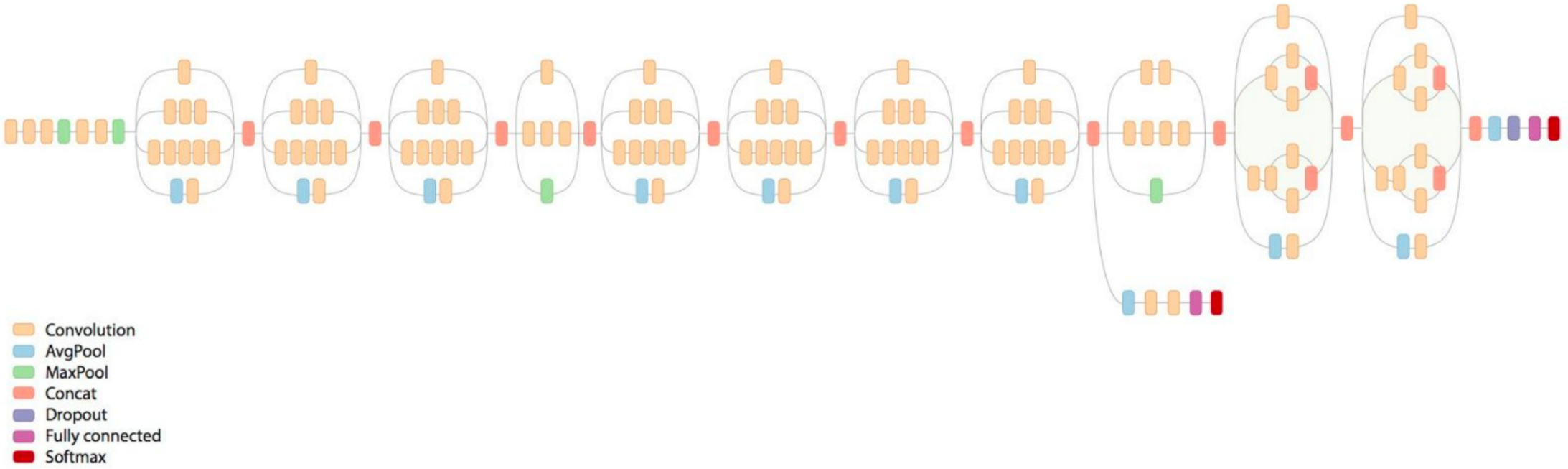
Mathematical functions

Auto-differentiation & first-rate optimizers

Versatility

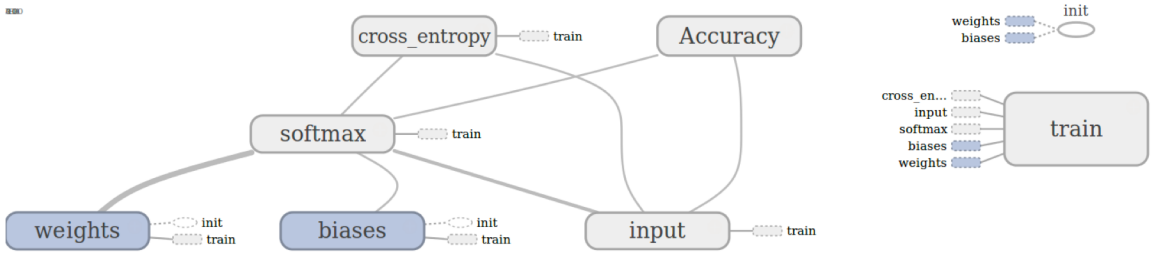
Why Tensorflow for Deep Learning ?

Align cognitive model to programming model



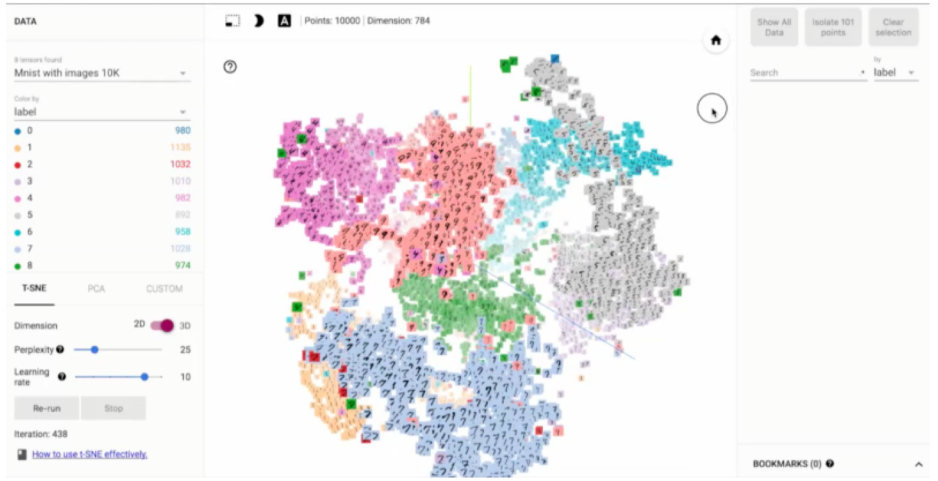
Why Tensorflow for Deep Learning ?

Use Tensorboard to Visualize and Debug Deep Learning Network



Network Graph

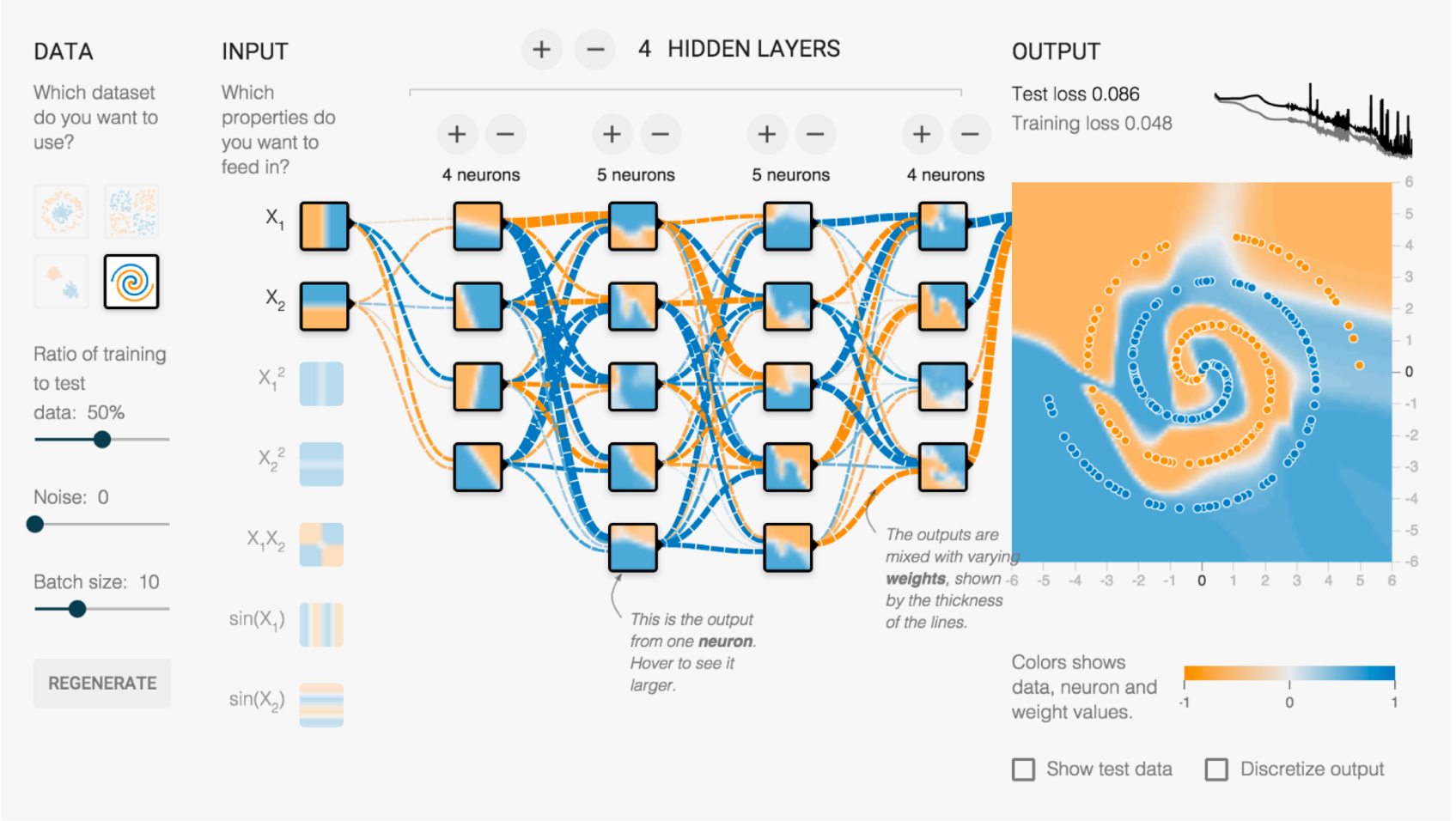
Cost Trends



Visualize Embedding

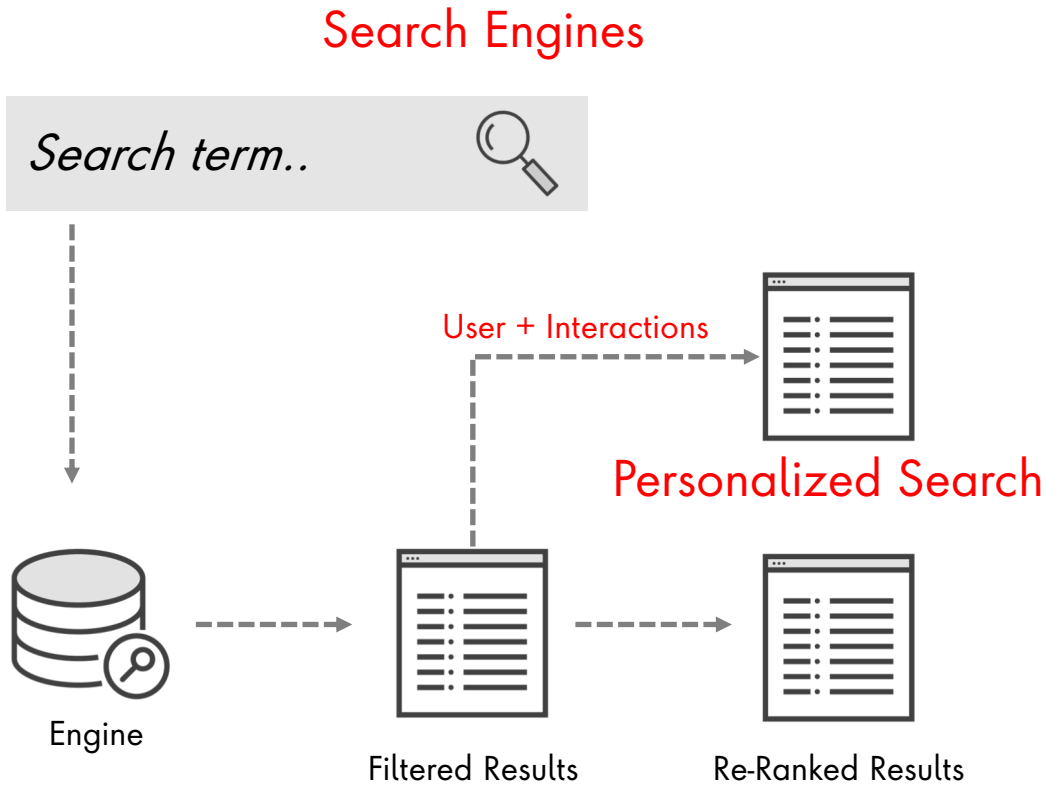
Why Tensorflow for Deep Learning ?

