

Applied Machine Learning for Business Analytics

Lecture 5: Auto-encoders

Logistics

- Some project proposals from the past two years are shared in Luminus.
 - https://bt5153msba.github.io/project/BT5153_ProjectGuidelines_Grading%20Criteria.pdf
- Thanks Groups 15 and 17 for adding new teammates in.
- Appreciate if you keeps video on!

Agenda

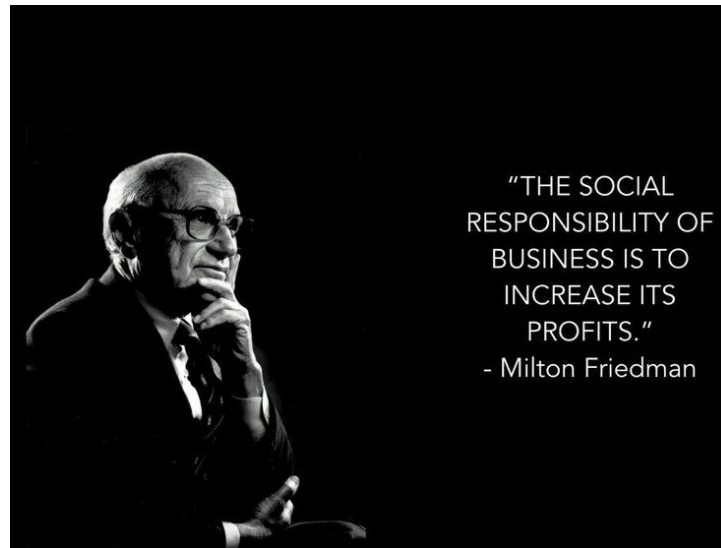
1. Project Scoping
2. Autoencoders
3. Applications of Autoencoders
4. Recommendation Systems

1. Project Scoping

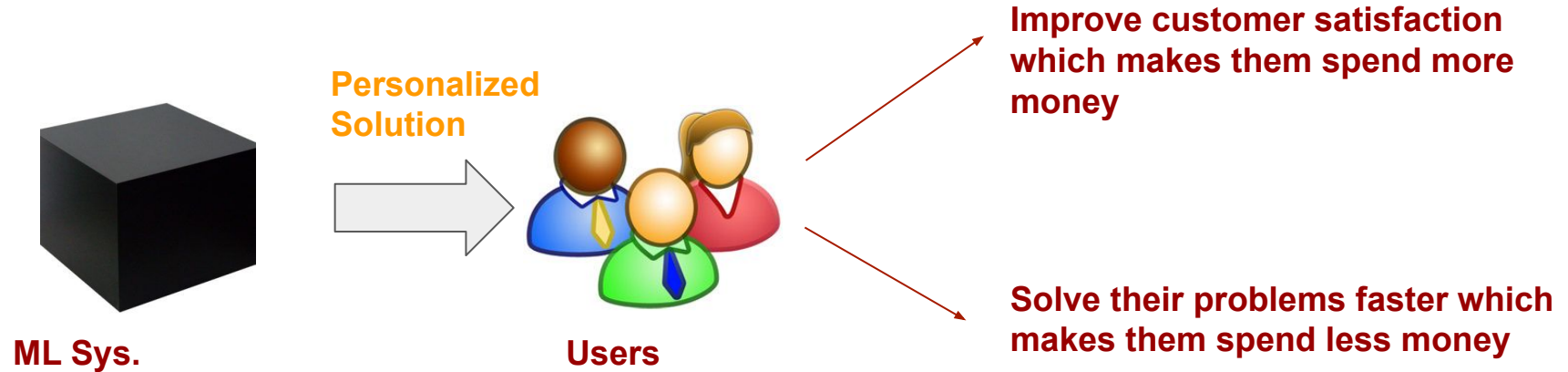
Goals of ML Projects

- An ML project should be aimed at increasing profits directly or indirectly.
 - Increasing sales
 - Cutting costs
 - Increasing satisfaction
 - Increasing time spent on a website
- Do we have non-profits projects? Yes
 - Climate change
 - Public health
 - Education

Connect business metrics to your machine learning models




Case Study



Case Study: Movie Recommendation

- When building a recommendation system for movie
 - Maximize Engagement
 - Maximize Revenue from sponsored content
 - Click more, ads fee more
 - Minimize the spread of restricted content

How to set goals?

- Goals: General Purpose of a Project
 - Maximize users' engagement while minimizing the spread of violent content and maximize revenue from sponsored content
 - Objectives: Specific steps on how to achieve the above goals
 - Filter out unclassified movies
 - Rank movies by quality
 - Rank movies by their ads fee
 - Rank movies by engagement: how likely users will watch it
- 

**How to combine
these two
targets via ML
systems?**

Multi-objective System

- Rank Movies by quality
 - Predict films' rating
 - Minimize Rating_loss: loss between predicted rating and true rating
- Rank movies by engagement: how likely users will watch it
 - Predict watch times
 - Minimize Engagement_loss: loss between predicted watch times and true times

Solution: combine different models

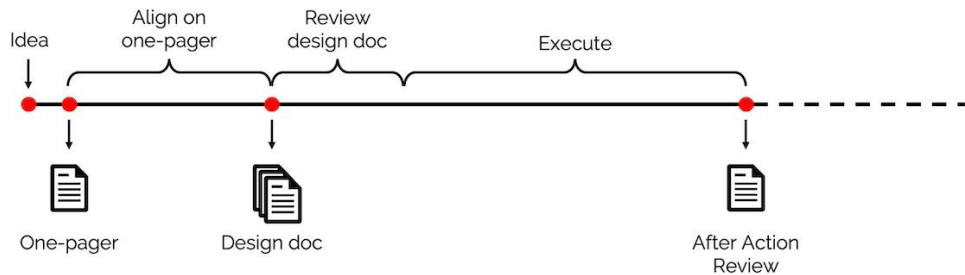
- Train two models
 - Model A: rating_loss
 - Model B: engagement_loss
 - Rank movies by $\alpha \cdot \text{pred_modelA} + \beta \cdot \text{pred_modelB}$

Decouple different objectives

- Easier for training
- Easier to tweak our systems
 - No need to retrain the whole system if weights for different objectives are changed
- Easier for maintenance
 - Different objectives might need different maintenance schedules

One-Pager for Machine Learning Projects

- Amazon Writing Style Tip
 - <https://medium.com/fact-of-the-day-1/amazon-writing-style-tip-a349b4bd3839>
- How to write design documents for data science/machine learning projects?
 - <https://eugeneyan.com/writing/writing-docs-why-what-how/>

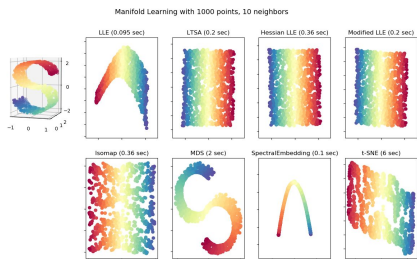


Three types of documents required during projects

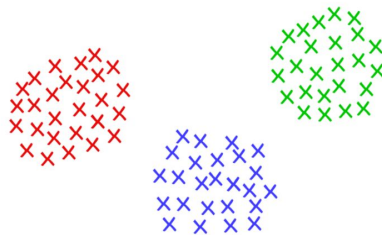
2. Autoencoders

Unsupervised Learning

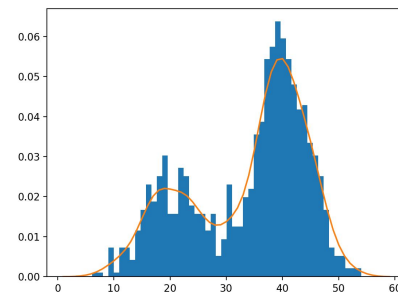
- Given the data x without labels
- Goal: Learn hidden structure (low dimension)



Representation Learning
Data lies on a low-dimensional manifold



Clustering
Group data points based their similarity

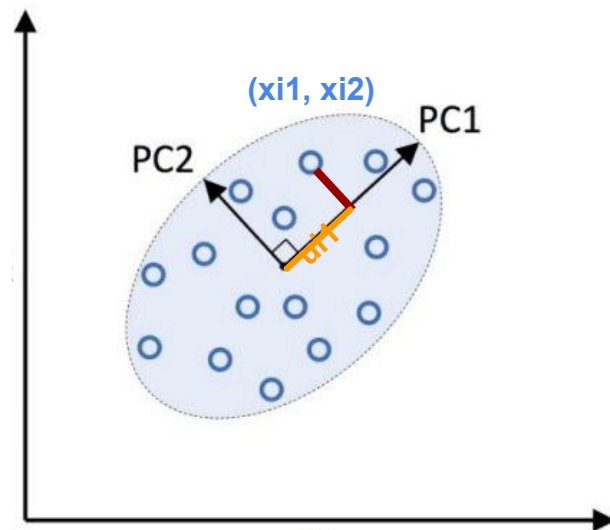


Density Estimation
Estimate data probability $p(x)$ from data x_1, x_2, \dots, x_n

