Representation Learning: Autoencoder

Supervised Learning

- Give the data (x->y), x is the data, y is the label
- Goal: Learn the mapping: from x to y.

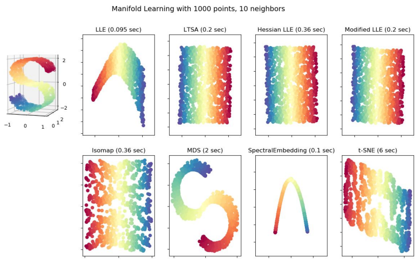
Stark Classification

Game of Thrones

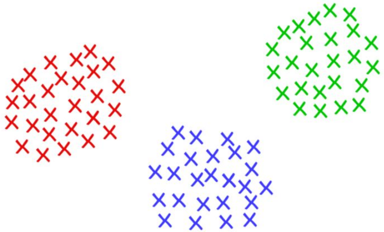
Iron Man

Unsupervised Learning

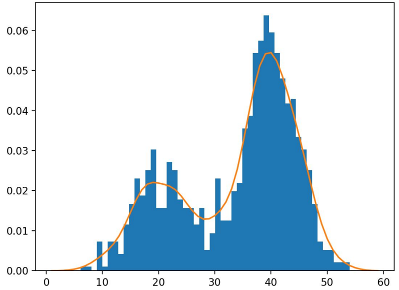
- Given the data x without labels
- Goal: Learn hidden structure(low dimension) from



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 x1, x2, ...,, xn

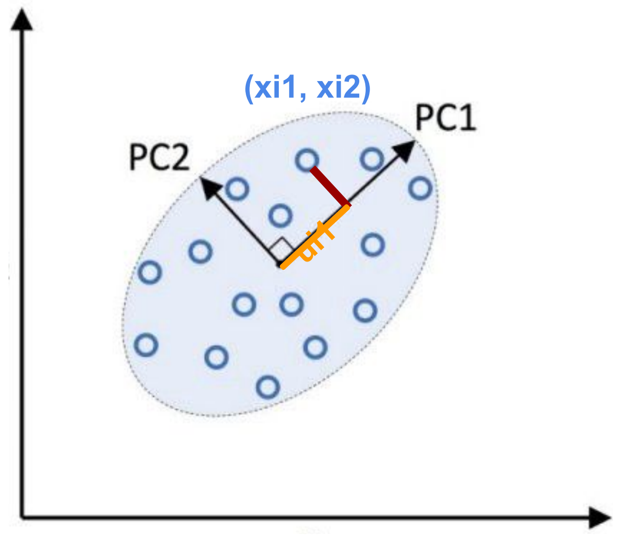
Autoencoder

Principal Component Analysis: Maximize Variance

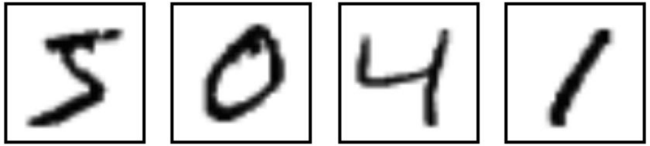
PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one

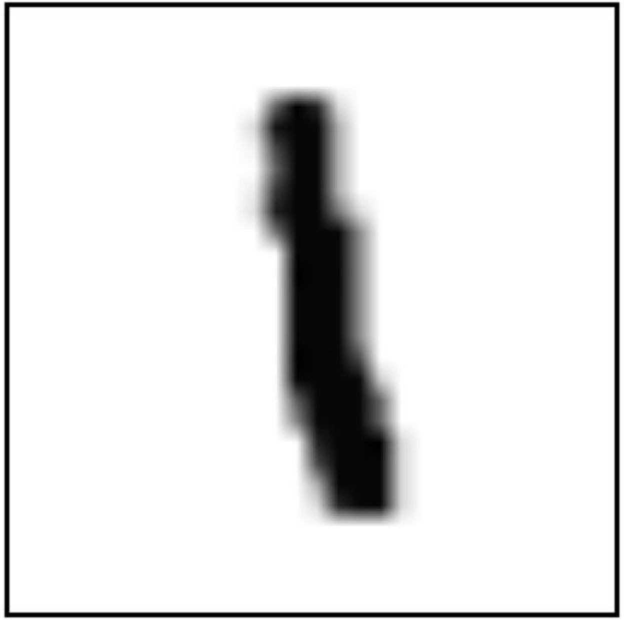
Original Space | Projection Matrix | New/Latent Space

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ \vdots & \vdots \\ u_{n1} & u_{n2} \end{bmatrix}$$



MNIST Dataset

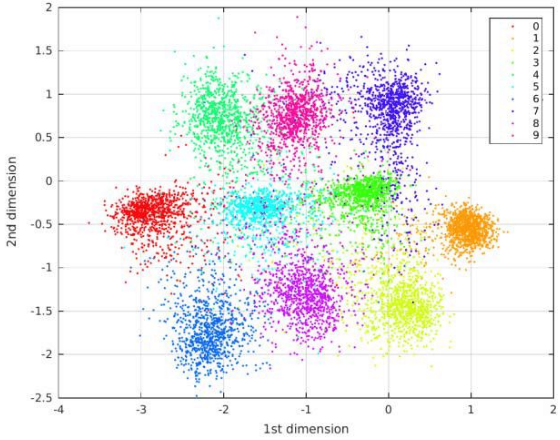




$$\cong \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

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 projection, each image can be encoded into a 2-Dimensional space



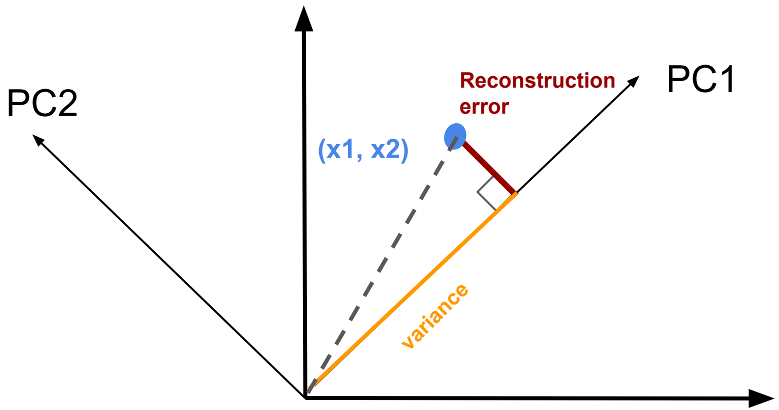
Principal Component Analysis: Minimize Reconstruction Error