Tensorflow Playbook : http://playground.tensorflow.org/



Deep Learning in Search

Representation : A Key Aspect

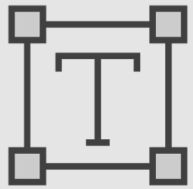


Search or Query	Collection
Word	Set of Documents
Phrase	Set of Documents
Sentence	Set of Documents
Image	Set of Images
Audio	Set of Audio

Representation

Representation

Representation : A Key Challenge



Text

Query	Collection
Word Sentence	Documents

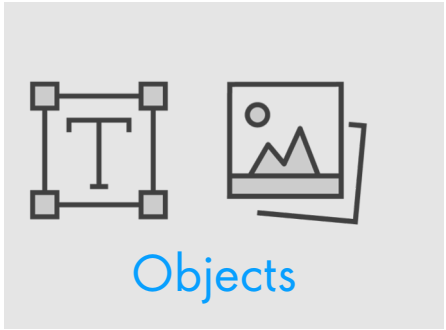
Image

Query	Collection
Image	Set of Images

Treat “Representation” as the problem of “**Embedding**” to encode objects (text, images)
into **continuous space** (set of numeric values)

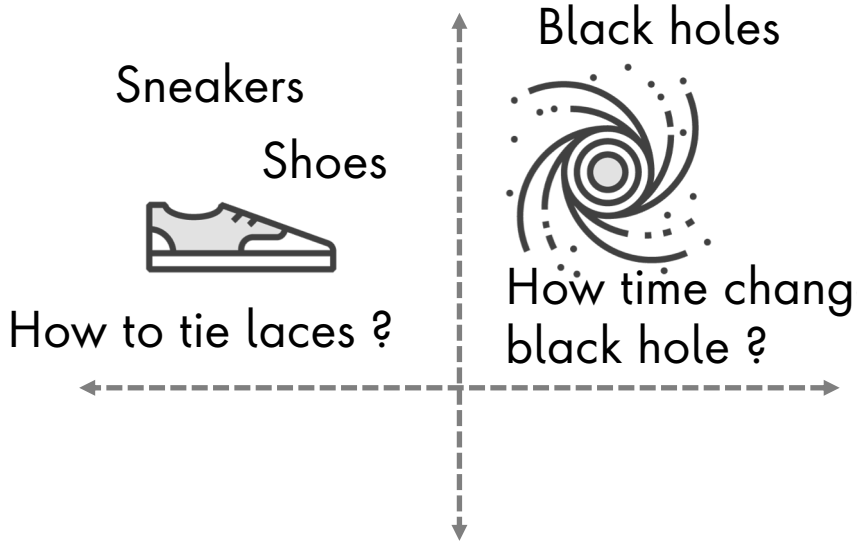
Search Problem In Context of Embedding

1



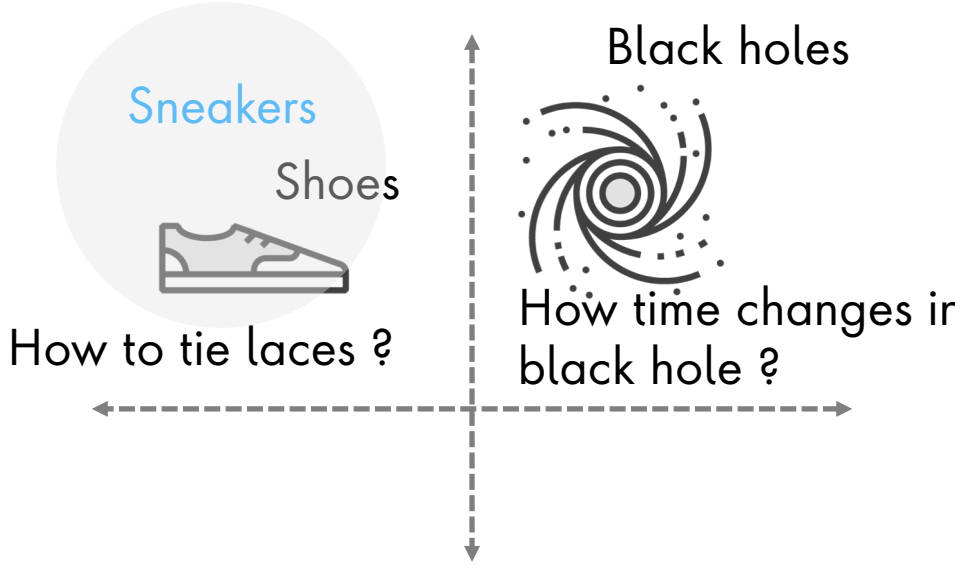
Create embedding for objects into continuous space

2



Put similar objects together Based on embedding

3



Given a query embedding, find neighbors quickly

Word Embedding : One-Hot Encoding

	shoe	sneakers	tree	book	black	...	..	..
Sneakers	0	1	0	0	0	0	0	0
Shoe	1	0	0	0	0	0	0	0
Tree	0	0	1	0	0	0	0	0

Number of columns = Number of unique words in the vocabulary

Issues :

- Sparse (all values are zero except one)
- Large embedding dimension (equal to vocab size)
- Semantic meaning not captured
  - "shoe" is at same distance as "tree"

Word Embedding : Prediction Based Encoding

Sneakers [0.322 0.122 0.231 0.111 0.222 ..... 0.445]
Embedding Size : d

Benefits:

- Dense representation
- Smaller embedding dimension (equal to embedding size : d)
- Semantic meaning captured
  - "shoe" is at smaller distance than "tree"

Word2Vec Model
(Google, 2013)

CBOW

Use surrounding words to predict target word

Skip-Gram

Use target word to predict surrounding words

GloVe Model
(Stanford, 2014)

Use word-word co-occurance matrix and nearest neighbor to create embedding

Demo : Short Introduction to Embedding

Goal :

- Embedding in Tensorflow
- Word Embedding using GloVe pre-trained model

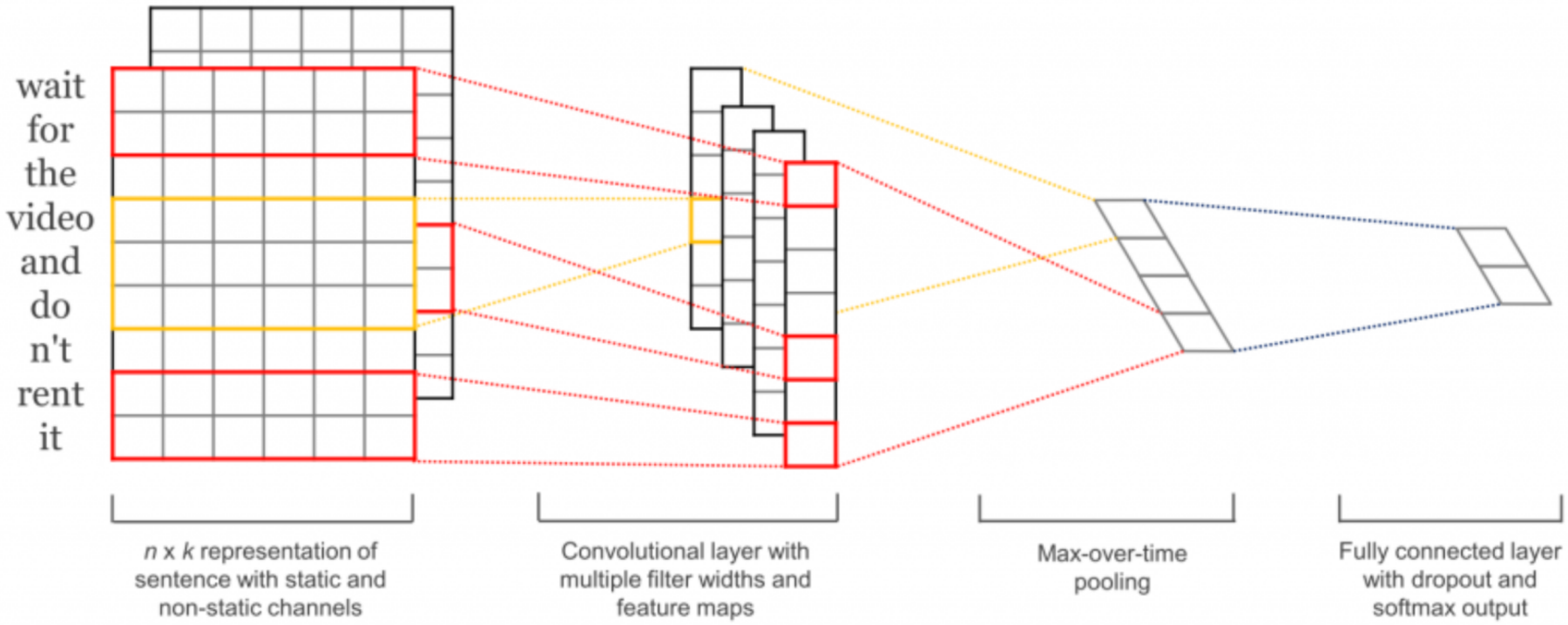
Sentence Embedding

I like white sneakers.

	I	Like	White	Sneakers
Embedding size (d)	..	...	..	..
	x1	x2	x3	x4
	..	..	..	..

- Sentence embedding can be considered as word embedding matrix where each word is represented as a column of size d
- Leverage pre-trained word embedding to learn sentence embedding
- Weights are further tuned during training process

Sentence Embedding Using Convolution Neural Network



Paper : Convolutional Neural Networks for Sentence Classification" by Yoon Kim [Link]

Image Embedding : Option 1 : Flattened Arrays

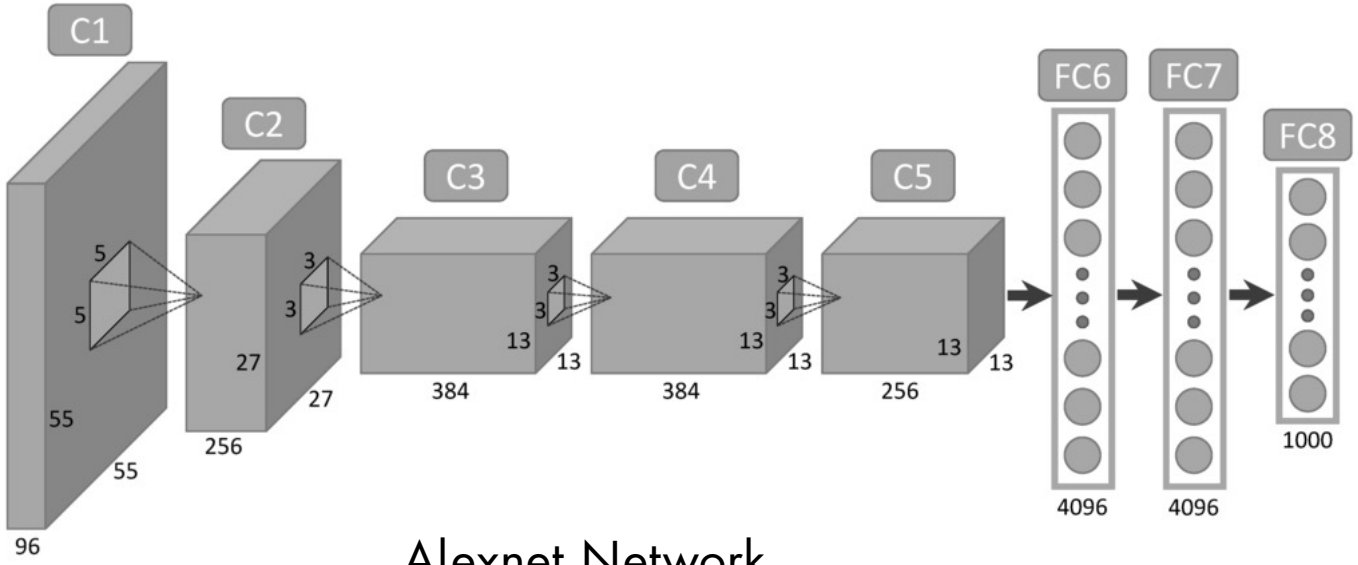
Each image is set of pixels
Flatten the pixels matrix into arrays

100 px by 100 px image : 10000 dimensional array

Issues :

- Very large embedding dimension
- Search in a very large embedding space will be very expensive
- Spatial features (edges, contours, textures) are not captured : Poor search results

Image Embedding : Option 2 : Pre-Trained Deep Learning Models



Alexnet Network

Inception
VGG
Lenet
GoogLenet
ResNet

Benefits:

- Smaller embedding dimension
- Spatial features (edges, contours, textures) are captured in intermediate layers
- Enhanced search experience

Image Source : https://www.saagie.com/blog/object-detection-part1

Demo : Image search using Alexnet Pre-Trained Model

Goal :

- Use Alexnet Pre-trained model to create image embedding
- Image Search

Deep Learning in Recommendation System

RecSys 101 : What is RecSys?