Principal Component Analysis: Maximize Variance

- PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one
- Goal: Learn hidden structure (low dimension)

Original Space		Projection Matrix		New/Latent Space
$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{bmatrix}$	\times	$\begin{bmatrix} w_{11} \\ w_{21} \end{bmatrix}$	$=$	$\begin{bmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{n1} \end{bmatrix}$
		PC1		



MNIST Dataset

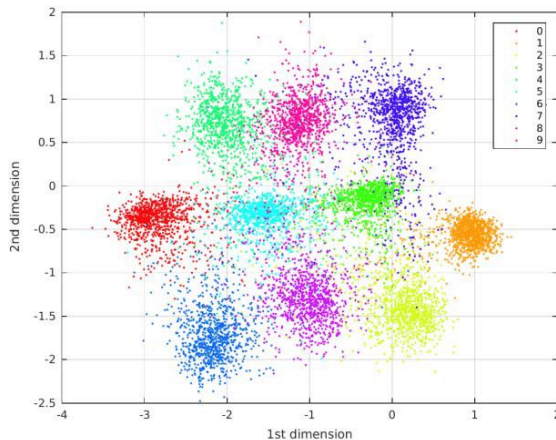


12

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	.6	.8	0	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0	0
0	0	0	0	0	0	.5	1	.4	0	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0	0
0	0	0	0	0	0	0	1	.7	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	.9	1	.1	0	0	0	0	0
0	0	0	0	0	0	0	.3	1	.1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCA for MNIST Visualization

- Each image has 28 by 28 pixels -> 28 by 28 matrix -> 784 dimensional vector
- Using PCA, find a project matrix $\mathbf{W} \in R^{784 \times 2}$
- After project, each image can be encoded into a 2-dimensional space



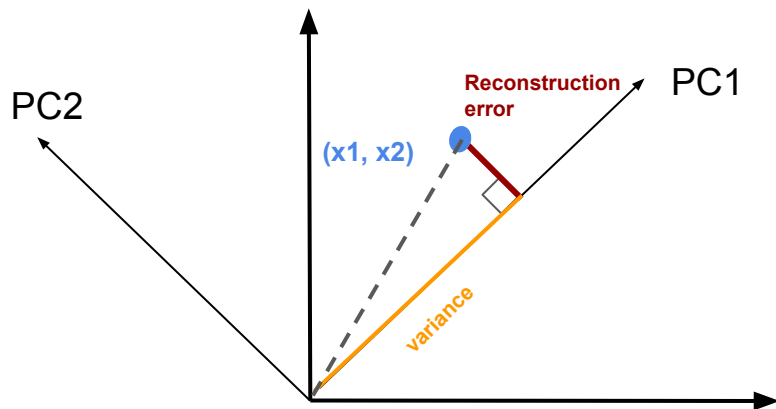
PCA: Minimize Reconstruction Error

- PCA aims to find a linear subspace that minimize the distance of the project in a least-square sense

minimize \mathbf{W} $\|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{W}^T\|_F^2$

subject to $\mathbf{W}^T\mathbf{W} = \mathbf{I}$

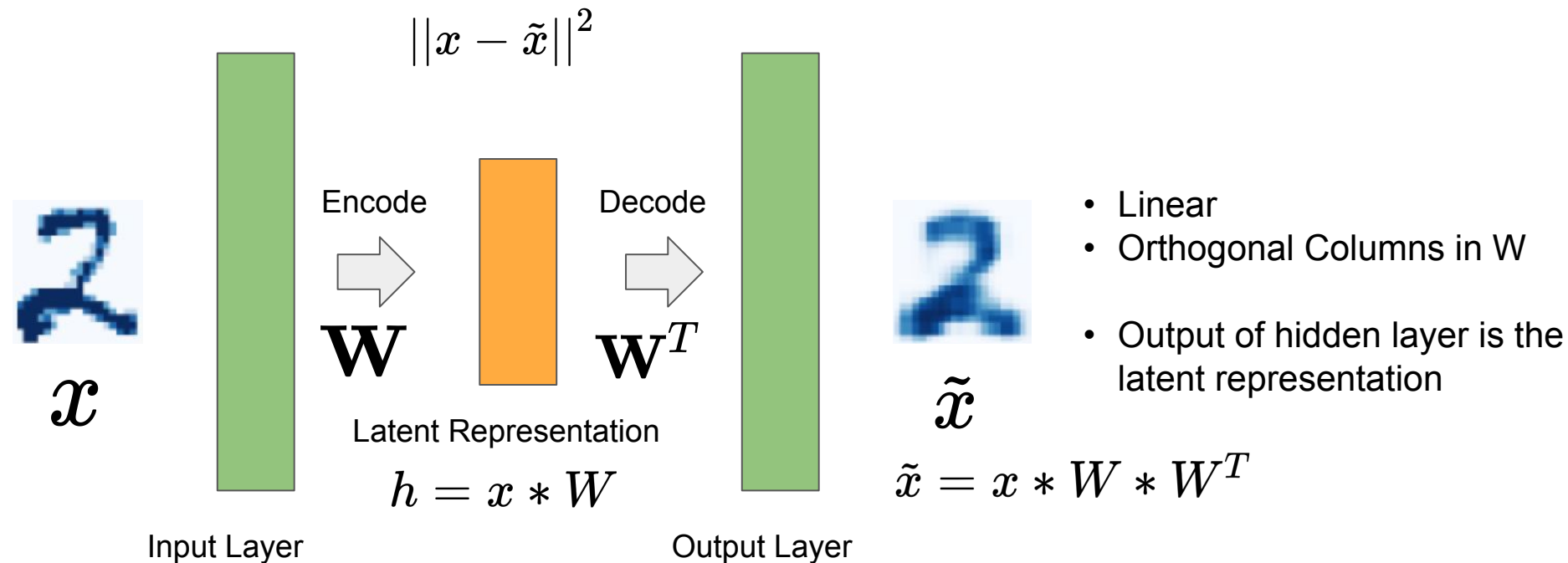
\mathbf{W} 's shape is (d, h) and $h < d$