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

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

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

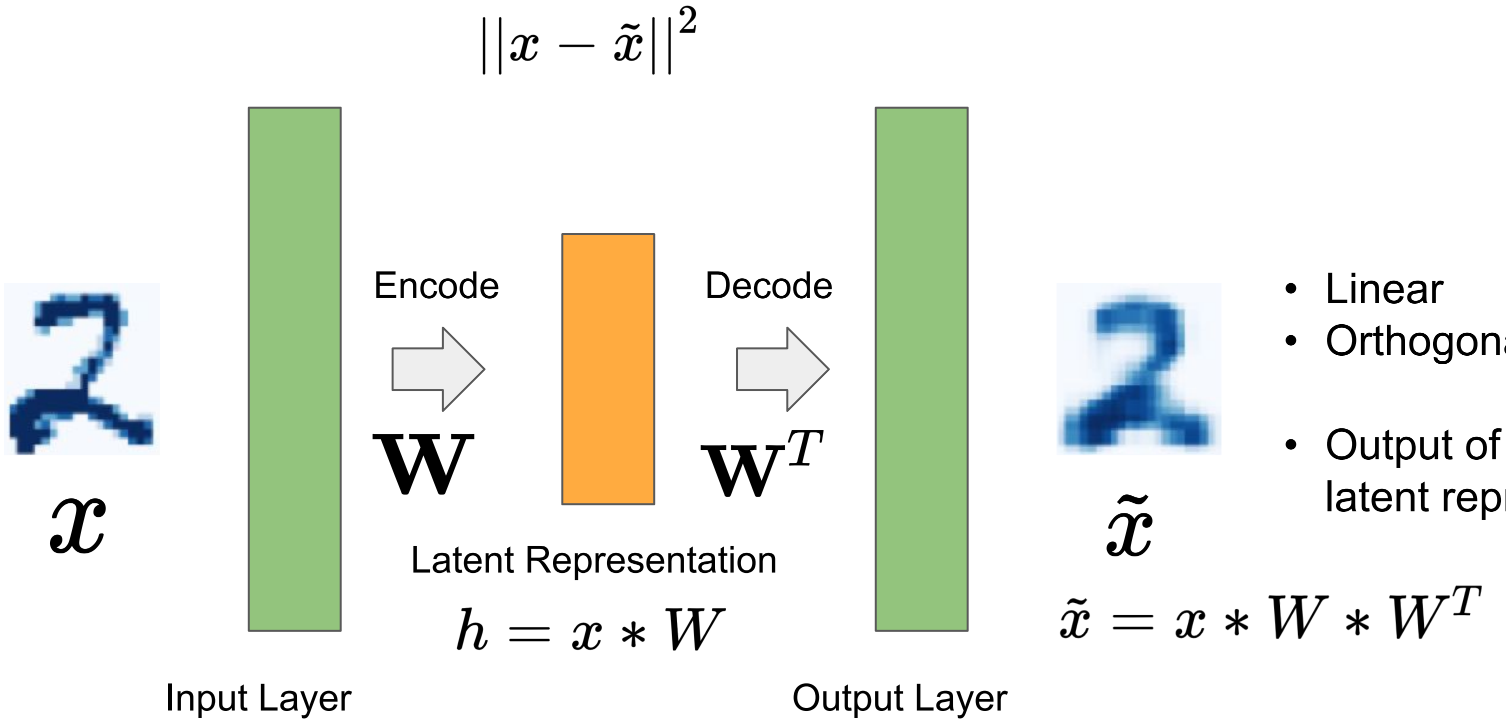
W's shape is (d, h) and h < d



Reconstruction Error + Variance = Constant

minimize *maximiz*

Principal Component Analysis



- Linear
- Orthogonal Columns in W

- Output of hidden layer is the latent representation

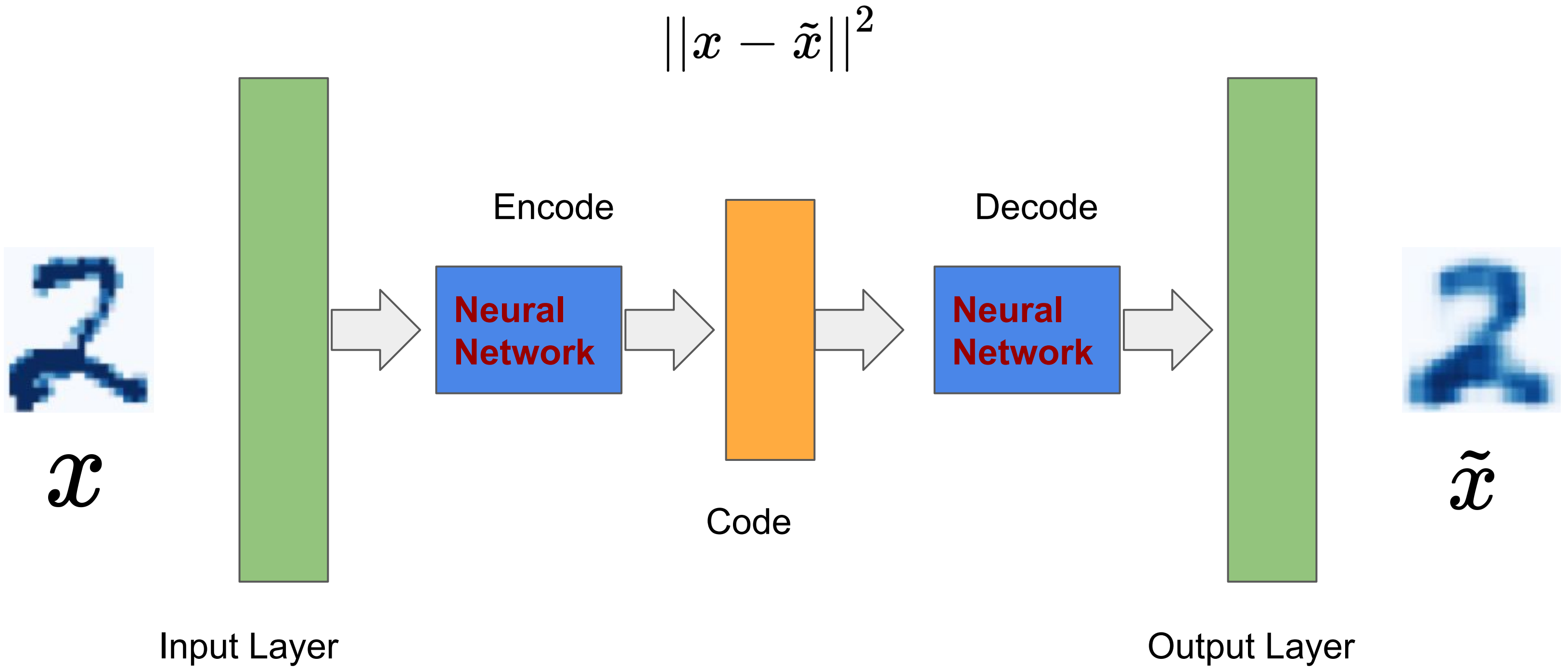
More Details on PCA: http://www.cs.cornell.edu/courses/cs4786/2016sp/lectures/lec03.pdf

- Non-linear relationship between original representation and latent features

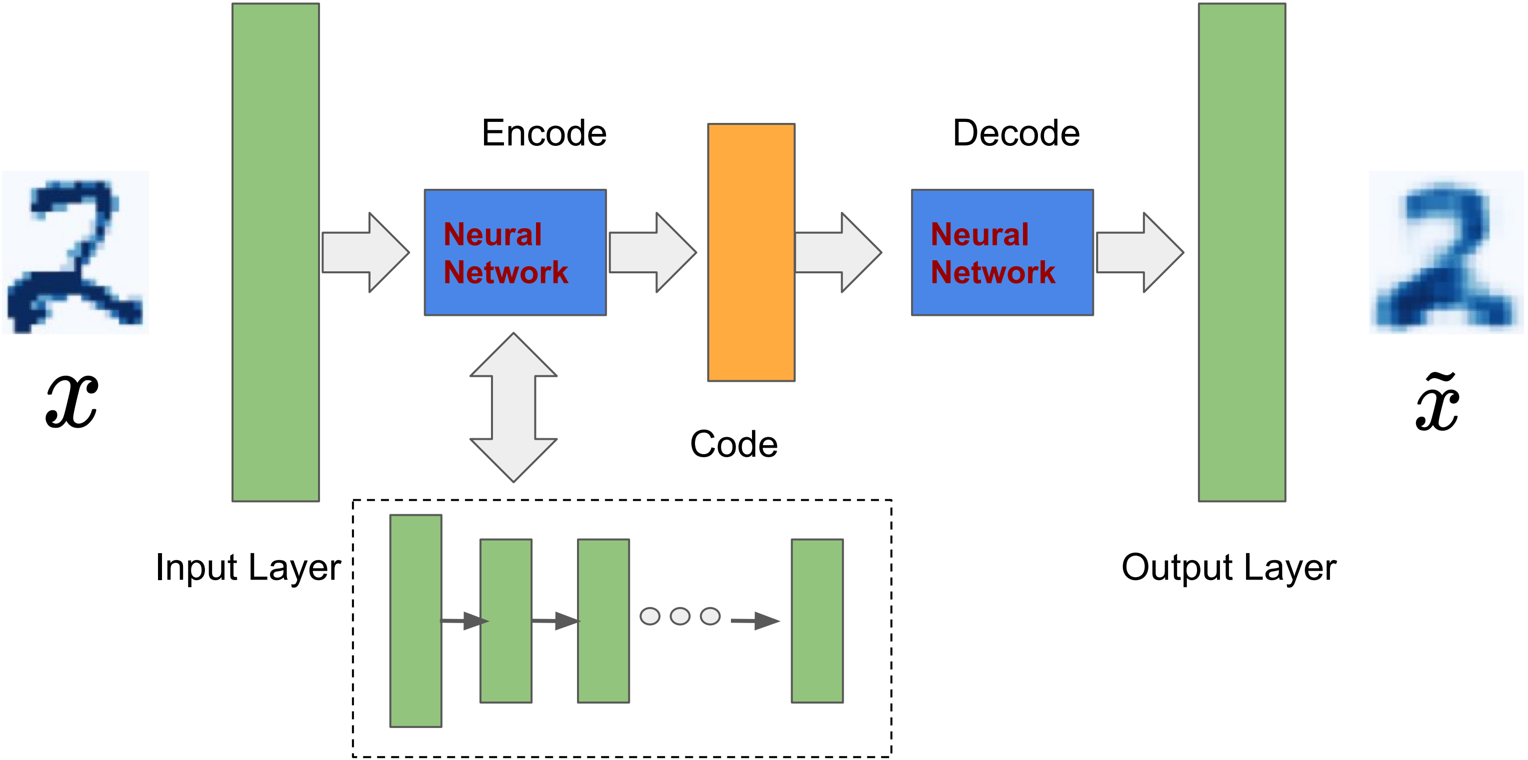
- Which machine learning model to use for **nonlinear approximation**?