"Serve the relevant items to users in an automated fashion to optimize short and long term business objectives"

RELEVANT (WHAT)

1. Novelty
2. Serendipity
3. Diversity

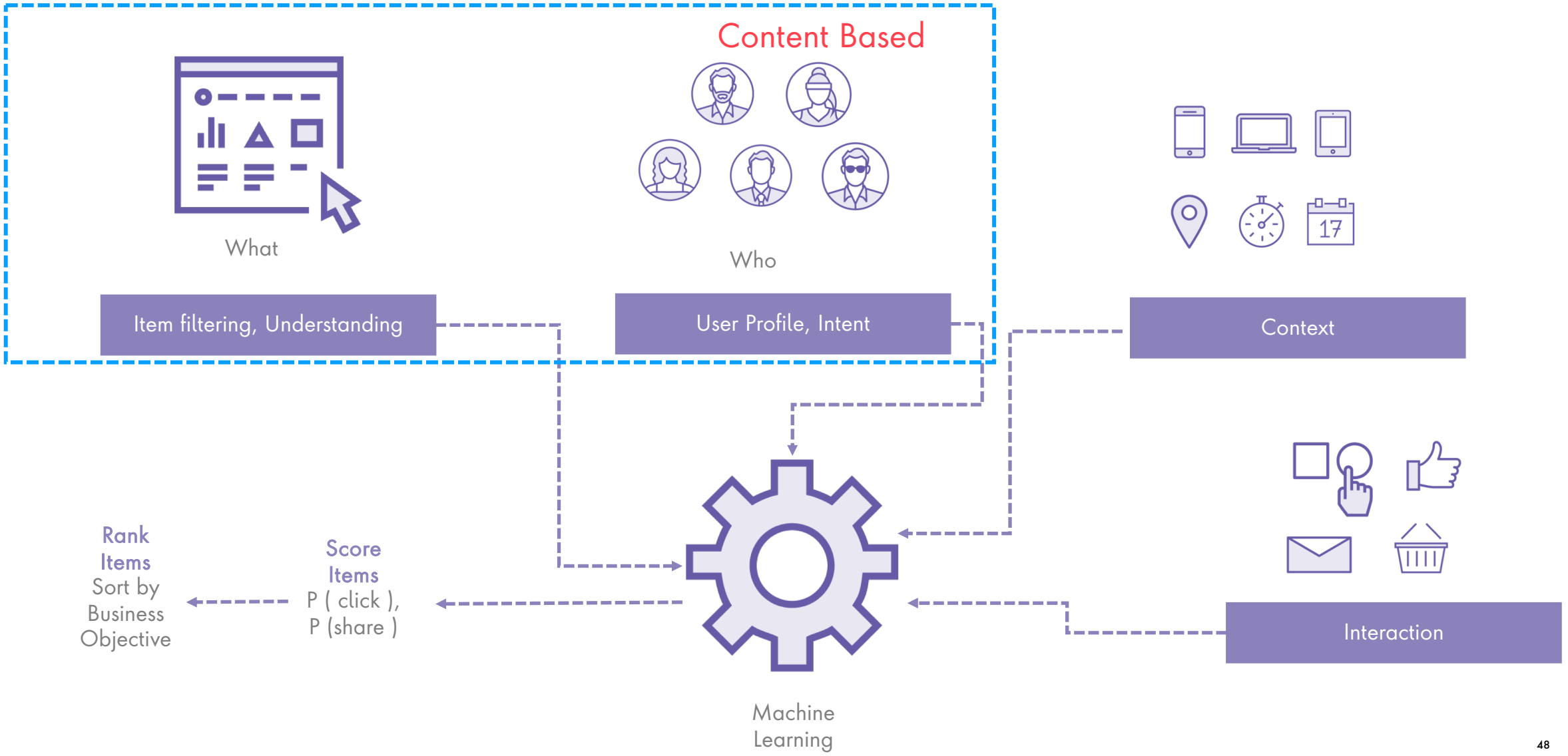
AUTOMATED (HOW)

1. No manual intervention
2. Scale Up

BUSINESS OBJECTIVES (WHY)

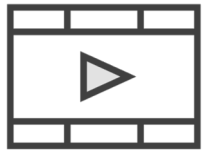
1. Short Term Business Objectives
   a. High clicks
   b. Revenue
   c. Positive explicit ratings
2. Long Term Business Objectives
   a. Increased engagement
   b. Increase in social action
   c. Increase in Subscriptions

RecSys 101 : Internals



RecSys 101 : Content Based Recommendation

Recommends an item to a user based upon a description of the item and a profile of the user's interests



Representing Items using Features

Drama	Arty	Comedy	Action	…	…	…	Commercial
0.7	0	0.2	0				0.8

User Profile

Creating a user profile that describes the types of items the user likes/dislikes

RecSys 101 : Content Based Recommendation

- More than 100 million monthly active users
- Over 30 million songs

Track: May 16

Artist: Lagawagon

Album: Let's Talk About Feelings

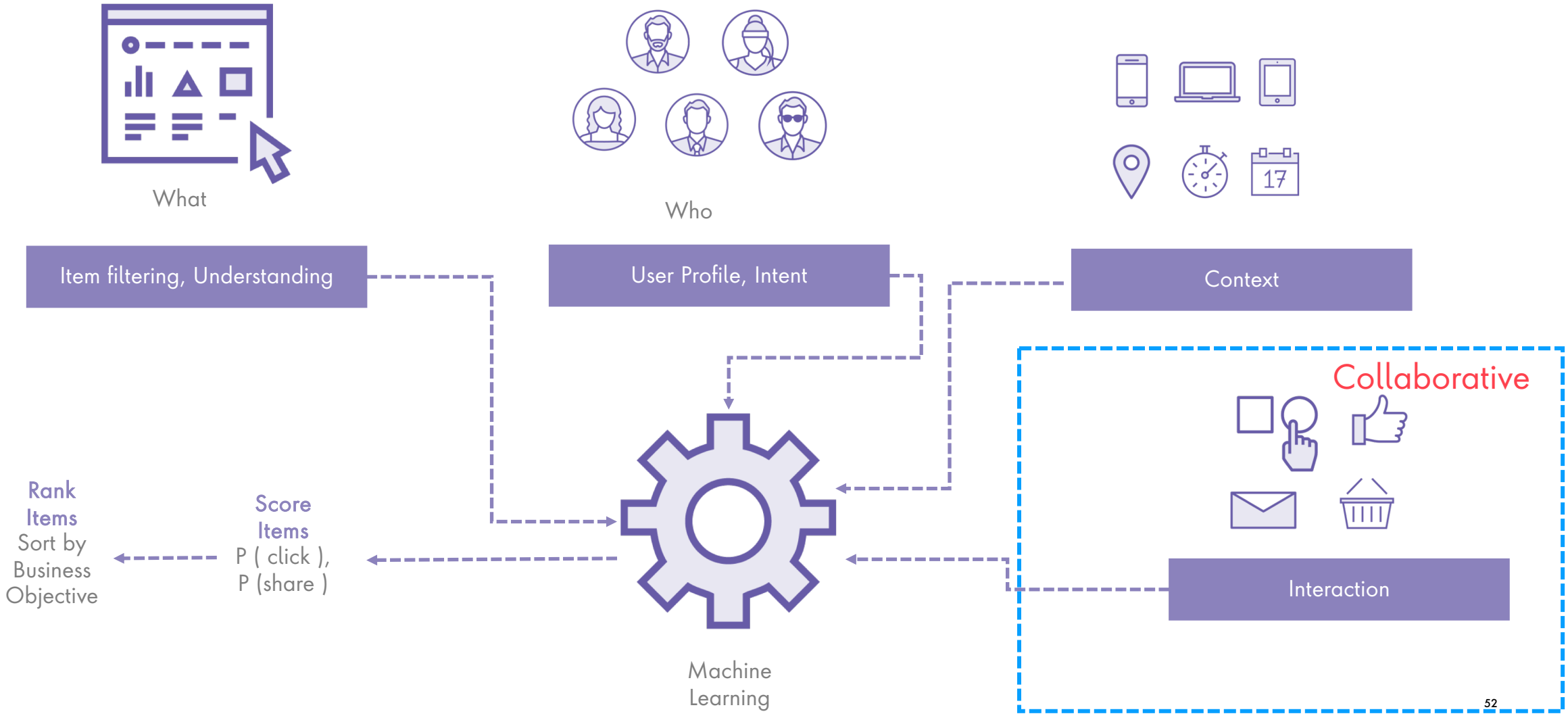
Release: 1998


```
{
  "danceability" : 0.560,
  "energy" : 0.527,
  "key" : 2,
  "loudness" : -9.783,
  "mode" : 1,
  "speechiness" : 0.0374,
  "acousticness" : 0.516,
  "instrumentalness" : 0.0000240,
  "liveness" : 0.156,
  "valence" : 0.336,
  "tempo" : 93.441,
  "type" : "audio_features",
  "id" : "2z7D7kbpRcTvEdT71tdiNQ",
  "uri" : "spotify:track:2z7D7kbp
  "track_href" : "https://api.spo
  "analysis_url" : "http://echone
  "duration_ms" : 168720,
  "time_signature" : 4
}
```

RecSys 101 : Content Based Recommendation

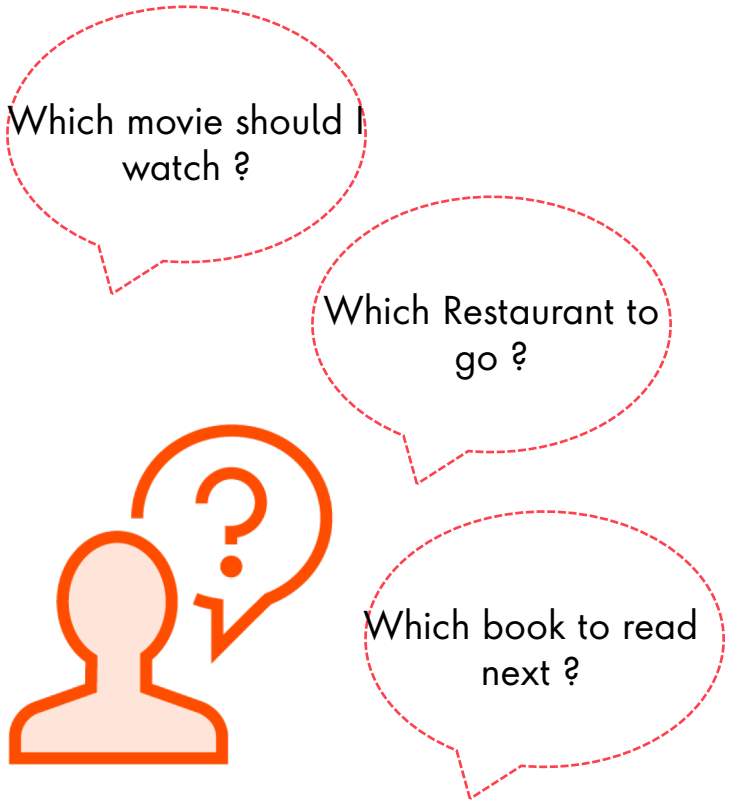
Pros	Cons
No need of other users data	Can only be effective in limited circumstances
Easy to understand reason behind recommendation	No suitable suggestions if content doesn't have enough information
Capable of recommending new and unknown items	Depend entirely on previous selected items and therefore cannot make predictions about future interests of users

RecSys 101 : Internals



RecSys 101 : Collaborative Filtering

Unlike Content based filtering , Collaborative Filtering doesn't require any product description at all