Reconstruction Error + **Variance** = Constant

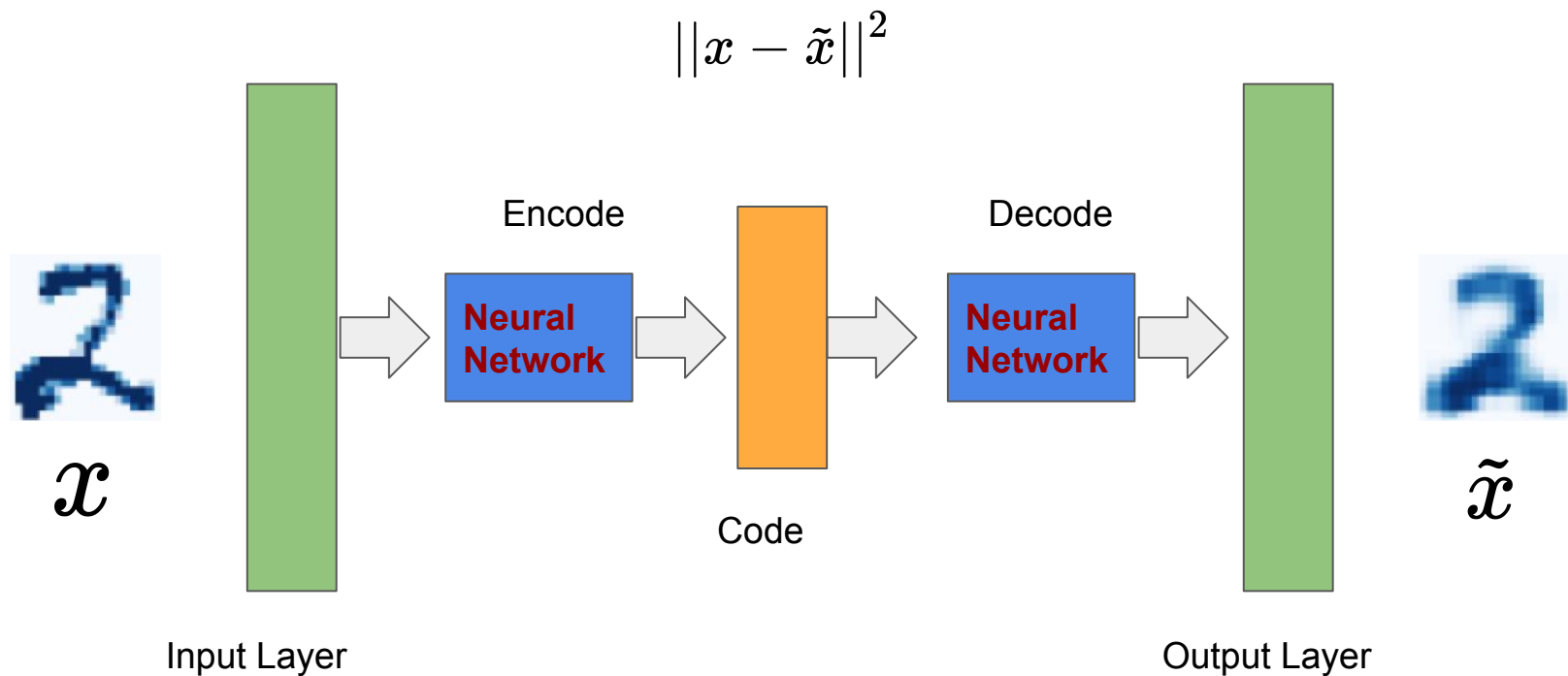
minimize *maximize*

PCA in neural network format

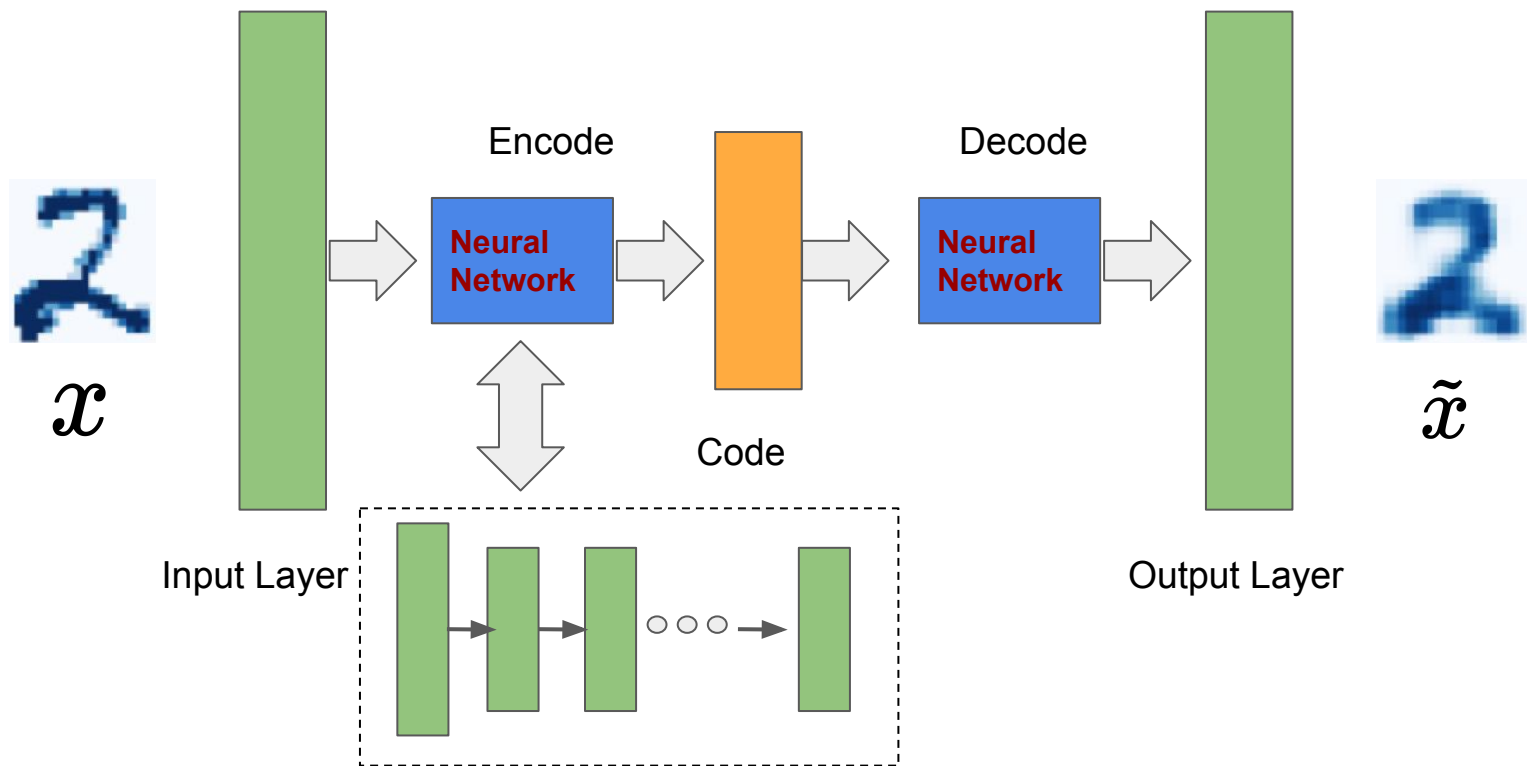


- Non-linear relationship between original representation and latent features
- Which machine learning models is used for **nonlinear approximation**?