Purpose of HW1

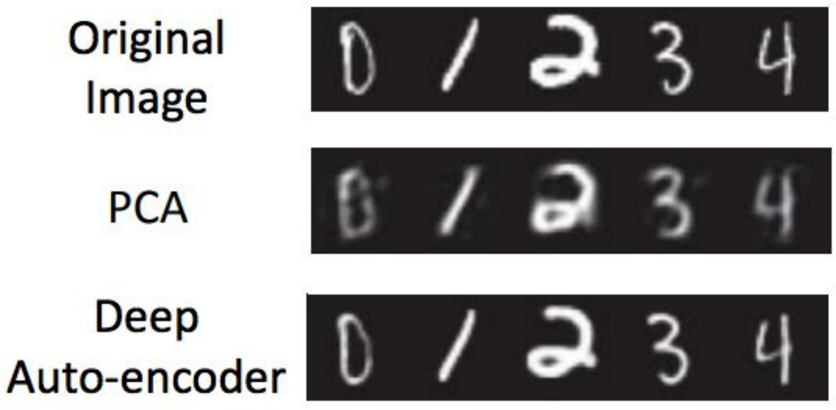
Autoencoder: NonLinear

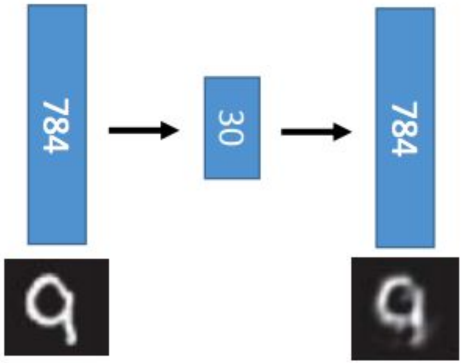


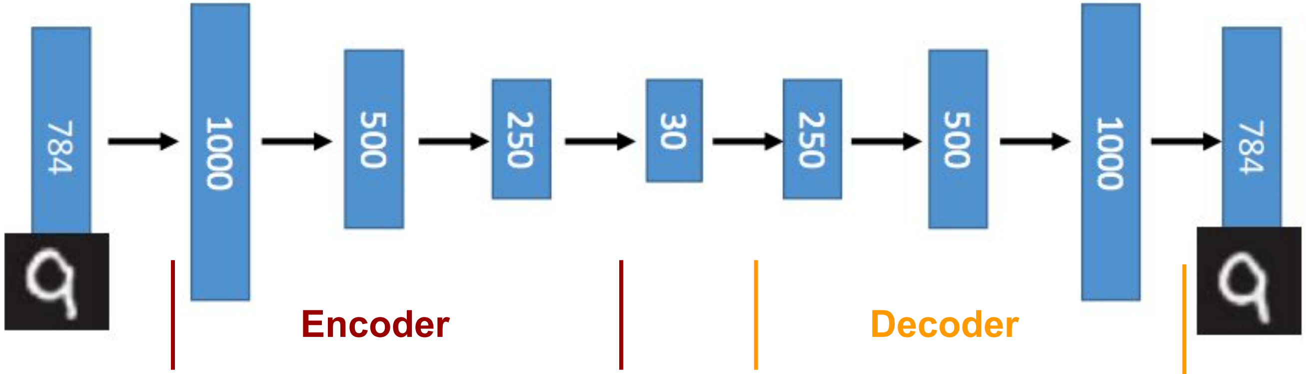
Deep Autoencoder



Deep Autoencoder vs PCA

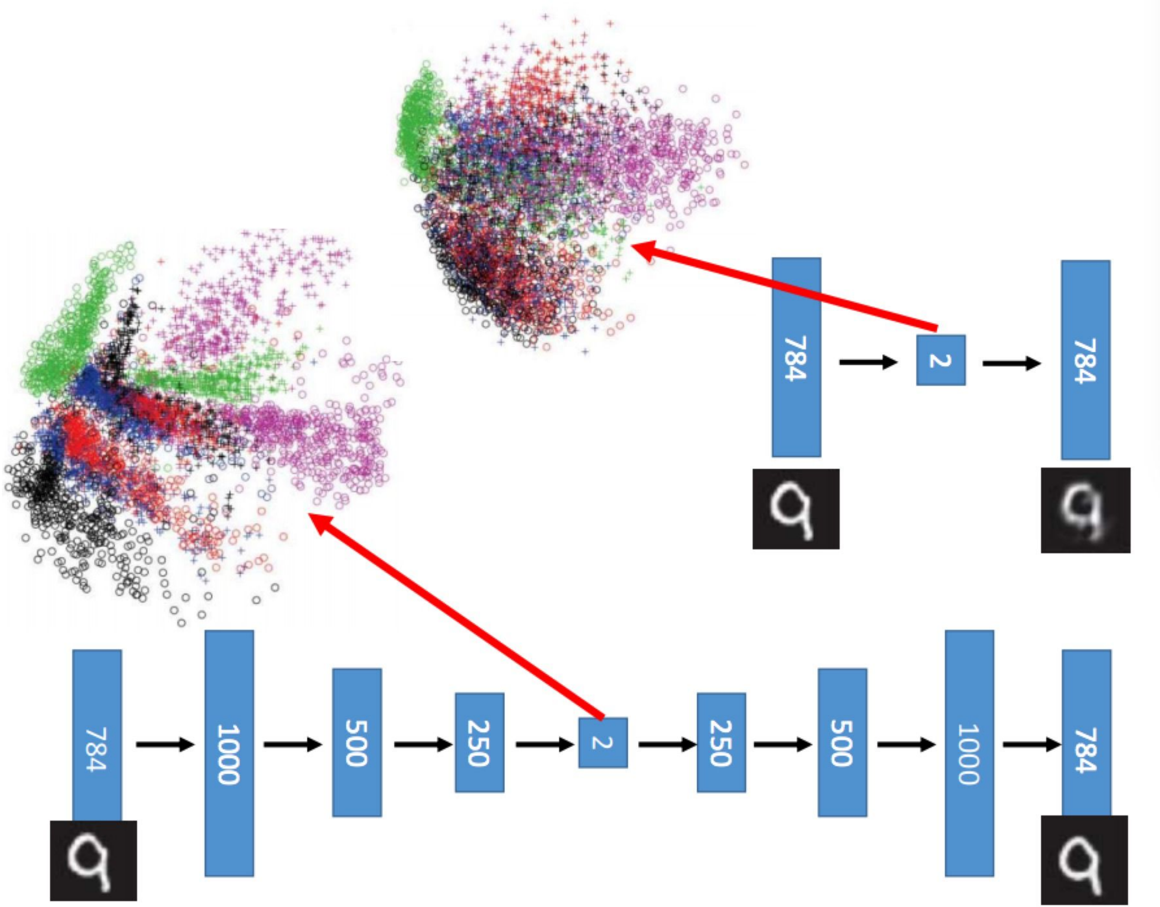




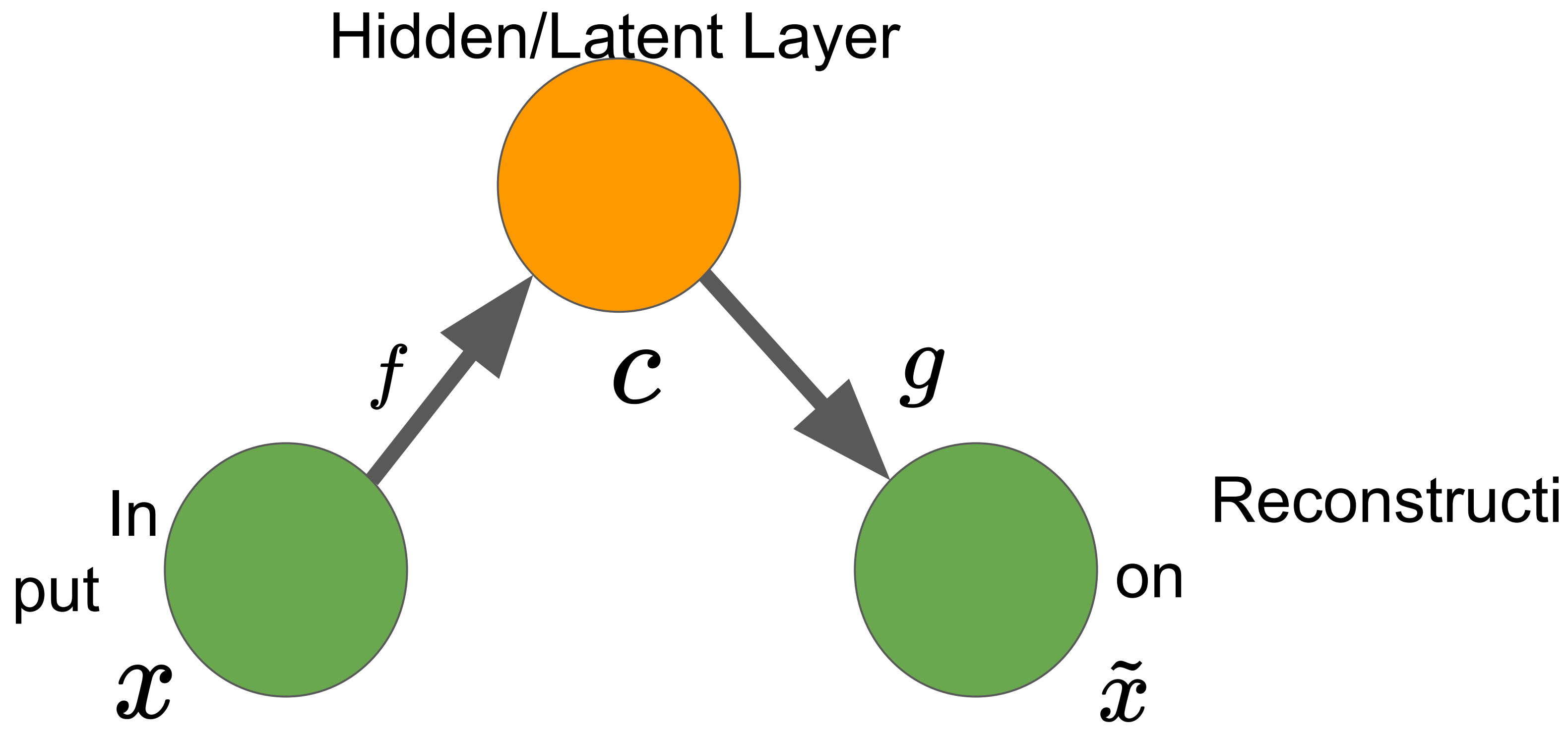


Symmetric Structure

Deep Autoencoder vs PCA



Structure of Autoencoder

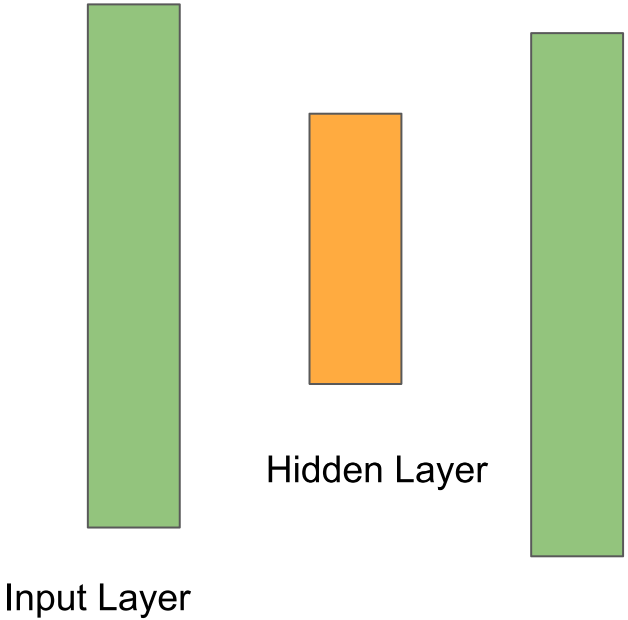


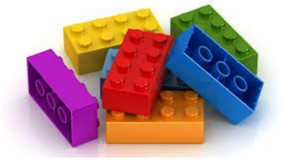
Undercomplete Autoencoder

- 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 issue, we should use undercomplete autoencoder
  - 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



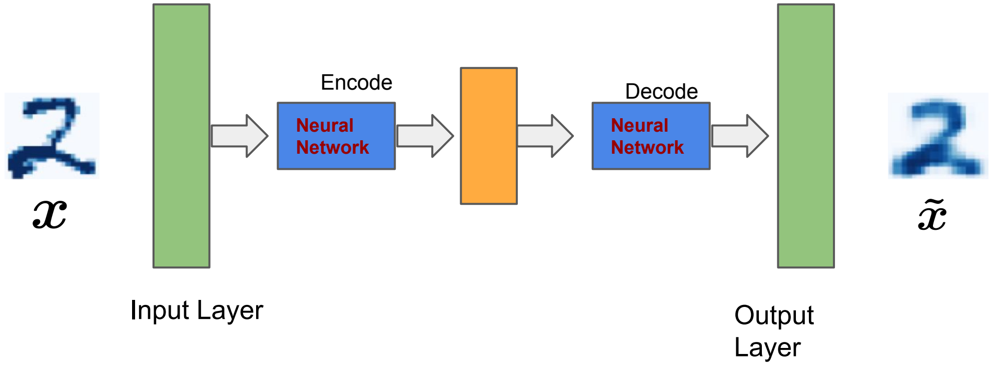