RecSys 101 : Collaborative Filtering : Interactions / Feedback

Explicit

Ratings

Implicit

Purchased

Add to cart

Viewed

Shared

RecSys 101 : Collaborative Filtering : Interactions / Feedback

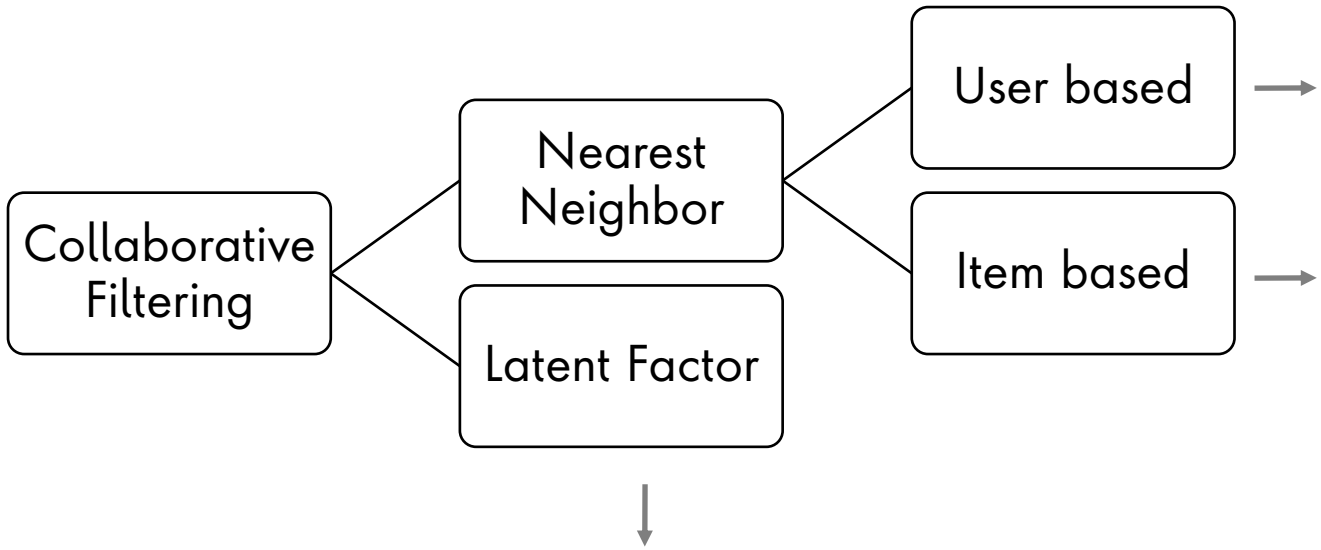
Explicit

- Very few users leave ratings
- Very less explicit data
- Ratings are biased
- Often not easy for user to express likeness in terms of Ratings or score

Implicit

- Easy to track & Store web logs data
- Lots of implicit data generated for each user
- More the data , better the recommendations
- Noisy
- Difficult to infer Negative Feedback

RecSys 101 : Collaborative Filtering



Find similar users like me and recommend what they liked

Find similar items to those I have previously liked

Factor based techniques (Matrix Factorization, Factorization Machine)

$ 1 Million Netflix Challenge

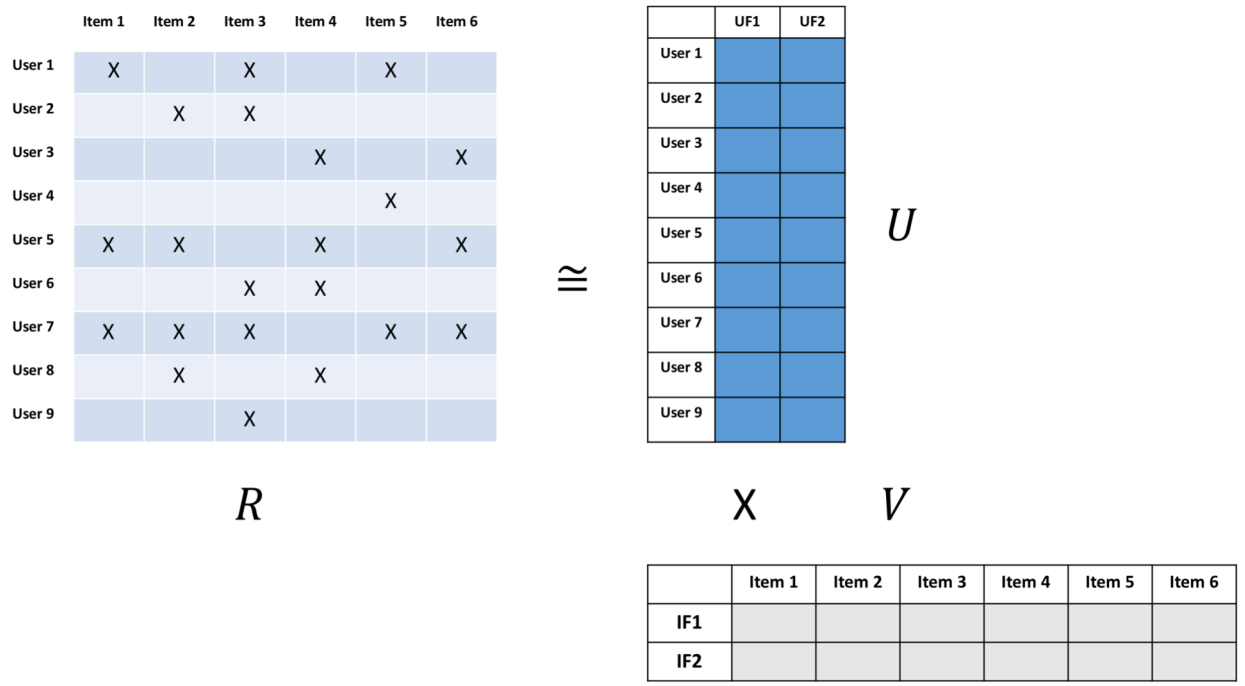
- Scalability
- Predictive accuracy
- Can model real-life situations (e.g. Biases, Additional Input sources , Temporal Dynamics)

RecSys 101 : Collaborative Filtering : Latent Factor

Take the users and their feedback for different items and identify hidden factors that influence the user feedback

The idea is to factorize or decompose the user item matrix into two matrices

- Users are mapped on to hidden factors
- Items are mapped on to hidden factors



RecSys 101 : Collaborative Filtering

Pros

Content information not required either of users or items

Personalized recommendations using other user's experience

No domain experience required

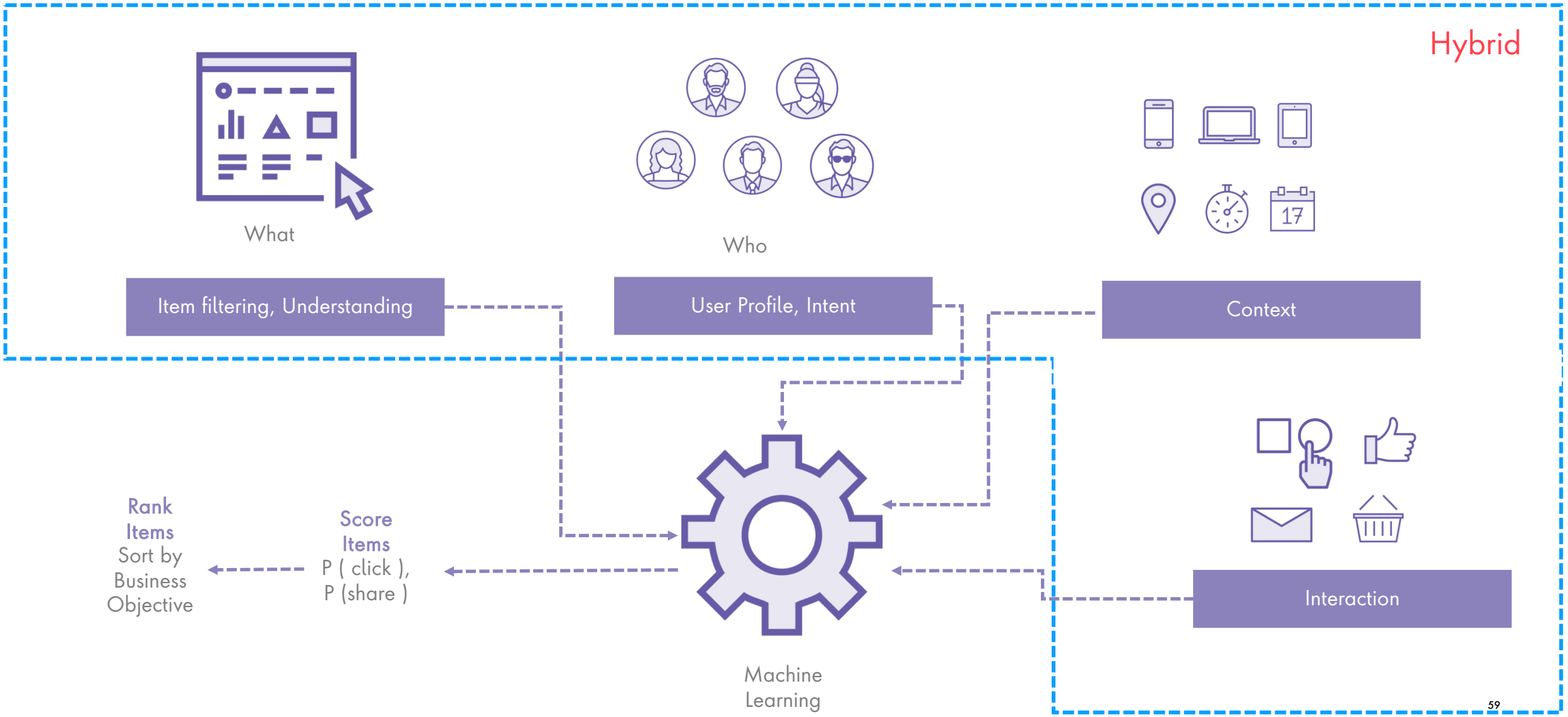
Cons

Cannot produce recommendations if there is no interaction data available (Cold Start Problem)

Often demonstrate poor accuracy when there is little data about users' ratings (Sparsity)

Popular items get more feedback (Popularity bias)