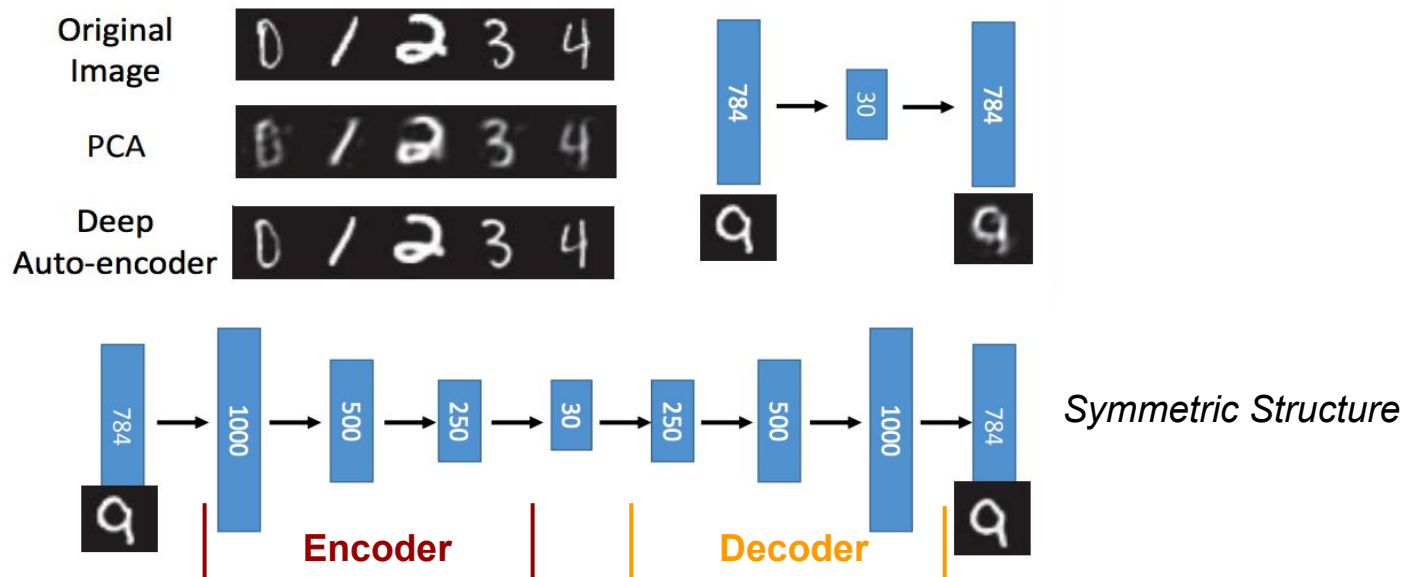
Autoencoder: NonLinear



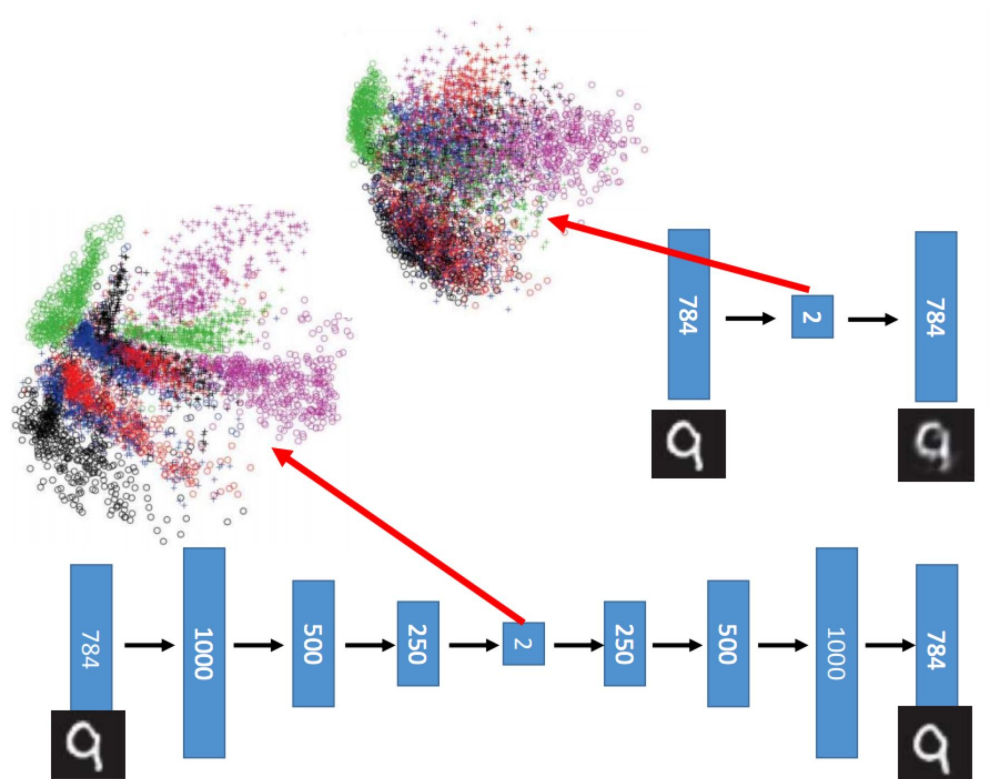
Deep Autoencoder



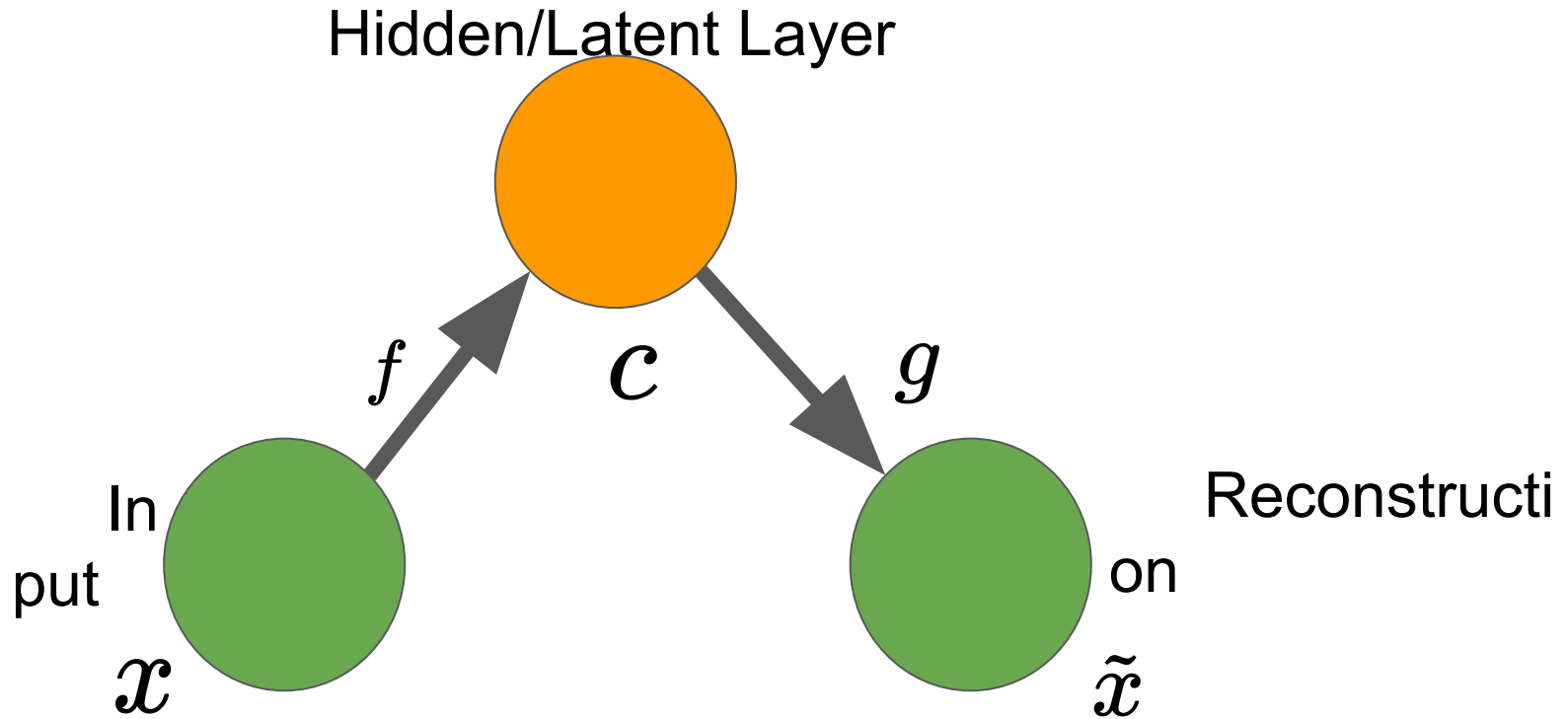
Deep Autoencoder vs PCA



Deep Autoencoder vs PCA



Structure of Autoencoder

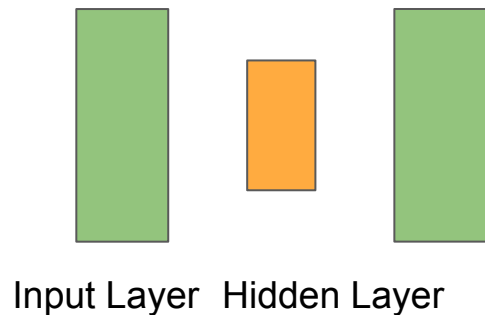


Undercomplete Autoencoder

- Simply copy input to output without learning anything useful
 - The autoencoder just mimic the identity function
 - Reconstruct the training data perfectly
 - Overfitting
- To avoid the above issues, we should use undercomplete autoencoders
 - The hidden layer size c is small compared to the original feature dimensionality

Sandwich Architecture in Autoencoder

- Forcing c (hidden layer size) is less than d (the input layer size)
 - Learn the important features
 - Information bottleneck:
 - A kind of trade-off between compression and retaining information

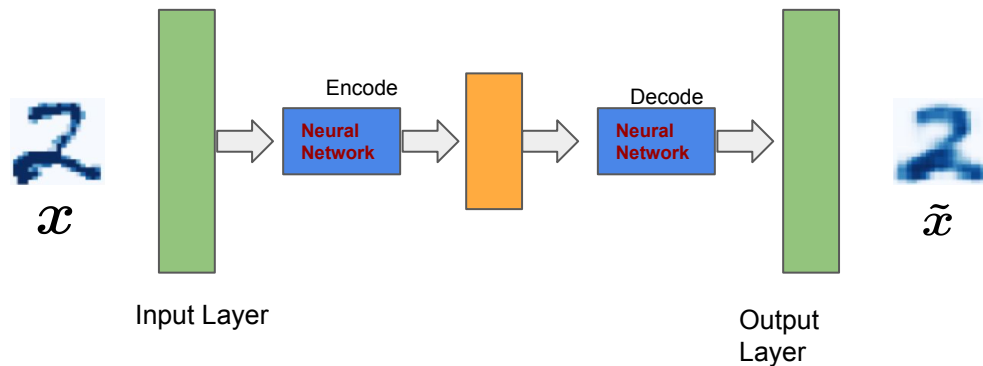


Original **6** Bricks

Can we use only **4** bricks to rebuild the previous shape?

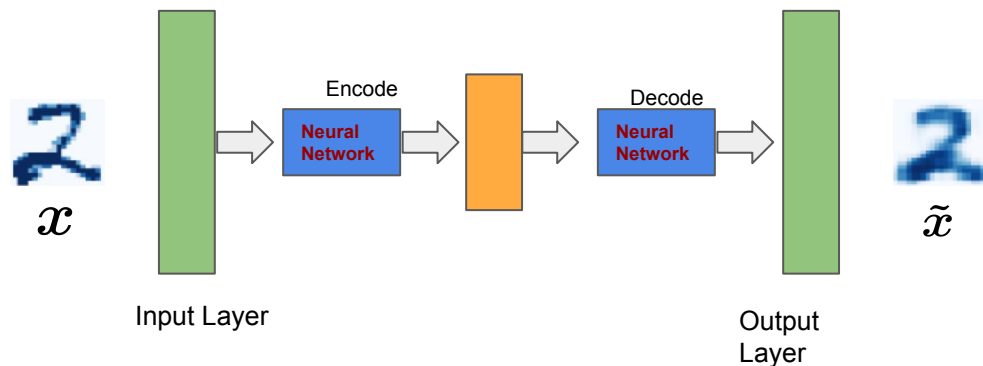
Optimization Targets

- For Autoencoder, the training objective is to minimize $||x - \tilde{x}||^2$
- Hidden representation is what we really want to learn



Unsupervised or Self-supervised

- Autoencoder is one kind of self-supervised learning $||x - \tilde{x}||^2$
- Input is x , target is x
- Pretend there is part of the input you do not know and predict that