Original **6** Bricks

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

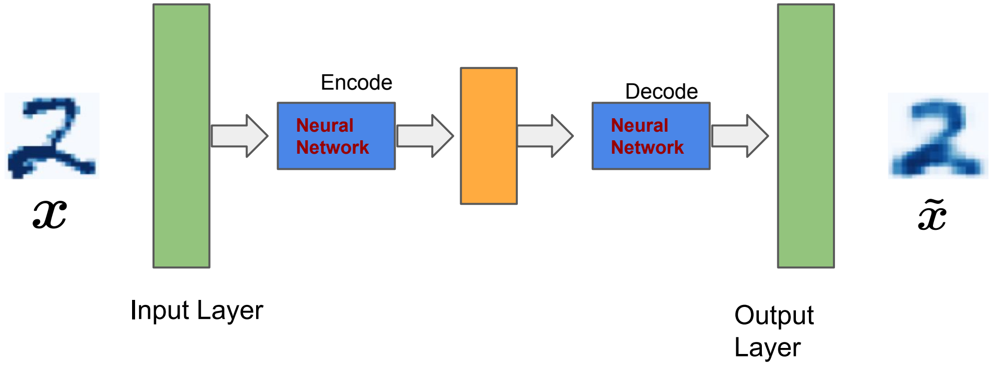
Optimization Targets

- For Autoencoder, the training objective is to minimize $||x - \tilde{x}||^2$
- But we do not care the output layer $\tilde{x}$
- Hidden representation is what we really want to learn



Unsupervised or Self-supervised?

- Autoencoder is one kind of self-supervised learning
- Input is x, target is x
- **Pretend there is part of the input you do not know and predict that**
- Word2vec



Build Autoencoders in Keras

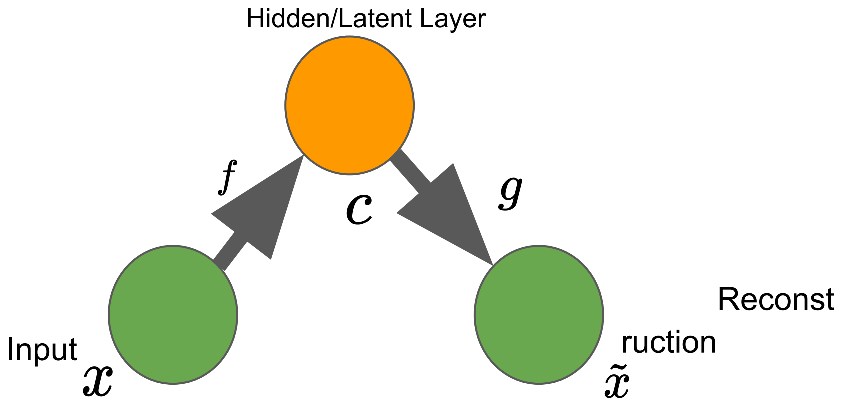
https://blog.keras.io/building-autoencoders-in-keras.html