RecSys 101 : Internals



RecSys 101 : Hybrid Recommendation Engine

Pros

Solve the issue of Cold Start by leverage both content and collaboration

Use of Implicit feedback reduces the sparsity issues to a large extent

Can include higher order feature interactions as well

Cons

Difficult to implement

Representation : A Key Aspect

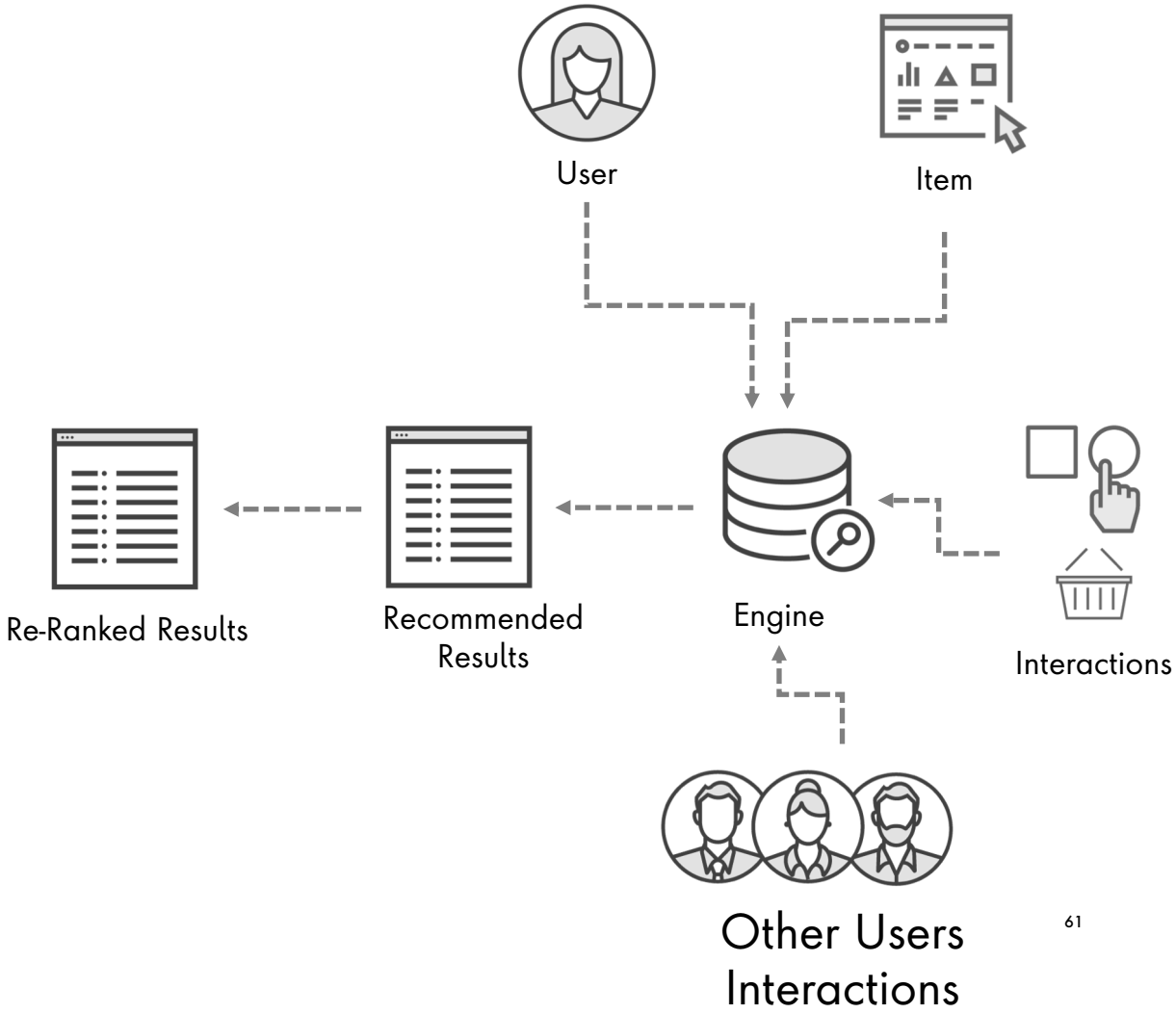
User ID

Item ID

User / Item Metadata

Representation

Recommendation Engines



Matrix Factorization

Items

Users

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	X		X		X	
User 2		X	X			
User 3				X		X
User 4					X	
User 5	X	X		X		X
User 6			X	X		
User 7	X	X	X		X	X
User 8		X		X		
User 9			X			

R

$\cong$

	UF1	UF2
User 1		
User 2		
User 3		
User 4		
User 5		
User 6		
User 7		
User 8		
User 9		

U

X

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
IF1						
IF2						

V

How to better represent users and items ?

What about item and user metadata ?

Matrix Factorization

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	X		X		X	
User 2		X	X			
User 3				X		X
User 4					X	
User 5	X	X		X		X
User 6			X	X		
User 7	X	X	X		X	X
User 8		X		X		
User 9			X			

R

$\cong$

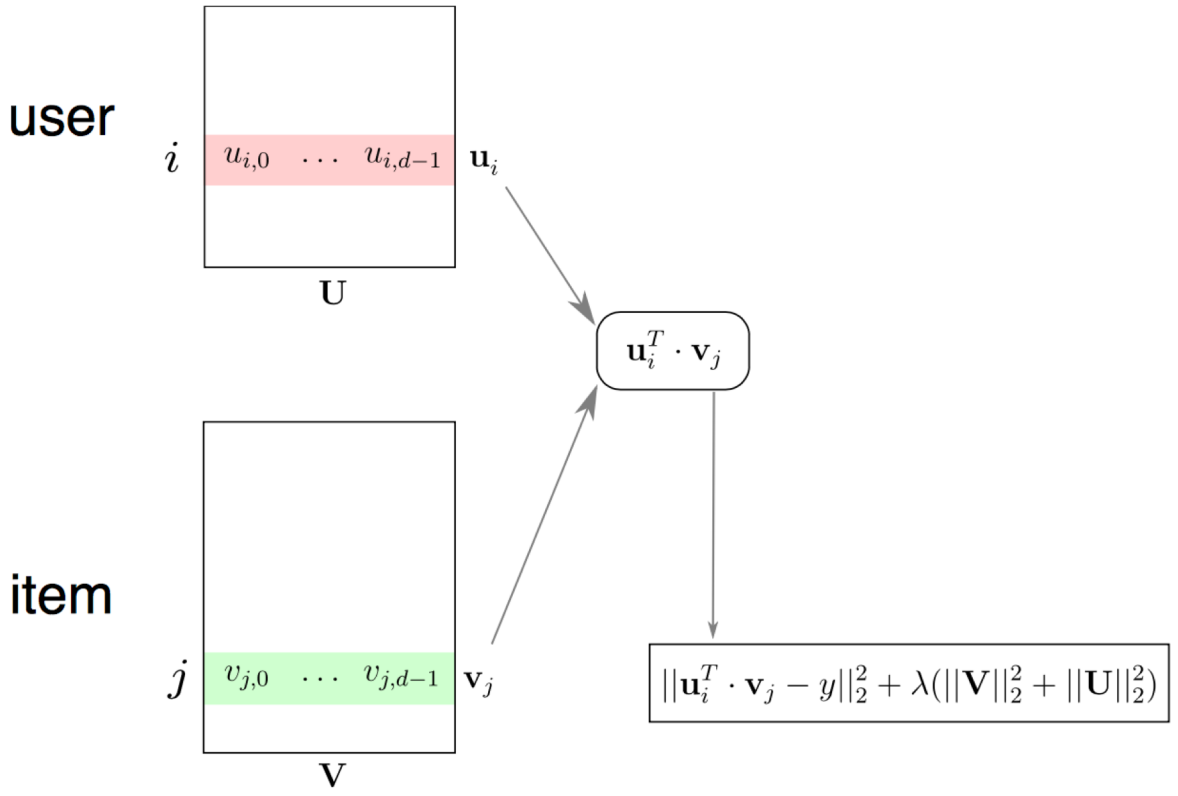
	UF1	UF2
User 1		
User 2		
User 3		
User 4		
User 5		
User 6		
User 7		
User 8		
User 9		

U

X

V

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
IF1						
IF2						



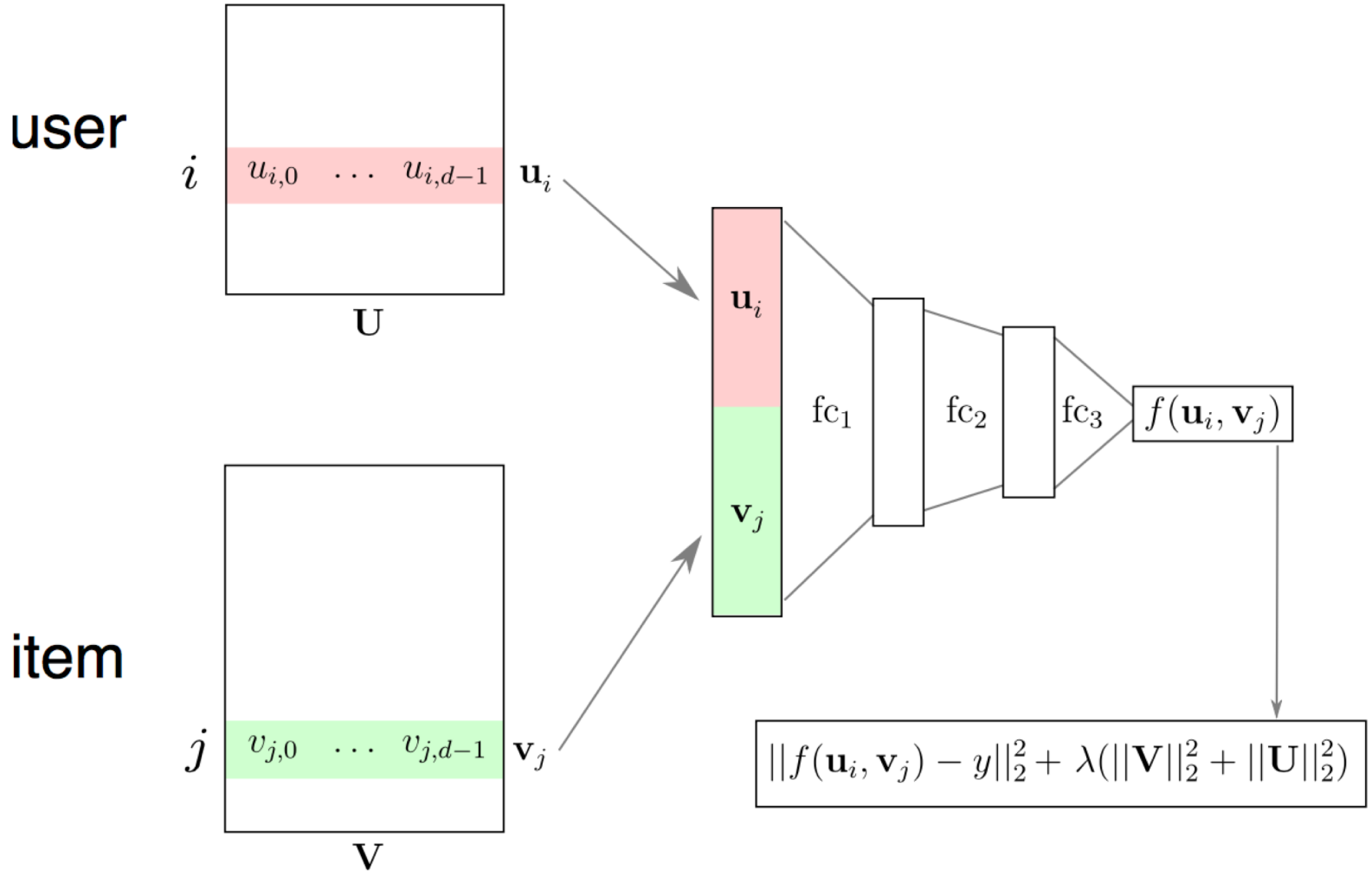
Credit : Olivier Grisel

Demo : User and Item Embedding in Matrix Factorization using Tensorflow

Goal :