Build Autoencoders in Keras

<https://blog.keras.io/building-autoencoders-in-keras.html>

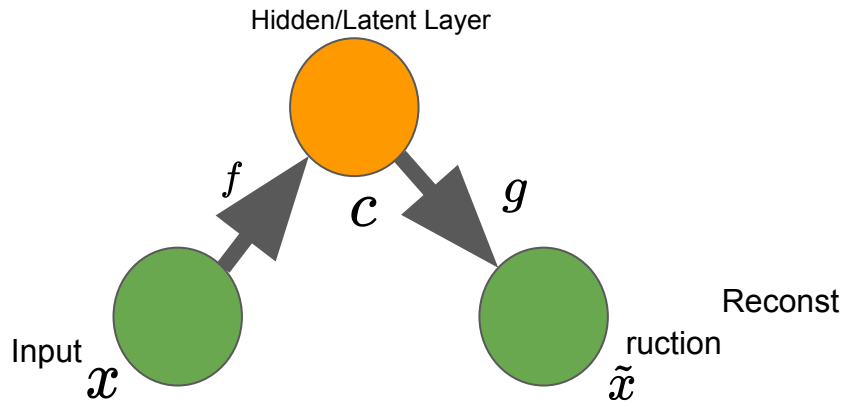
Regularized Autoencoder

Add constraints in case the identity transformation is learned, i.e., overfitting

Sparse Autoencoders

- Constrain on c that penalizes it from dense
- Regularization on output of encoder, not parameters

$$L(x, g(f(x))) + \Omega(c)$$



```
• kernel_regularizer : instance of keras.regularizers.Regularizer  
• bias_regularizer : instance of keras.regularizers.Regularizer  
• activity_regularizer : instance of keras.regularizers.Regularizer
```

Example

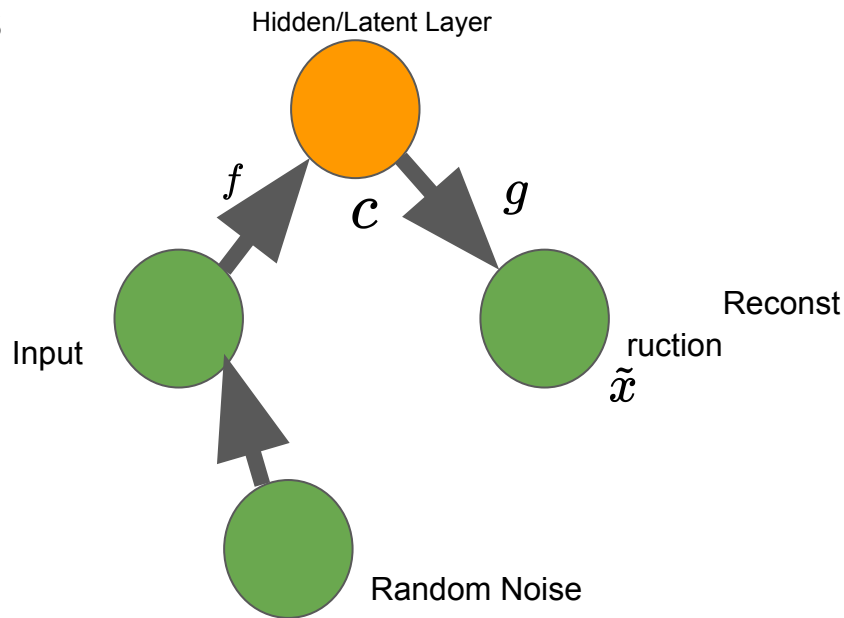
```
from keras import regularizers  
model.add(Dense(64, input_dim=64,  
                kernel_regularizer=regularizers.l2(0.01),  
                activity_regularizer=regularizers.l1(0.01)))
```


Denoising Autoencoders

- Add noise into original data points
- Still reconstruct the original data points

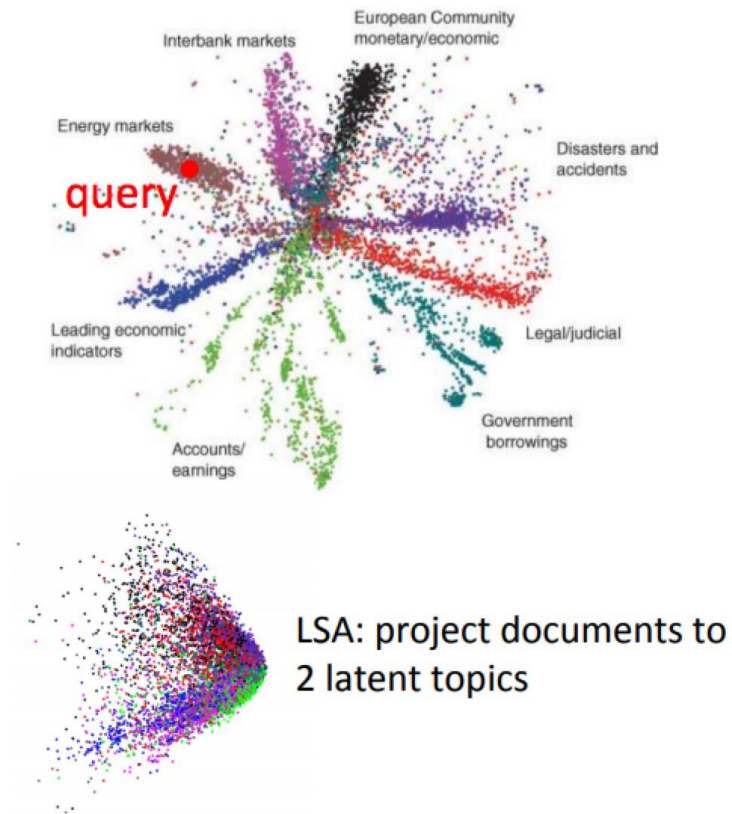
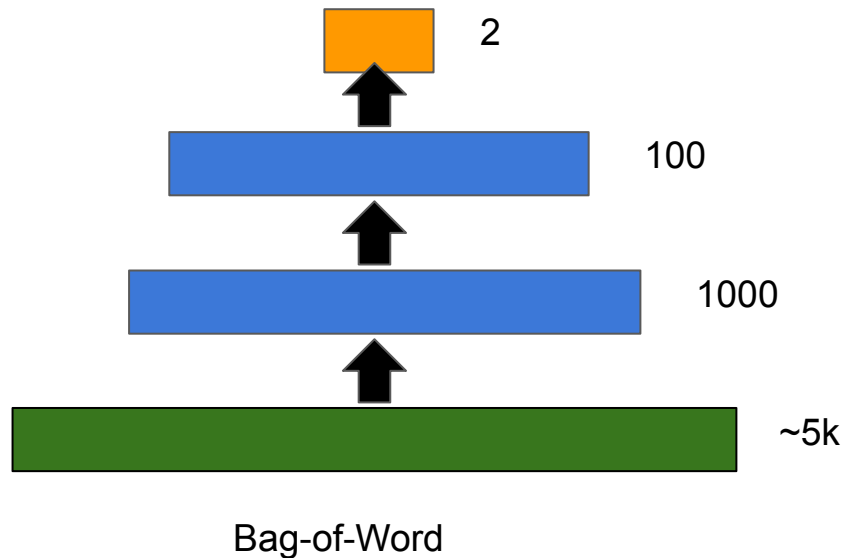
$$L(x, g(f(\bar{x})))$$