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

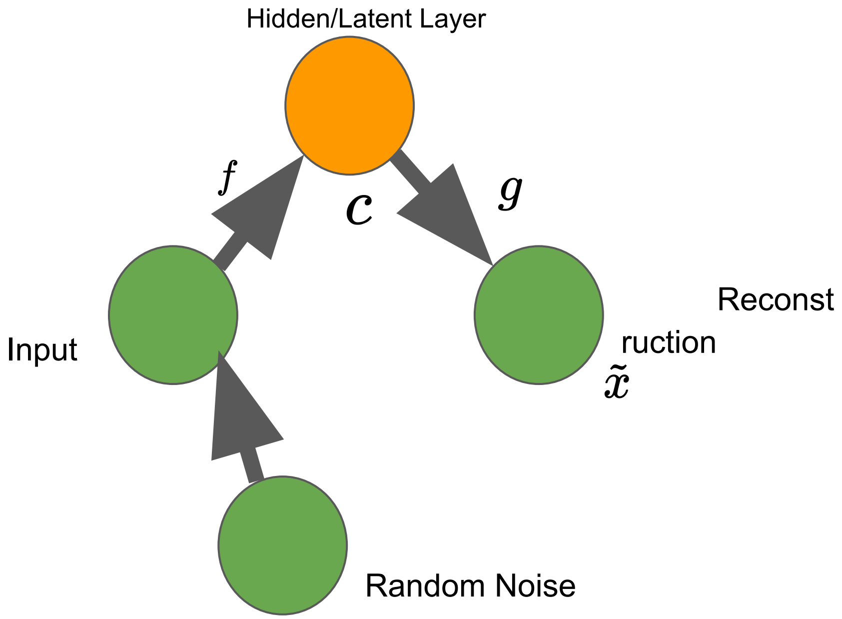
```
from keras import regularizers
model.add(Dense(64, input_dim=64,
                kernel_regularizer=regularizers.l2(0.01),
                activity_regularizer=regularizers.l1(0.01)))
```