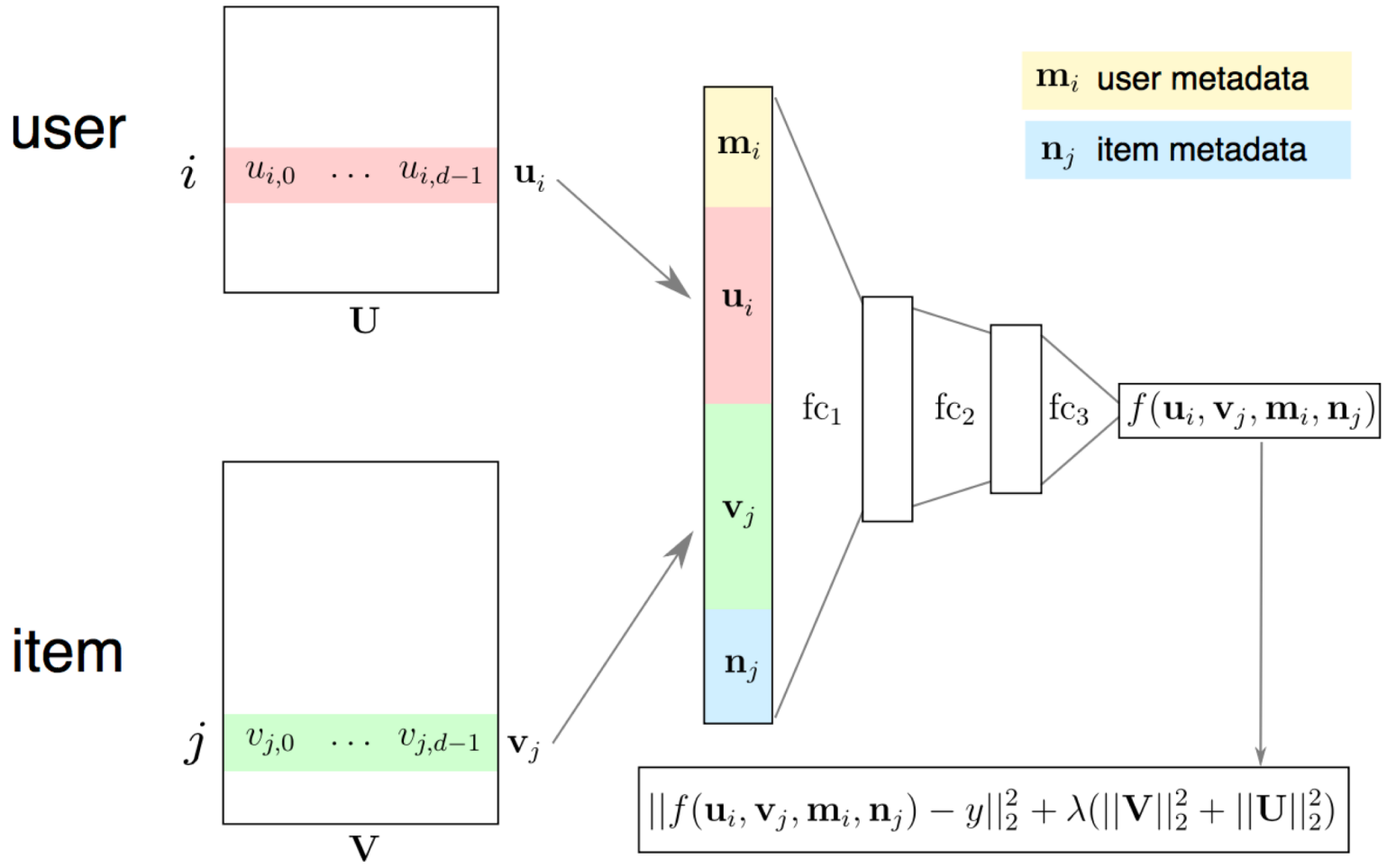
- Use embedding in RecSys
- How to add metadata for building hybrid RecSys

Matrix Factorization : Deep Neural Networks



Credit : Olivier Grisel

Matrix Factorization : Deep Neural Networks with Metadata

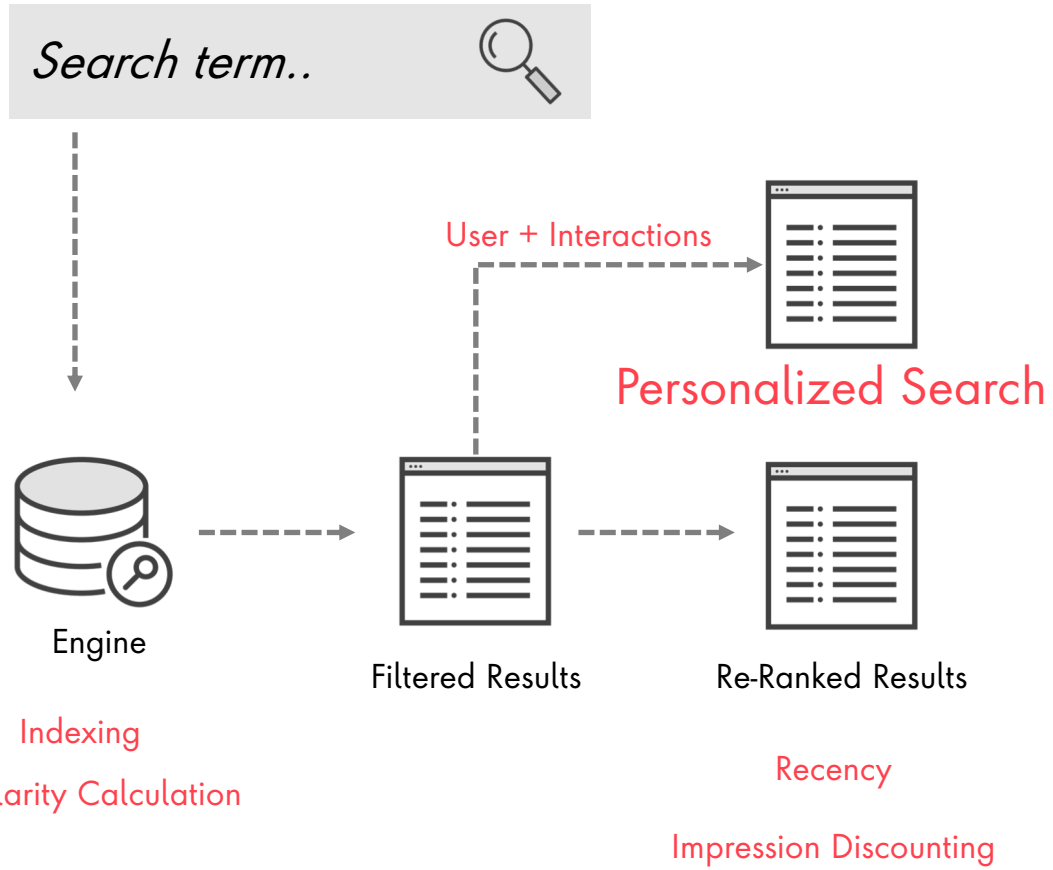


Credit : Olivier Grisel

Learning to Rank

Problem Space : Re-Ranking

Search Engines



Search
+
Re-Ranking Using Machine Learning
=
Learning to Rank

Why Learning to Rank is Required ?

Classical Information Retrieval

- Inputs
  - Query, Document
    - TF, IDF , |D|, P(t|D)
- Output
  - Relevance Score (q, d)

- Manual function : VSM Cosine , BM25, LM Dirichlet

- Not many factors to tune

Learning to Rank

- To improve search results quality need to consider many features
  - Query word in anchor text
    Number of images
  - Number of out links
  - Page rank

- Classical IR don't work as many factors and their combinations have to be tuned

- Machine learning based ranking system learn a function to automatically rank results

Learning to Rank Formulation

Query q_i

Documents $D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$

Relevance [0 , 1] $y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

Learn a ranking function H

H (w, func(q_i, D_i)) → Optimal Ranking R_i

Query + Documents Representation

Re-ranked Documents based on predicted relevance

Learning to Rank Techniques : Point Wise Approach

Query q_i

Documents $D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$

Relevance [0 , 1] $y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

H (w, func(q_i, D_i)) → Optimal Ranking R_i

Triples

Input (Query, Document Pair Representation)	Output Class
q_1 , d_{11},	y_{11}
q_1 , d_{12},	y_{12}
—	
q_2 , d_{21} ,	y_{21}
q_2 , d_{22},	y_{22}

Binary Classification

Use Probability as Relevance Score

Learning to Rank Techniques : Pair Wise Approach

Query q_i

Documents $D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$

Relevance [0 , 1] $y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

$H (w, func(q_i, D_i)) \rightarrow$ Optimal Ranking R_i

Triples

q_1 , d_i (Pos), d_j (Neg)
q_2 , d_i (Pos), d_j (Neg)

$H (w, func(q_i, D_{i(pos)}) >= H (w, func(q_i, D_{i(neg)}) + margin$

There is also a "List Wise Approach" that ranks all documents in one go. But complicated to implement

Learning to Rank In Recommendation System

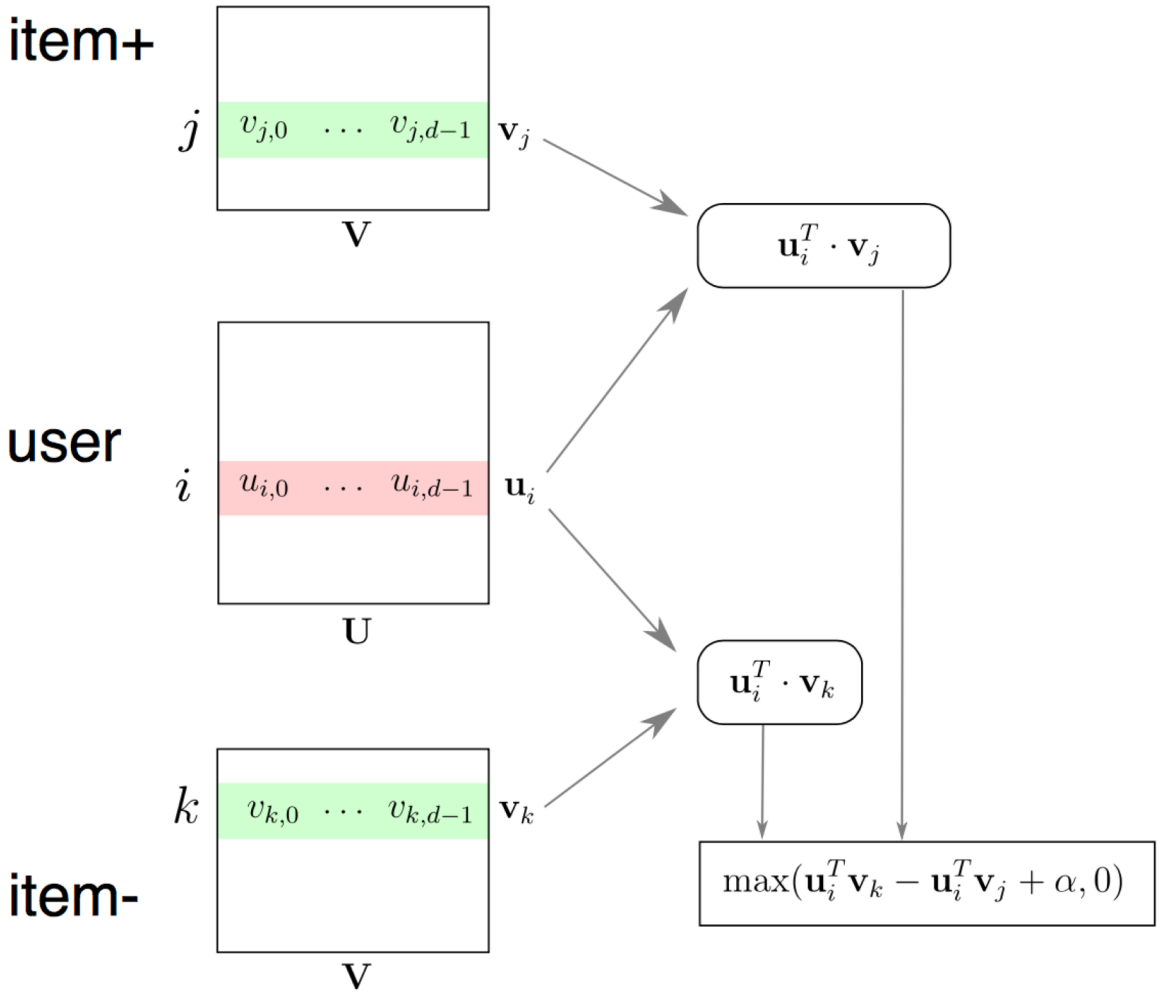
Query	q_i
Documents	$D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$
Relevance [0 , 1]	$y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

$H (w, func(q_i, D_i)) \rightarrow$ Optimal Ranking R_i

User	Q_i
Items	$D_i = d_{i1}, d_{i2}, d_{i3} \ldots d_{in}$
Implicit Feedback [0 , 1]	$y_{i1}, y_{i2}, y_{i3} \ldots y_{in}$