Corrupted copy of x



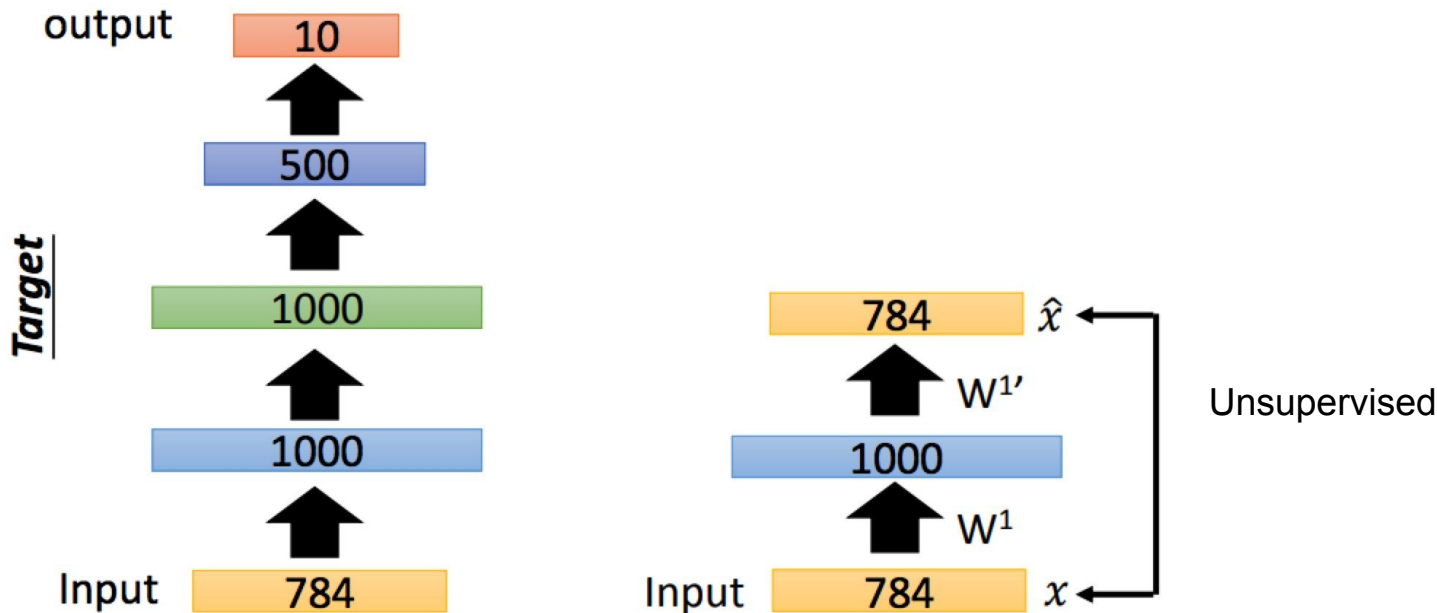
3. Applications of Autoencoders

Better Representation



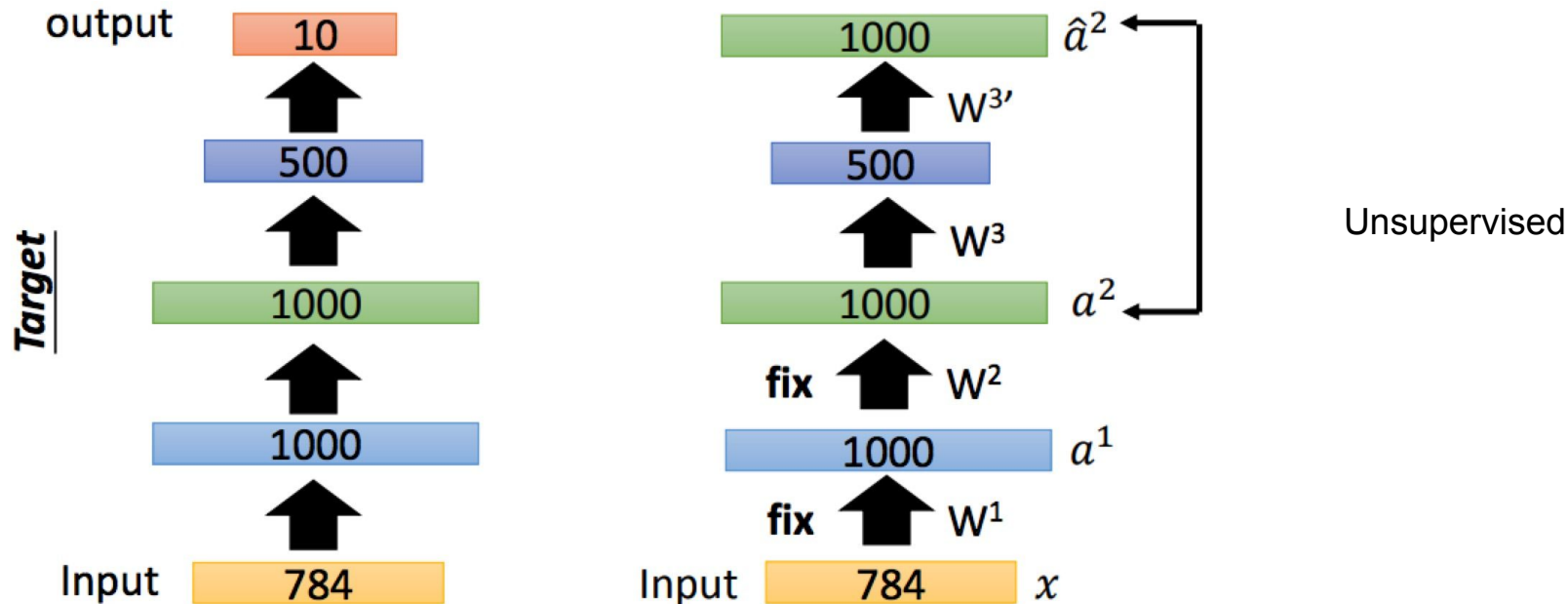
Pre-training Deep Neural Network

- Greedy Layer-wise Pre-training for W_1



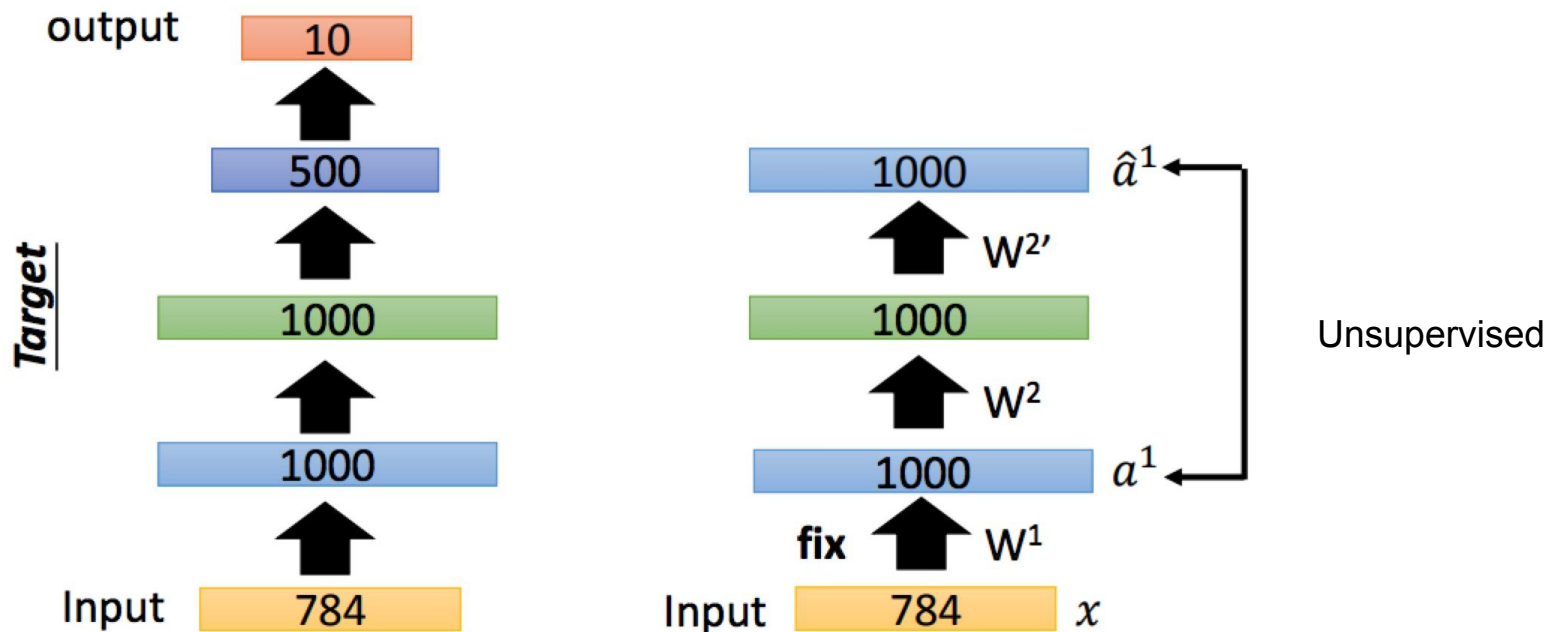
Pre-training Deep Neural Network

- Greedy Layer-wise Pre-training for W_3



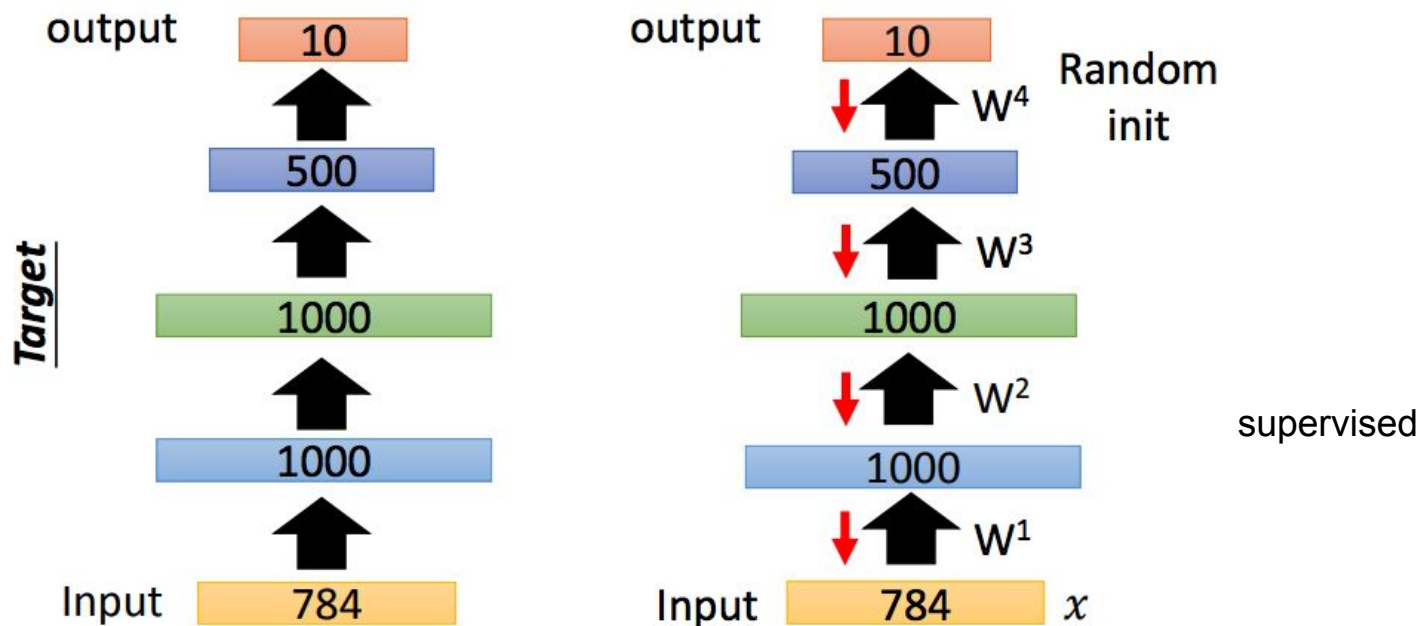
Pre-training Deep Neural Network

- Greedy Layer-wise Pre-training for W^2



Pre-training Deep Neural Network

- Fine-tune by backpropagation



4. Recommendation Systems



Arjun Narayan 

@narayanarjun

Follow



The two best performing public stocks of the decade - Netflix (+3700%) and Domino's Pizza (+3000%) - perfectly epitomize the 2010s. You either build the world's most advanced machine learning content recommender system, or make a better pizza sauce, there's no middle ground.

1:20 PM - 27 Dec 2019

3,926 Retweets 20,086 Likes



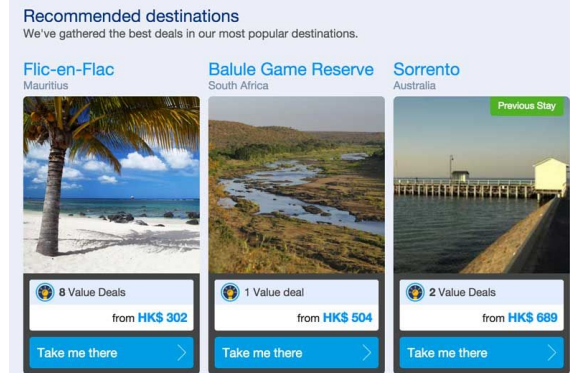
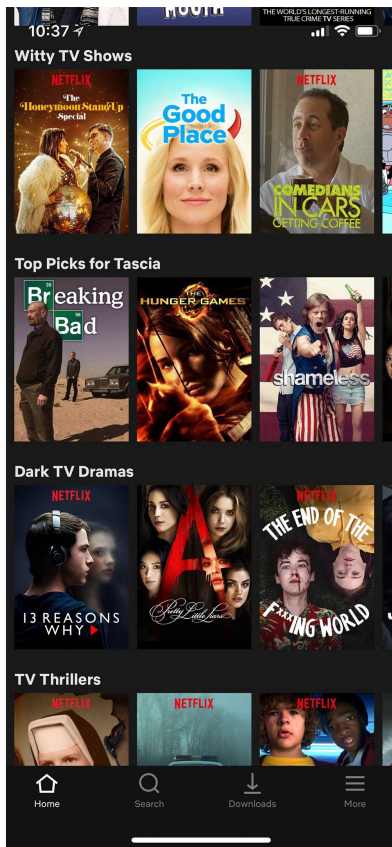
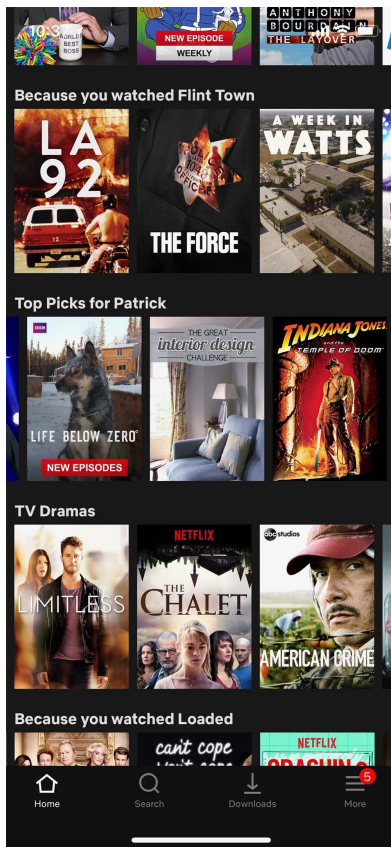
183



3.9K



20K



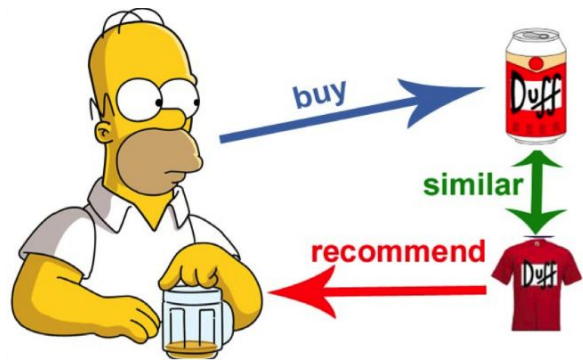
Core Problem in Rec. Sys.

- Filter Information for users
- Personalization is the key:
 - Given a certain user, compute the score that quantifies how strongly a user likes item i .



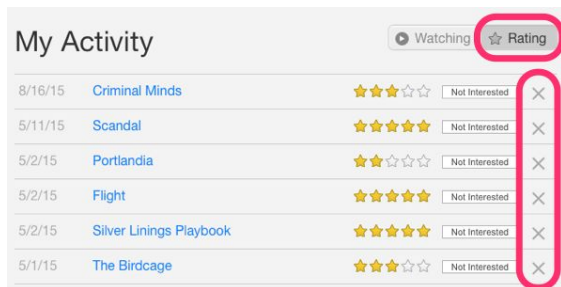
Content-based Method

- Define the similarity from items' content
 - Name: cosine similarity
 - Category
 - Rating
 - Description
 - Etc
- Combine them into a final score
- Ranked items based on their similar scores compared to users' purchased item.



User Behaviour

- Content-based methods: only look at the items' information
- The Insights behind the huge interaction behind users and items

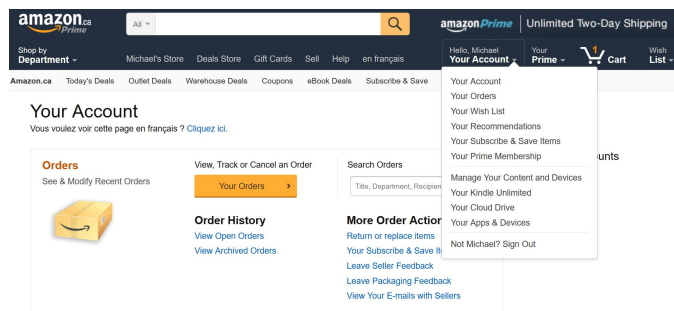


My Activity

Watching **Rating**

8/16/15	Criminal Minds	☆☆☆☆☆	Not Interested	×
5/11/15	Scandal	☆☆☆☆☆	Not Interested	×
5/2/15	Portlandia	☆☆☆☆☆	Not Interested	×
5/2/15	Flight	☆☆☆☆☆	Not Interested	×
5/2/15	Silver Linings Playbook	☆☆☆☆☆	Not Interested	×
5/1/15	The Birdcage	☆☆☆☆☆	Not Interested	×

Ratings in Netflix



Order History

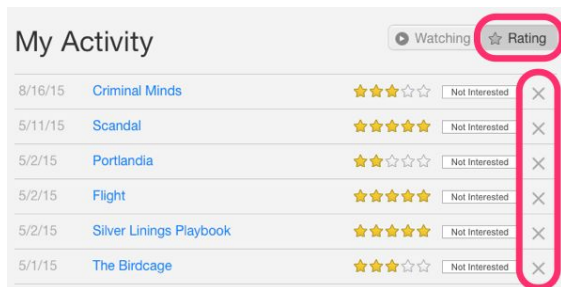
User-Item Matrix

- Content-based methods: only look at the items' information
- The Insights behind the huge interaction behind users and items

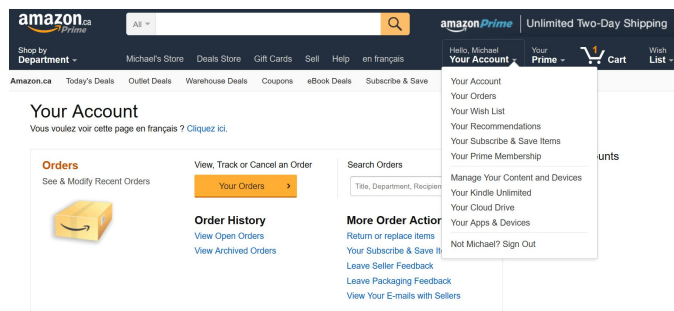
		Item Vector					
		Item 1	Item 2	Item 3	Item k-1	Imte k
User Vector	User 1	1	0	0		3	1
	User 2	0	3	1		0	2
	...						
	User n-1	0	2	0		1	1
	User n	0	0	0		0	0

User Behaviour

- Content-based methods: only look at the items' information
- The Insights behind the huge interaction behind users and items



Ratings in Netflix



Order History

User-based CF

- Find the similarity score between users
- Recommend products which these similar users have liked or bought previously

$$P_{u,i} = \frac{\sum_{v \in U} (r_{v,i} * s_{u,v})}{\sum_{v \in U} s_{i,v}}$$

The prediction of an item i for user u

The rating of item i given by user v

User Space

The similarity between users u and v

$$s_{u,v} = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} * \vec{v}}{||\vec{u}|| ||\vec{v}||}$$

Cosine similarity used a lot in information retrieval

Item-based CF

- Find the similarity score between items
- Recommend similar items which were liked or purchased by the users in the past

$$P_{u,i} = \frac{\sum_{m \in I} (r_{u,m} * s_{i,m})}{\sum_{m \in I} s_{i,m}}$$

The prediction of an item i for user u

The rating of item m given by user u

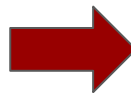
The similarity between items i and m

Item Space

$$s_{i,m} = \cos(\vec{i}, \vec{m}) = \frac{\vec{i} * \vec{m}}{||\vec{i}|| ||\vec{m}||}$$

Data Sparsity

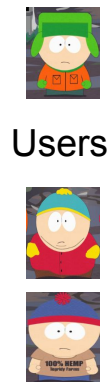
movieId	1	2	3	4	5	6	7	9	10	11	...	106487	106489	106782	106920	109374
userId																
316	-0.829457	NaN	NaN	NaN	NaN	NaN	-1.329457	NaN	-0.829457	NaN	...	NaN	NaN	NaN	NaN	NaN
320	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
359	1.314526	NaN	NaN	NaN	NaN	1.314526	NaN	NaN	0.314526	0.314526	...	NaN	NaN	NaN	NaN	NaN
370	0.705596	0.205596	NaN	NaN	NaN	1.205596	NaN	NaN	NaN	NaN	...	-1.294404	-0.794404	0.705596	0.205596	NaN
910	1.101920	0.101920	-0.39808	NaN	-0.39808	-0.398080	NaN	NaN	NaN	0.101920	...	NaN	NaN	-0.398080	NaN	NaN



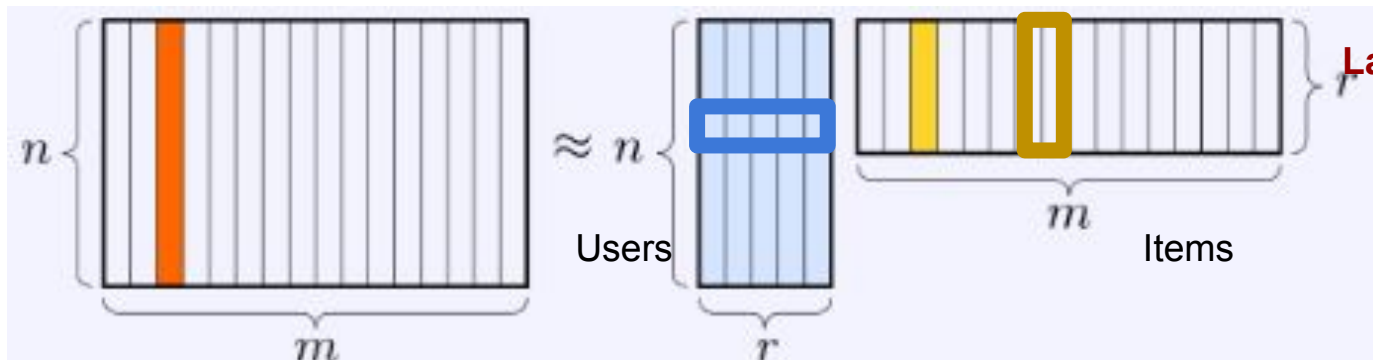
Similarities
between users
and items are
zero

- The core problem behind recommendation sys. is to fill these zero entries, i.e., infer the users preference over the item.
 - Address as data missing problems:
 - Use the mean value of the row
 - Use the mean value of the column
 - Matrix Factorization
 - Singular Value Decomposition
 - Non-Negative Matrix Factorization
 - Auto-encoder

NMF for Rec



Users

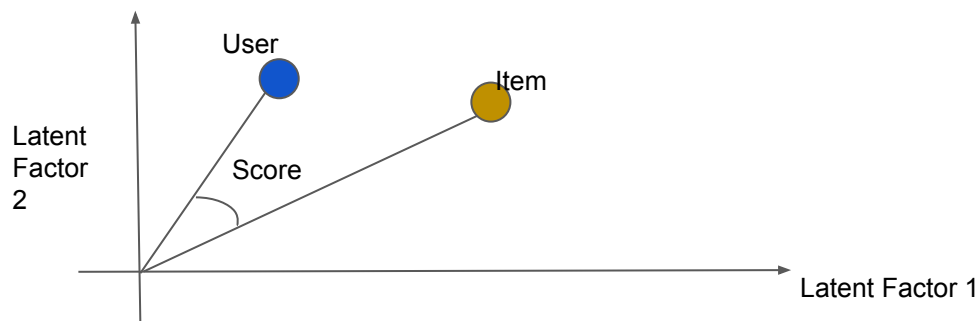


Latent Space

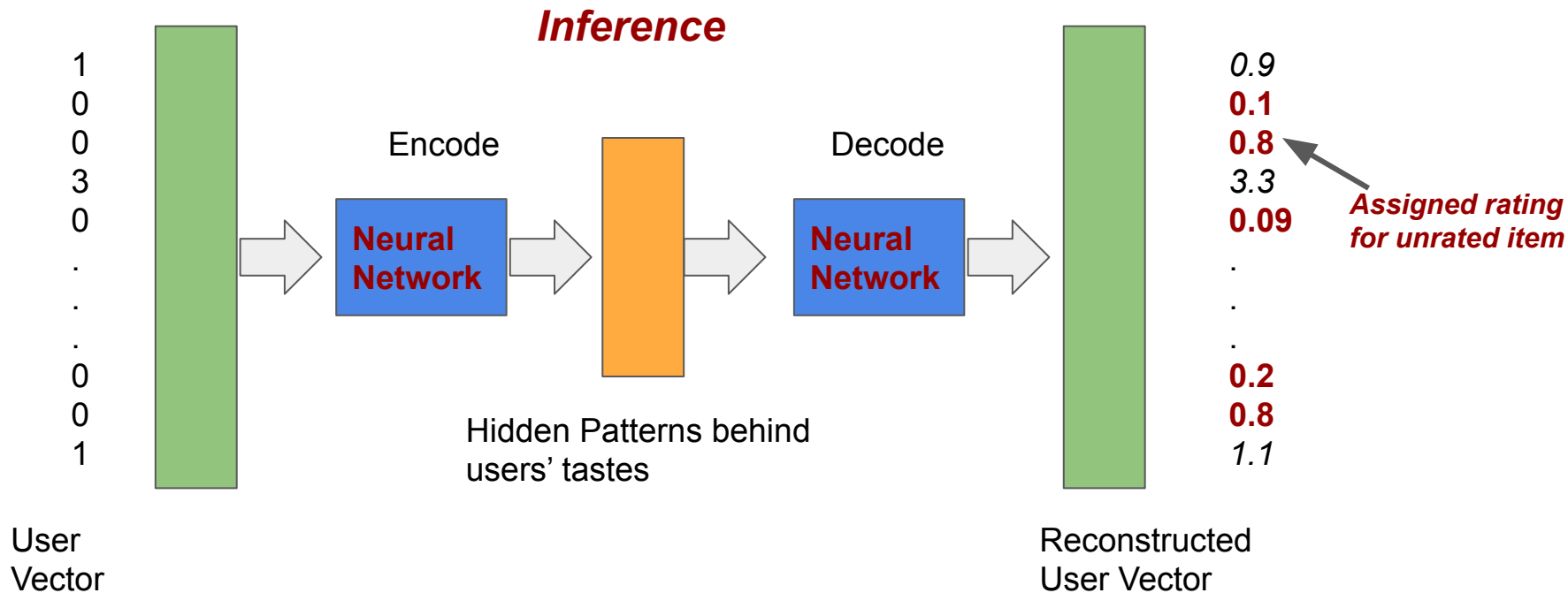


Items

Latent Space



Autoencoder for Rec.



Pros & Cons of CF

- Pros
 - Capture latent users and item factors
 - Can handle sparsity
 - Scalable computation (ALS)
- Cons:
 - Biases (Temporal and Popularity)
 - Cold Start Problem
 - No Context-awareness

How to evaluate Rec. Sys.

- Offline Evaluation
 - Train/test Splitting
 - RMSE
 - Recall
- Online Evaluation:
 - A/B Testing
 - Click-Through Rate (CTR)
 - Conversion Rate (CR)