Denoising Autoencoders

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

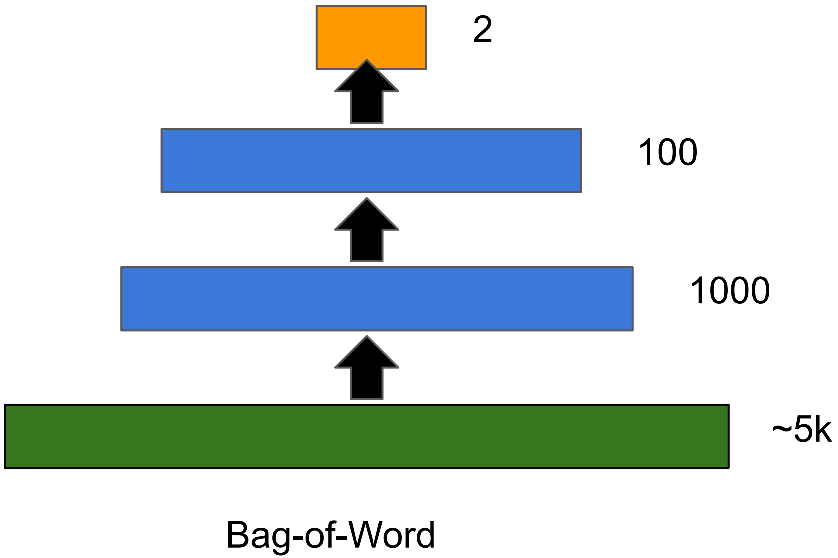
$$L(x, g(f(\bar{x})))$$

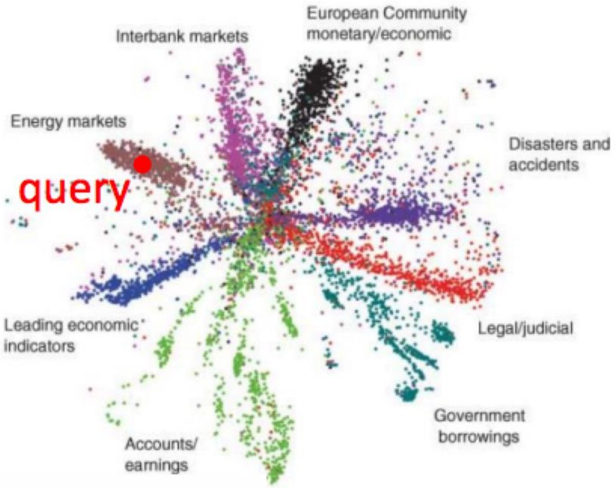
Corrupted copy of x



Applications of Autoencoders

Better Representation

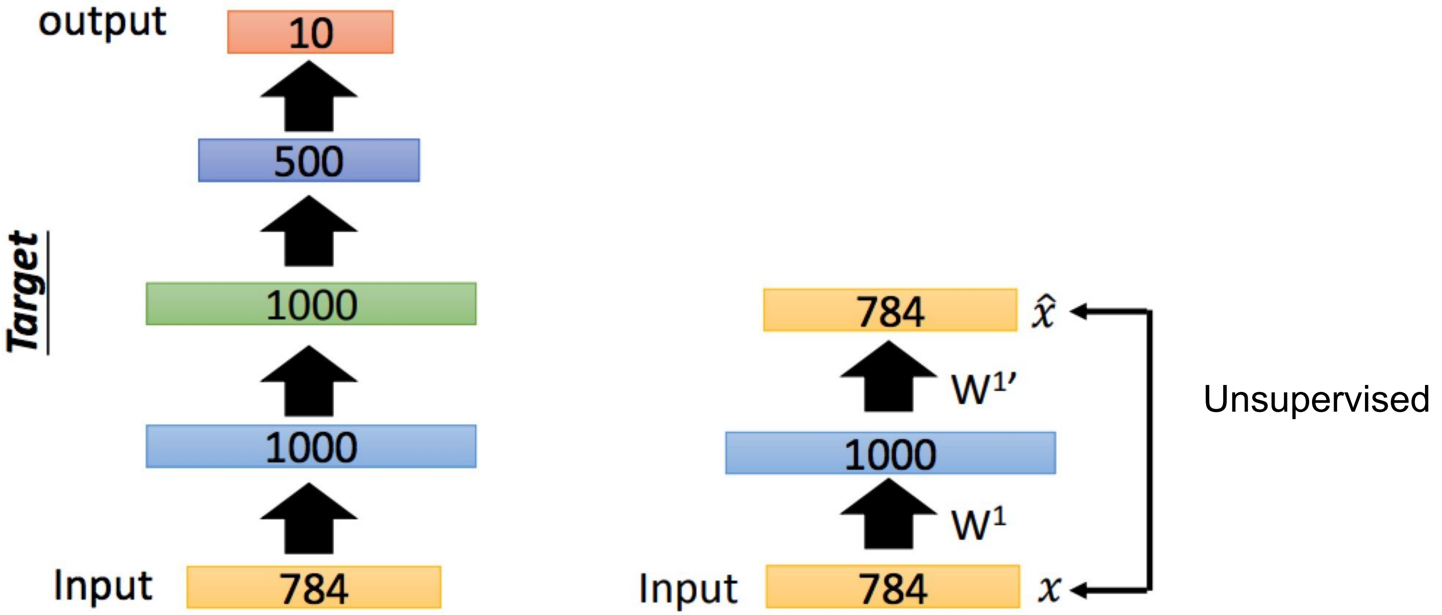




LSA: project documents to 2 latent topics

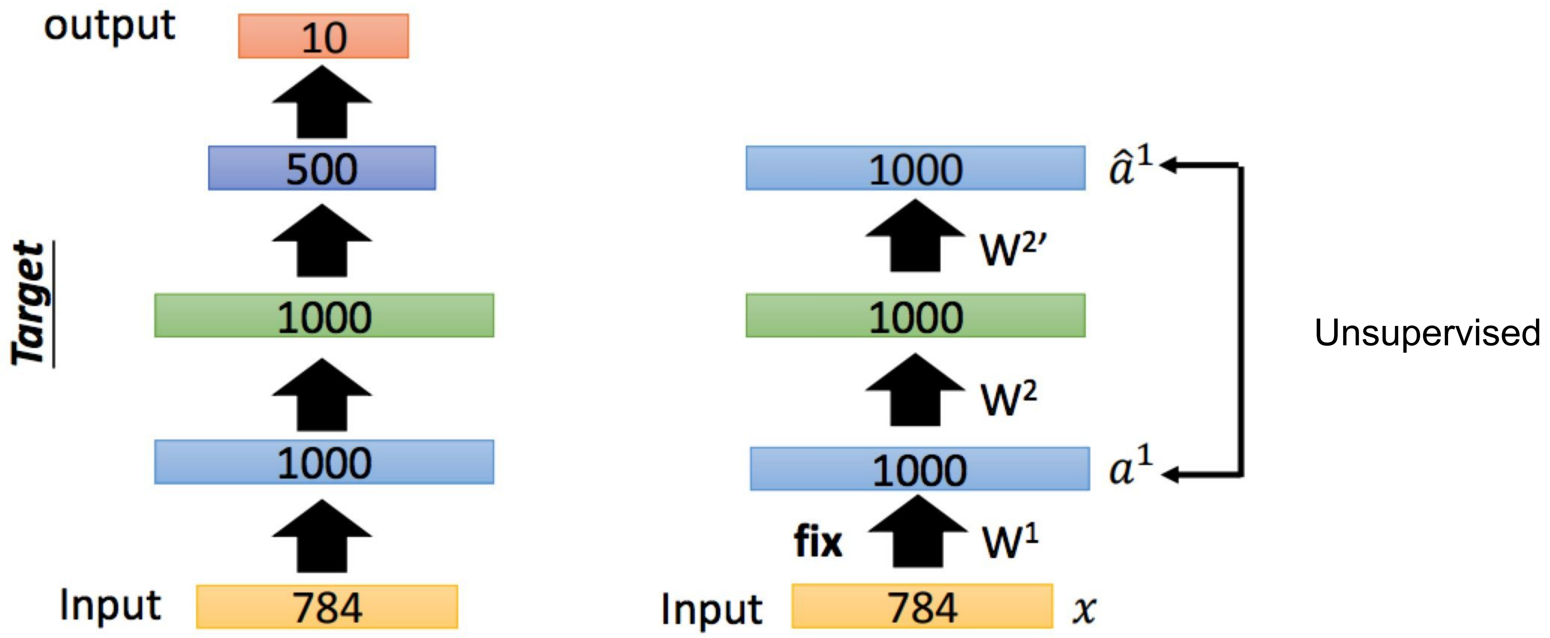
Pre-training Deep Neural Network

- Greedy Layer-wise Pre-training for W1



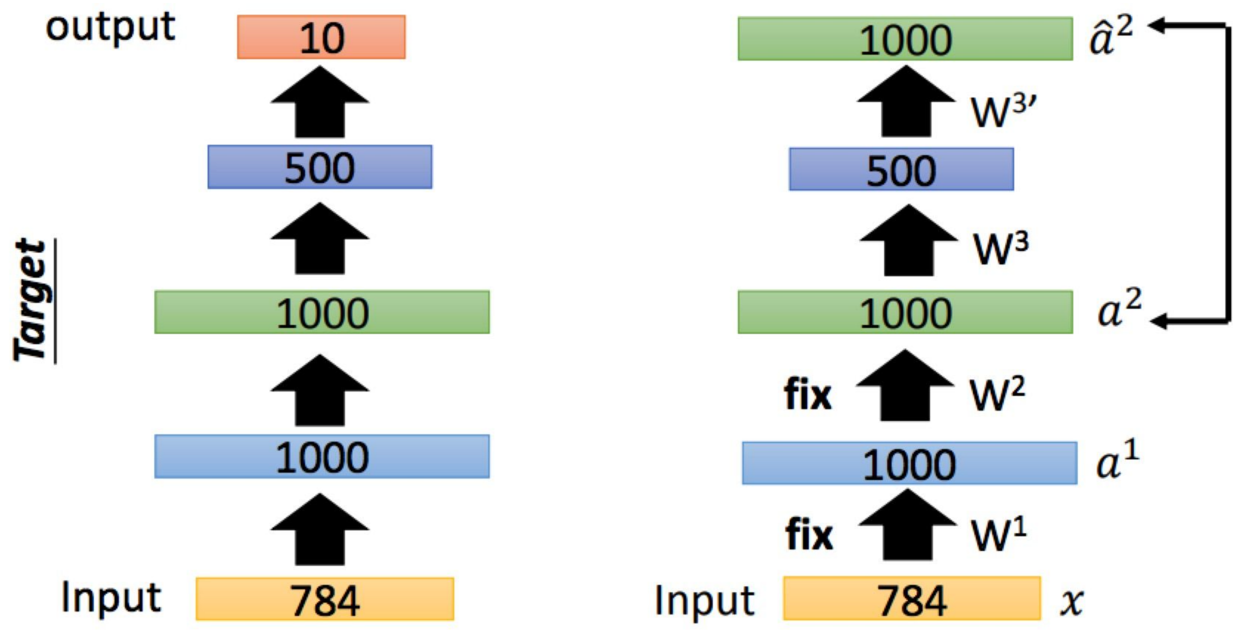
Pre-training Deep Neural Network

- Greedy Layer-wise Pre-training for W2



Pre-training Deep Neural Network

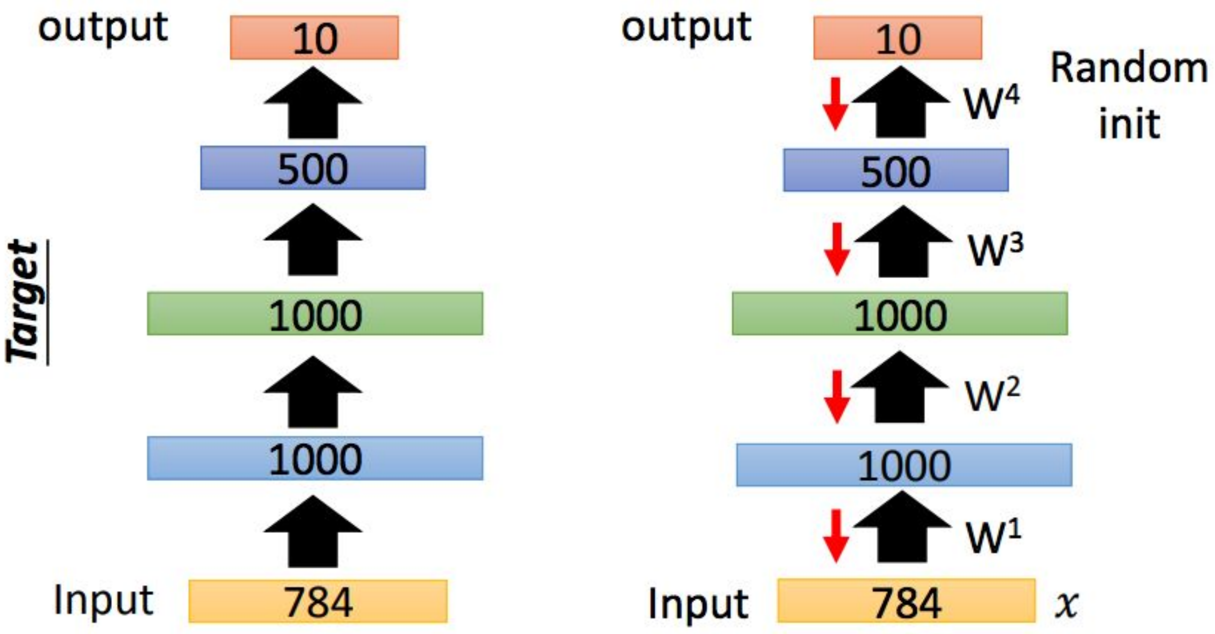
- Greedy Layer-wise Pre-training for W3



Unsupervised

Pre-training Deep Neural Network

- Fine-tune by backpropagation



supervised

Recommendation System

Arjun Narayan
@narayanarjun

Follow

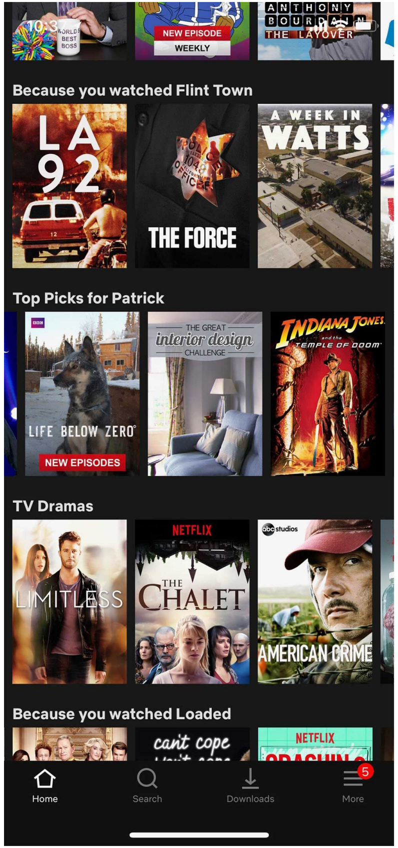
The two best performing public stocks of the decade - Netflix (+3700%) and Domino's Pizza (+3000%) - perfectly epitomize the 2010s. You either build the world's most advanced machine learning content recommender system, or make a better pizza sauce, there's no middle ground.