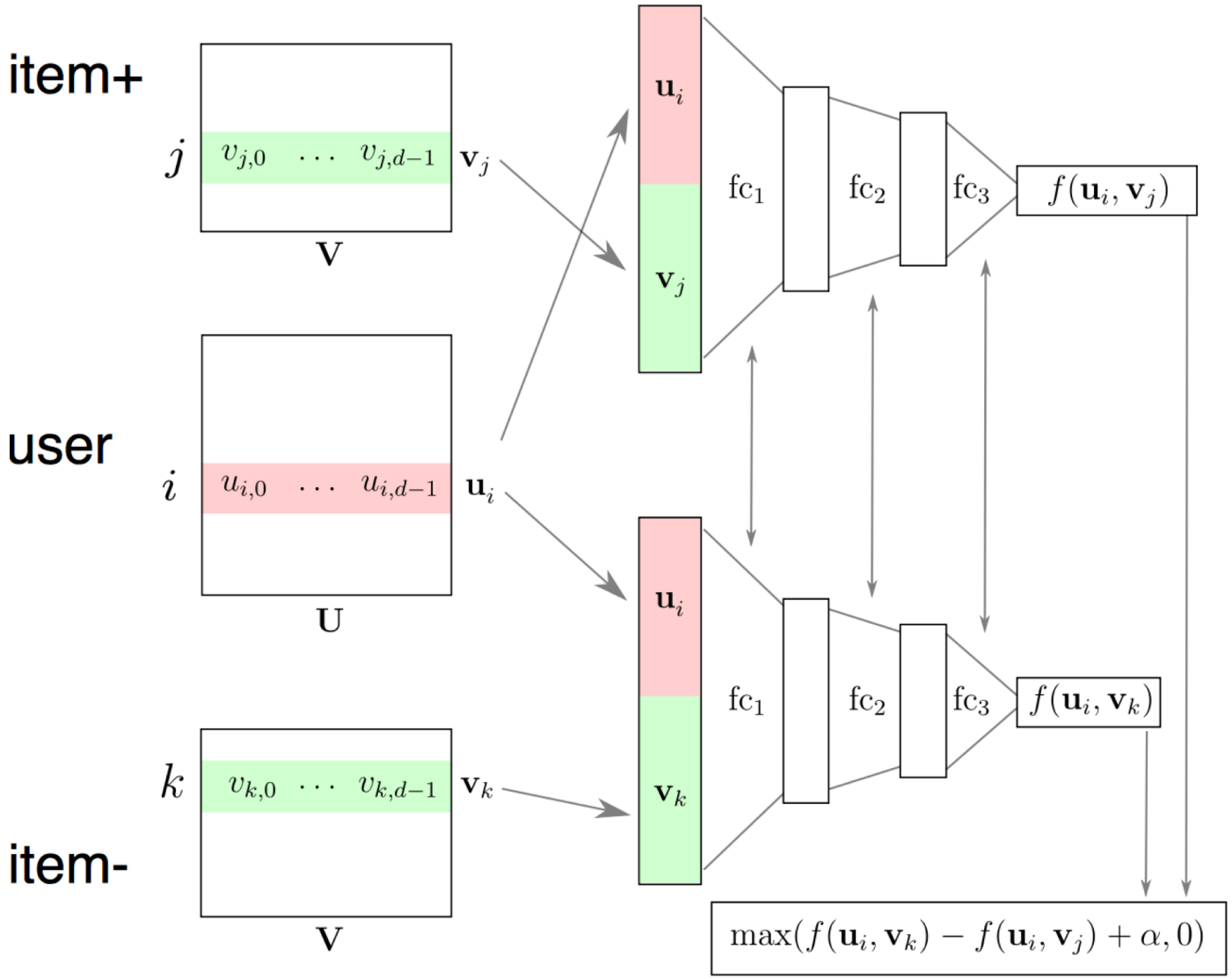
$H (w, func(q_i, D_i)) \rightarrow$ Optimal Ranking R_i

Triplet Loss with Implicit Feedback



Credit : Olivier Grisel

Deep Triplet Network with Implicit Feedback



Credit : Olivier Grisel

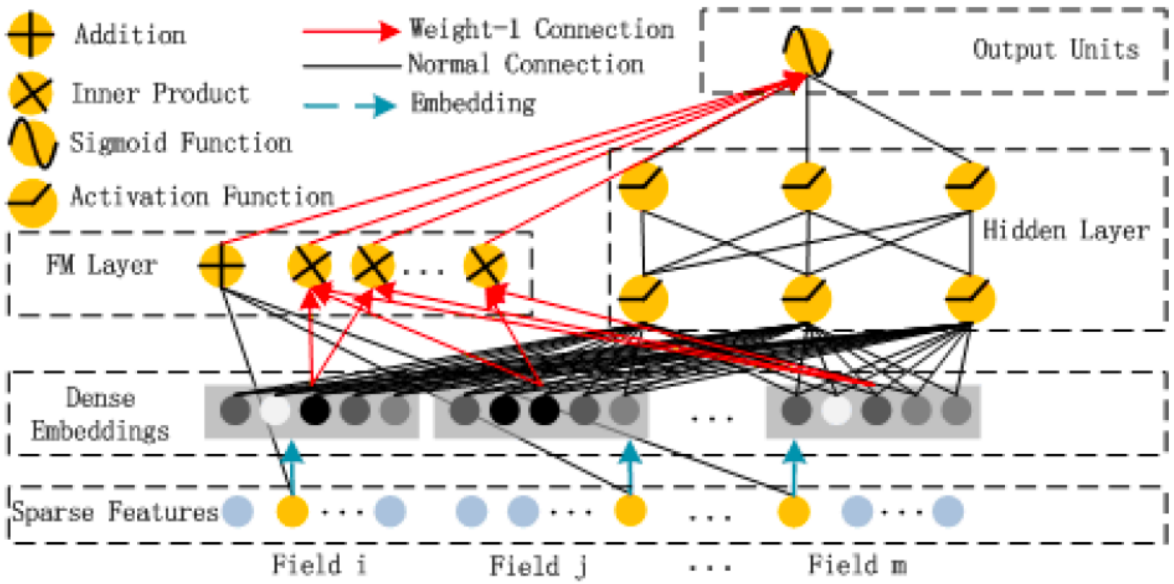
Demo : Using Triplet Loss for Implicit Feedback using Tensorflow

Goal :

- Use implicit feedback
- Use triplet network

Popular Alternatives

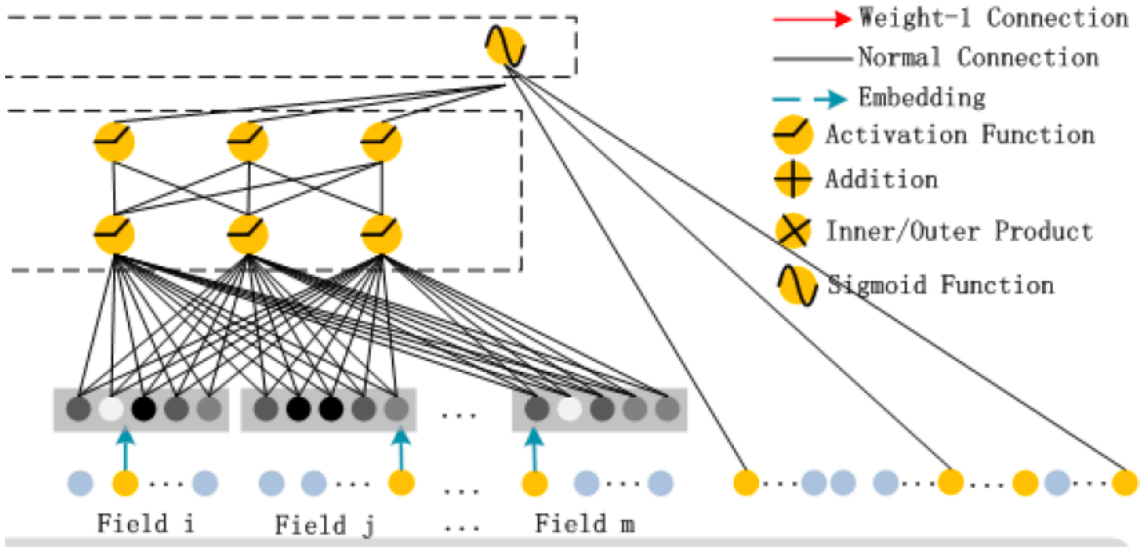
Deep FM



$$y_{FM} = \langle w, x \rangle + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} \langle V_i, V_j \rangle x_{j_1} \cdot x_{j_2},$$

$$\hat{y} = sigmoid(y_{FM} + y_{DNN}),$$

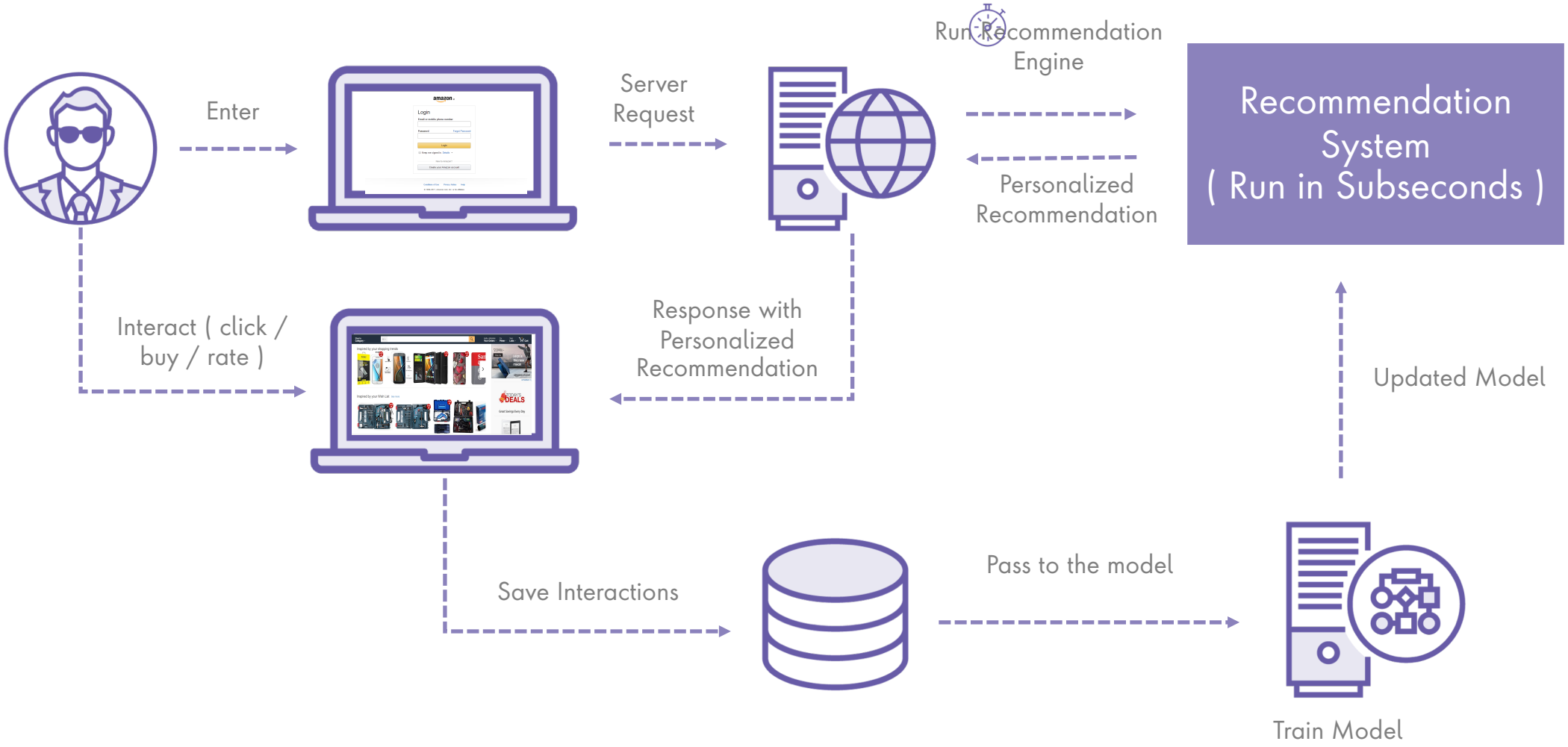
Wide & Deep (Google)



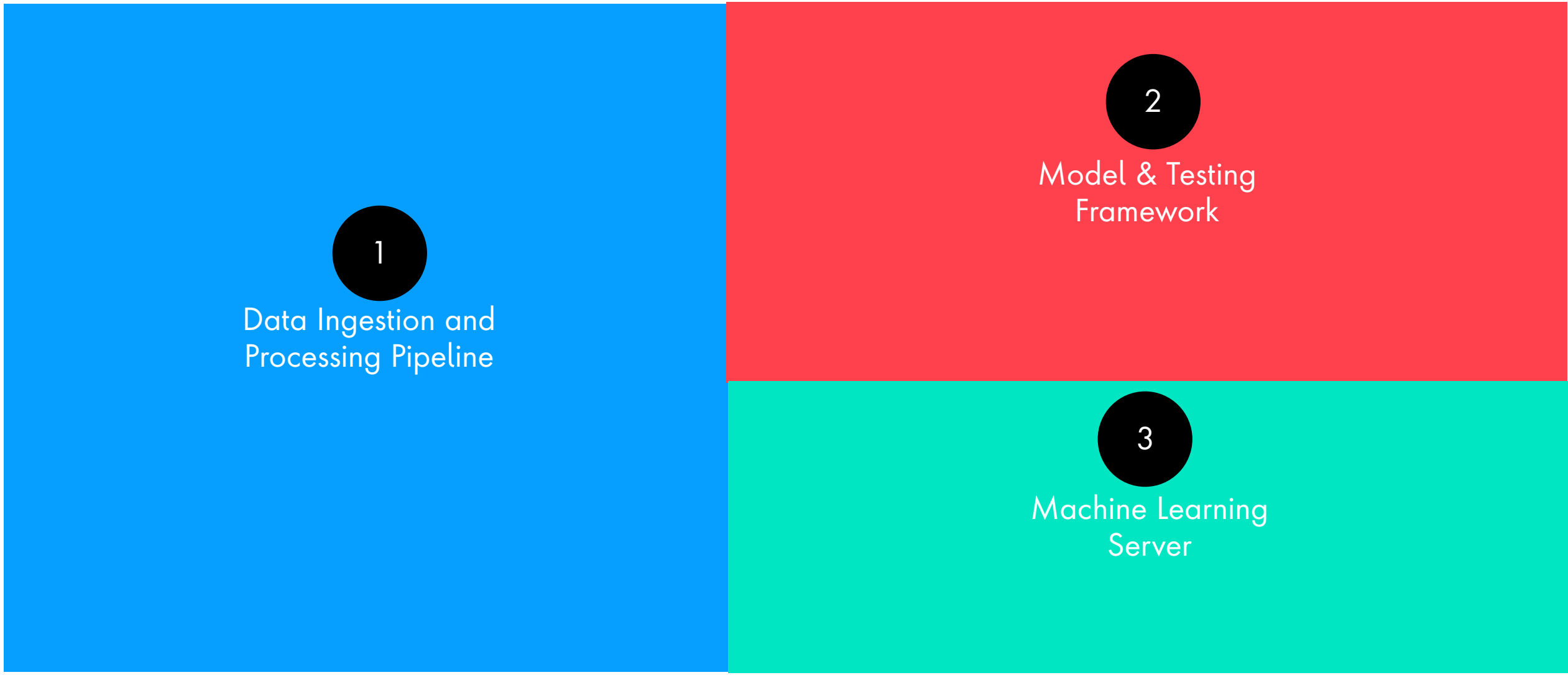
Paper : DeepFM: A Factorization-Machine based Neural Network for CTR Prediction (Link)

Recommendation Engines in Production

How Recommendation System Works



Machine Intelligence Architecture



Tensorflow in Production

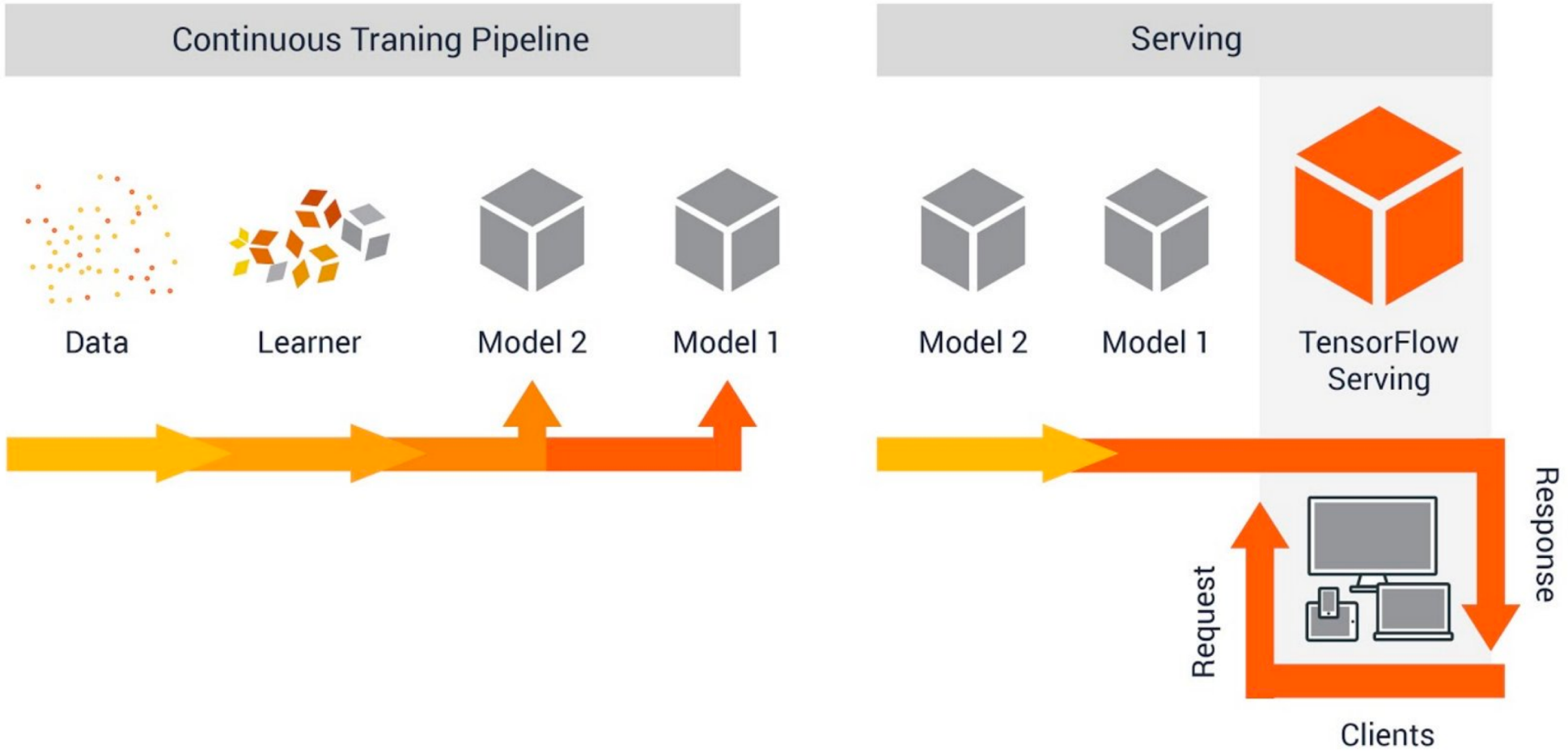
Training

- Distributed training
  - Distributed Tensorflow
  - Leverage Kubernetes to auto-scale training process
  - Tensorflow + Spark to get the best of both world
  - GCP Cloud ML for serverless processing
- HyperParameter Tuning
  - Kubernetes for parallel execution of hyperparameter tuning
  - Leverage Bayesian Optimization (Scikit-Optimize,)
- Specialized Hardware
  - Leverage GPUs, TPUs for faster training
  - Leverage Estimator and Experiment APIs

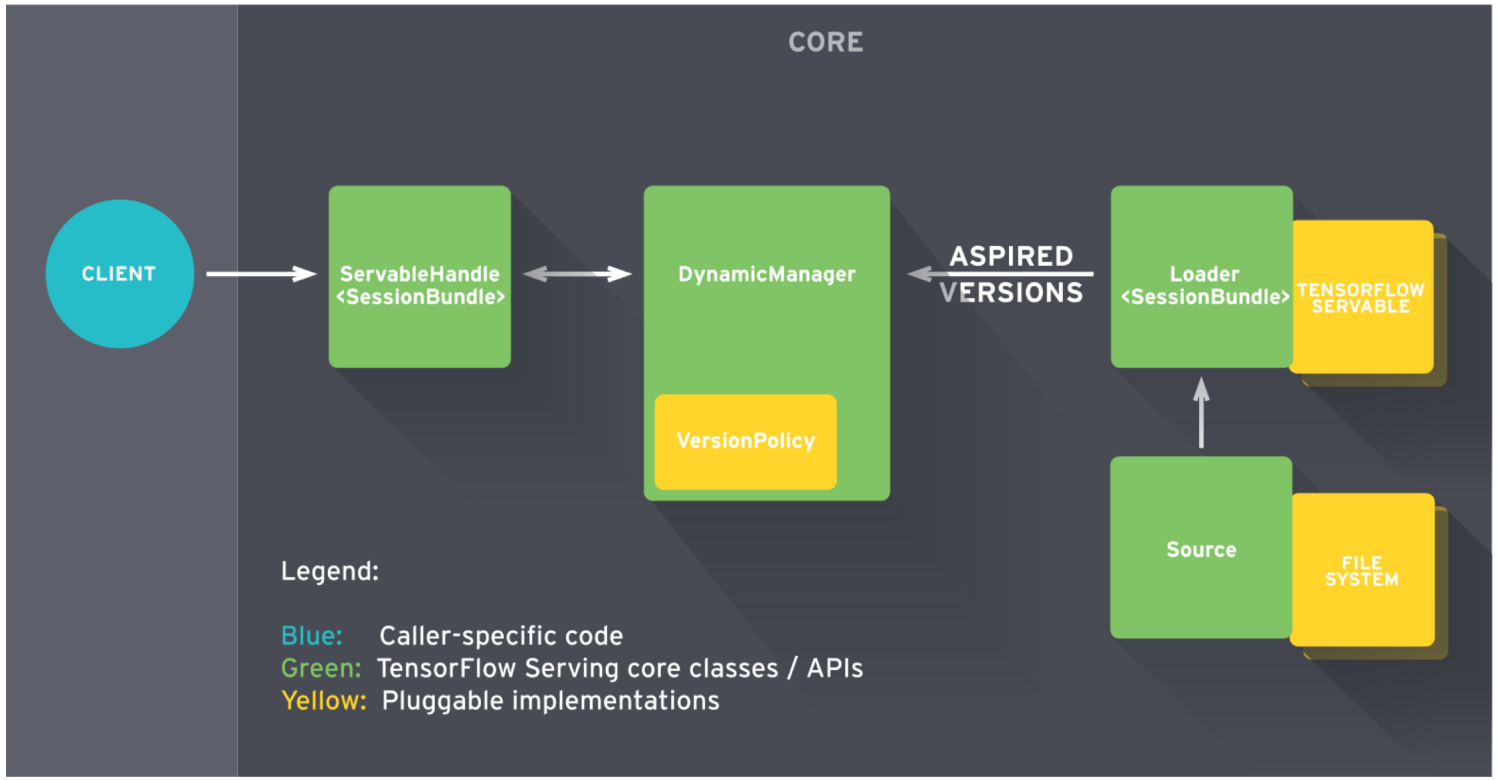
Inference

- Model Object Optimization
  - Graph Transformation Tool (GTT) to remove unreachable nodes, fuse adjacent operators , round weights for compression
  - Ahead of Time (AOT) compiler built on XLA (Accelerated Linear Algebra) for minimal Tensorflow Runtime
- Low latency inference
  - Tensorflow Serving to serve TF models

Tensorflow Serving



Tensorflow Serving



- Low latency inference
- Model versioning and rollback
- Custom version policy for A/B and Bandit tests
- Uses highly efficient gRPC and Protocol Buffers

Next Steps

- Provide feedback on the session
- Session Content
  - http://bit.ly/iith_ps
- Share
  - Progress, Issues, Use-cases
  - Connect on LinkedIn
  - Twitter
    - **@meabhishekkumar**
    - #PSAtIIITH
      @PubSapientIndia

thank you