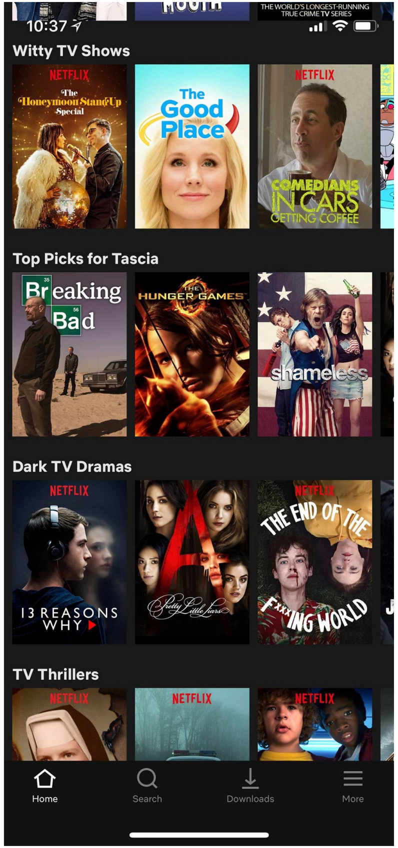
1:20 PM - 27 Dec 2019

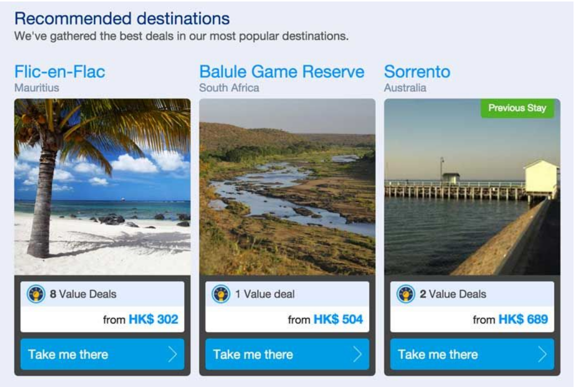
3,926 Retweets **20,086** Likes

183 3.9K 20K

Rec. Sys. are Everywhere

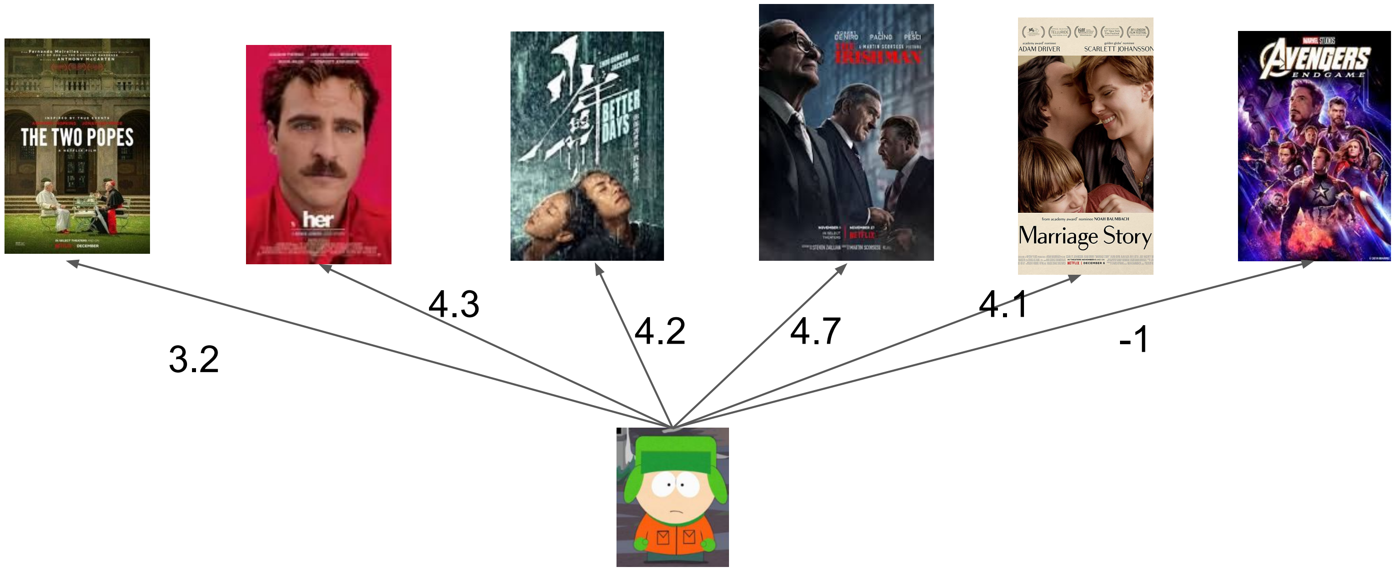






Core Problem in Rec. Sys.

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

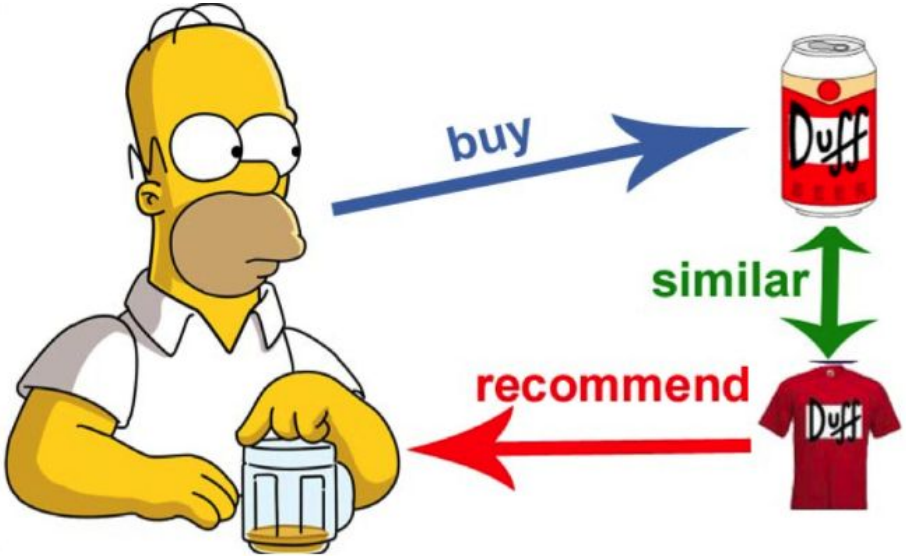


Content-based Method

- Define the similarity from items' content
  - Name: cosine similarity
  - Category
  - Rating
  - Description
  - Etc

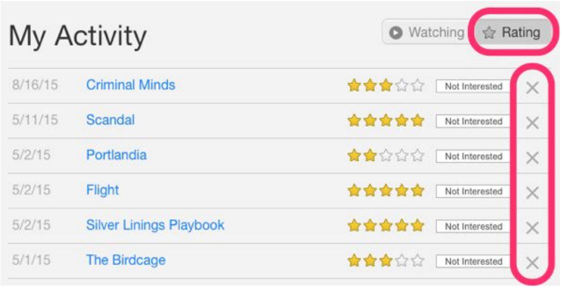
- Combine them into a final score

- Ranked items based on their similar scores compared to users' purchased item

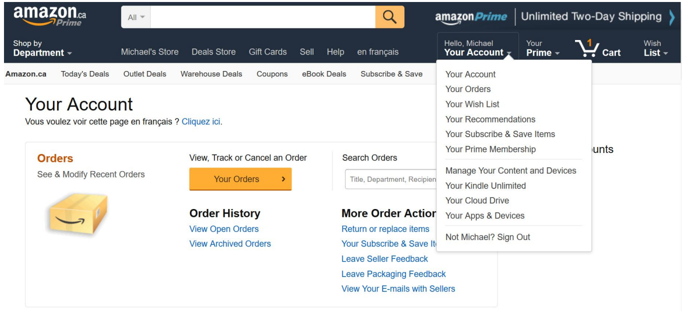


User Behavior

- 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-Item Matrix

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

Item
Vector

User
Vector

	Item 1	Item 2	Item 3	…….	Item k-1	Imte k
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-based CF

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

The rating of item i given by user v

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

The similarity between users u and v

The prediction of an item i for user u

User Space

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

Cosine similarity used a lot in information retrieval

Item-based CF

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

The rating of item m given by user u

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

The similarity between items i and m

The prediction of an item i for user u

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

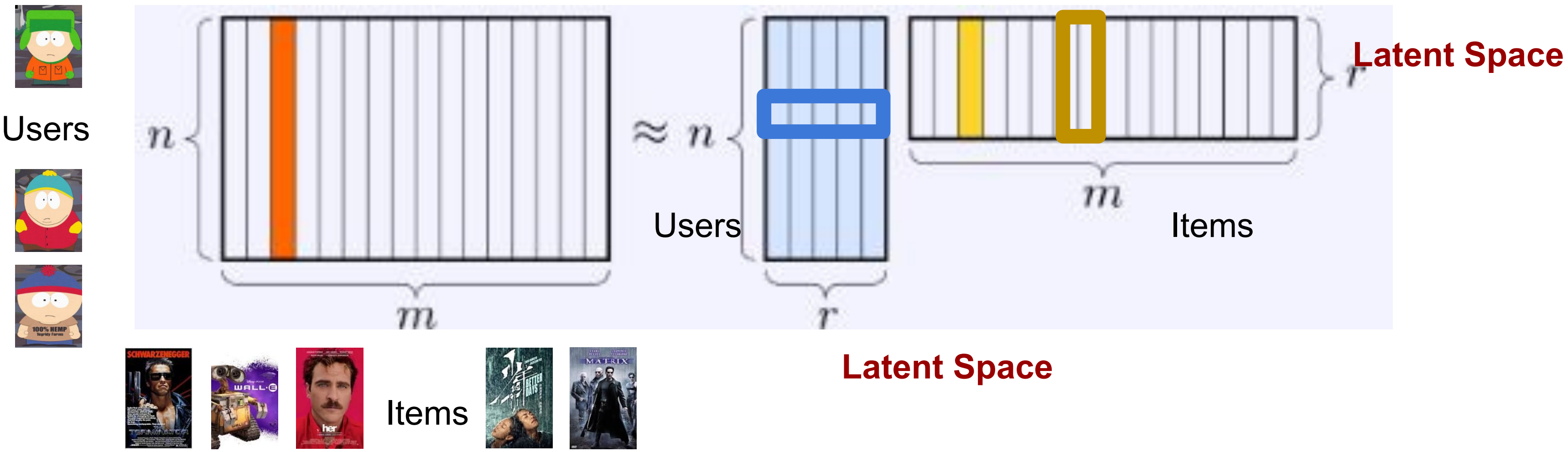
Similarities between users and items are zero

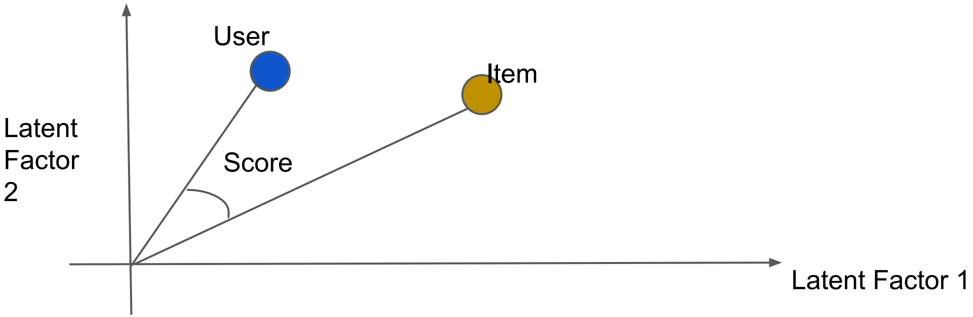
The core problem behind recommendation sys. is to **fill these zero entries**, i.e., *infer the user's preference over the item.*

- Data Preprocessing:
  - Use the mean value of the row
  - Use the mean value of the column

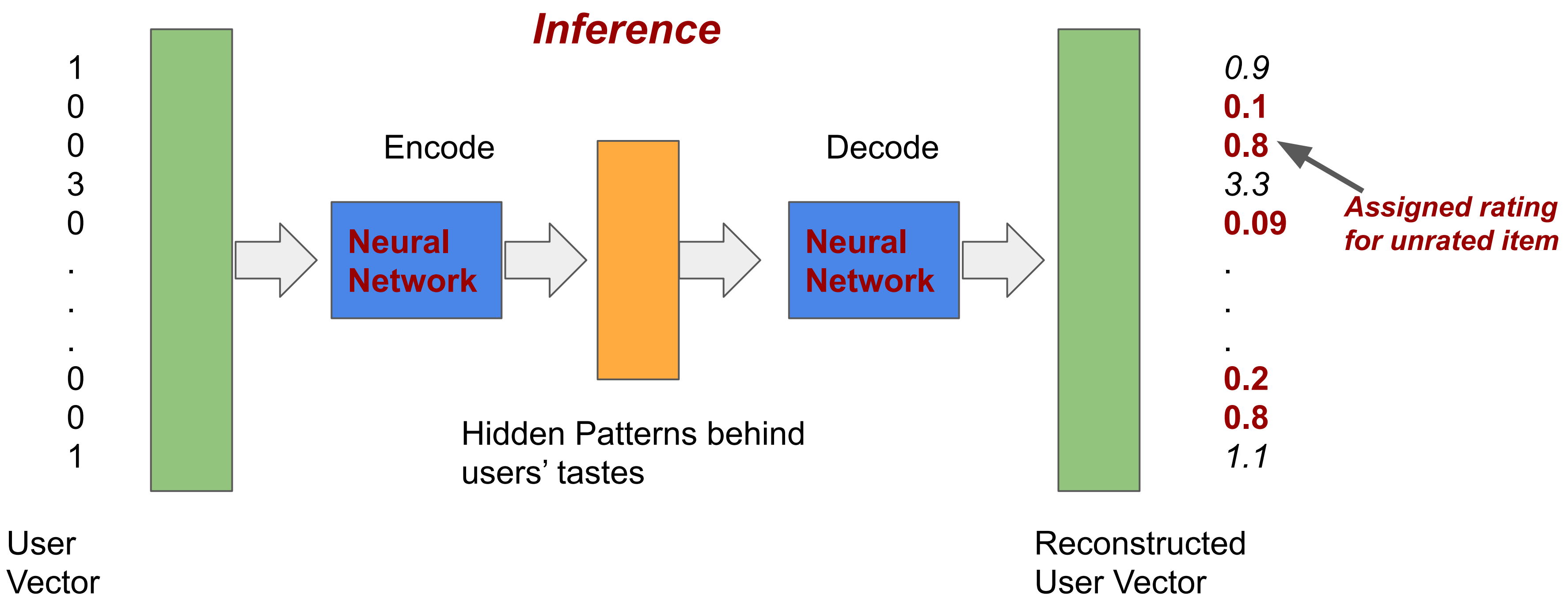
- Matrix Factorization
  - Singular Value Decomposition
  - Non-Negative Matrix Factorization
  - Auto-encoder

NMF for Rec.


Users
n
m
$\approx n$
Users
r
m
Items
r
Latent Space
Items
Latent Space


User
Item
Latent Factor 2
Score
Latent Factor 1

Autoencoder for Rec.



Pros & Cons of CF

- Pros:
  - Capture latent user and item factors
  - Can handle sparsity
  - Scalable computation (ALS)

- Cons:
  - Biases (Temporal and Popularity)
  - Cold Start Problem
  - No Context-awareness

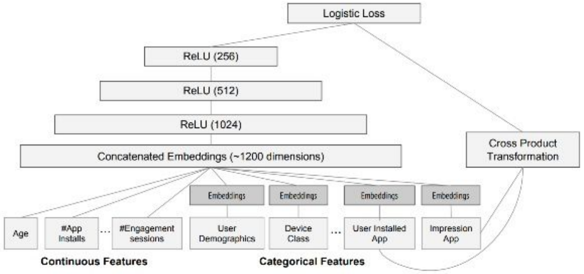
How to evaluate Rec. Sys.

- Offline Evaluation:
    - Train/Test Splitting
    - RMSE
    - Recall
    - Etc

- Online Evaluation:
    - A/B Testing
    - Click-Through Rate (CTR)
    - Conversion Rate (CR)
    - Etc

Advanced Rec. Sys.

- Deep Learning for Rec.:



Wide & Deep model

- Reinforcement Learning for Rec.: