Machine Learning Systems Design

Data Lifecycle

Lecture 9: Feature Engineering (cont.)

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

Agenda

1. Why Feature Engineering
2. Handling Missing Values
3. Feature Scaling and Transformation
4. Feature Selection
5. **Feature Encoding**
6. **Feature Extraction**

5. Feature Encoding

What is feature encoding

Feature encoding is a process of transforming categorical values of features into numerical ones:

- label encoding
- one-hot encoding
- ordinal encoding
- binary encoding
- frequency encoding
- hashing encoding
- positional encoding

Label encoding

It assigns a unique numerical value to each category

	color	length	price
1	brown	1.2	10
2	white	3.1	32
3	white	2	20
4	black	1	10
5	brown	1	9

	color	length	price
1	1	1.2	10
2	0	3.1	32
3	0	2	20
4	2	1	10
5	1	1	9

What is the problem?

Label encoding

Simple and fast, but it may introduce an artificial order among the categories that does not reflect their actual relationship.

One-hot encoding

Creates a new binary feature for each category.

	color_ black	color_ brown	color_ white	length	price
1	0	1	0	1.2	10
2	0	0	1	3.1	32
3	0	0	1	2	20
4	1	0	0	1	10
5	0	1	0	1	9

One-hot encoding

Avoids imposing an artificial order among the categories, but it may increase the dimensionality of the data and cause sparsity issues.

Ordinal encoding

Assigns a numerical value to each category based on some inherent order.

	size	price
1	small	10
2	large	32
3	medium	20
4	small	10
5	small	9

	size	price
1	1	10
2	3	32
3	2	20
4	1	10
5	1	9

Ordinal encoding

Preserves the order among the categories, but it may not capture the exact difference between them.

Binary encoding

Converts each category into binary digits and assigns each digit to a separate column.

	color1	color0	length	price
1	0	1	1.2	10
2	0	0	3.1	32
3	0	0	2	20
4	1	0	1	10
5	0	1	1	9

Binary encoding

It reduces the number of columns needed compared to one-hot encoding.

But, It can lose some information or introduce some correlation between the columns.

Frequency encoding

Replaces each category with its frequency of occurrence in the data.

	city	price
1	NY	10
2	London	32
3	Paris	20
4	Paris	10
5	NY	9

NY: 0.4,
Paris: 0.3,
London: 0.3

	city	price
1	0.4	10
2	0.3	32
3	0.3	20
4	0.3	10
5	0.4	9

Frequency encoding

Reduces the dimensionality of the data and captures the popularity of each category, but it may lose some information about the uniqueness of each category.

Hashing encoding

- Example: you want to build a recommendation system for Amazon
  - There are over 2 million brands that we need to recommend

Hashing encoding

- Example: you want to build a recommendation system for Amazon
  - There are over 2 million brands that we need to recommend

How do we encode the different brands/vendors?

Hashing encoding

- one-hot encoding!

How do we encode the different brands/vendors?

Hashing encoding

- one-hot encoding!

How do we handle a new brand that wants to join Amazon?

Hashing encoding

- one-hot encoding!
- encode unseen brands with “UNKNOWN”

How do we handle a new brand that wants to join Amazon?

Hashing encoding

- one-hot encoding!
- encode unseen brands with “UNKNOWN”

Problem! “UNKNOWN” was not seen during training, so none of the products in this category are being recommended

Hashing encoding

- one-hot encoding!
- encode unseen brands with “UNKNOWN”

Fix – encode brands as themselves, group bottom-performing 1% of brands as “UNKNOWN”

Hashing encoding

- one-hot encoding!
- encode unseen brands with “UNKNOWN”
- Group bottom 1% of brands and newcomers into “UNKNOWN” category

Hashing encoding

- one-hot encoding!
- encode unseen brands with “UNKNOWN”
- Group bottom 1% of brands and newcomers into “UNKNOWN” category

Alert! Nike wants to join Amazon as a new vendor

Hashing encoding

- one-hot encoding!
- encode unseen brands with “UNKNOWN”
- Group bottom 1% of brands and newcomers into “UNKNOWN” category
- Problem – this treats all newcomers the same as unpopular brands on the platform

Hashing encoding

How do we implement a flexible method of handling new brands as they are introduced to our system?

Hashing encoding

1. Represent each category with its attribute
    a. E.g. to represent a brand, use features: yearly revenue, company size, etc..
2. Hashing trick

Hashing encoding

- Hashing – use a hash function to hash categories to different indexes

Hashing encoding

- Hashing – use a hash function to hash categories to different indexes
  - e.g. hash(“Nike”) = 0, hash(“Adidas”) = 27, etc…

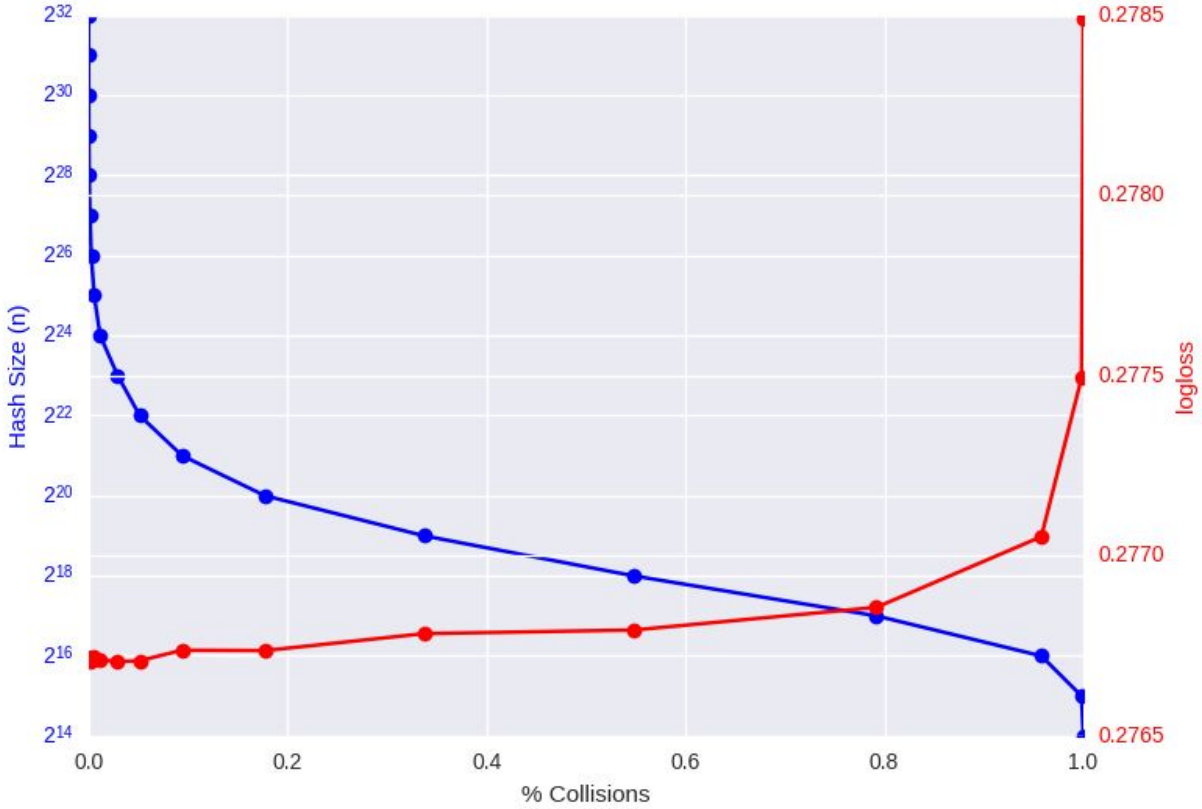
Hashing encoding

- Hashing – use a hash function to hash categories to different indexes
  - e.g. hash(“Nike”) = 0, hash(“Adidas”) = 27, etc…
- Benefits – you can choose how large the hash space is

Hashing encoding

- Hashing – use a hash function to hash categories to different indexes
  - e.g. hash(“Nike”) = 0, hash(“Adidas”) = 27, etc…
- Benefits – you can choose how large the hash space is
- Drawbacks – two categories being hashed to the same index

Hashing encoding



A 50% collision rate only causes the log loss to increase less than 0.5%

Hashing encoding

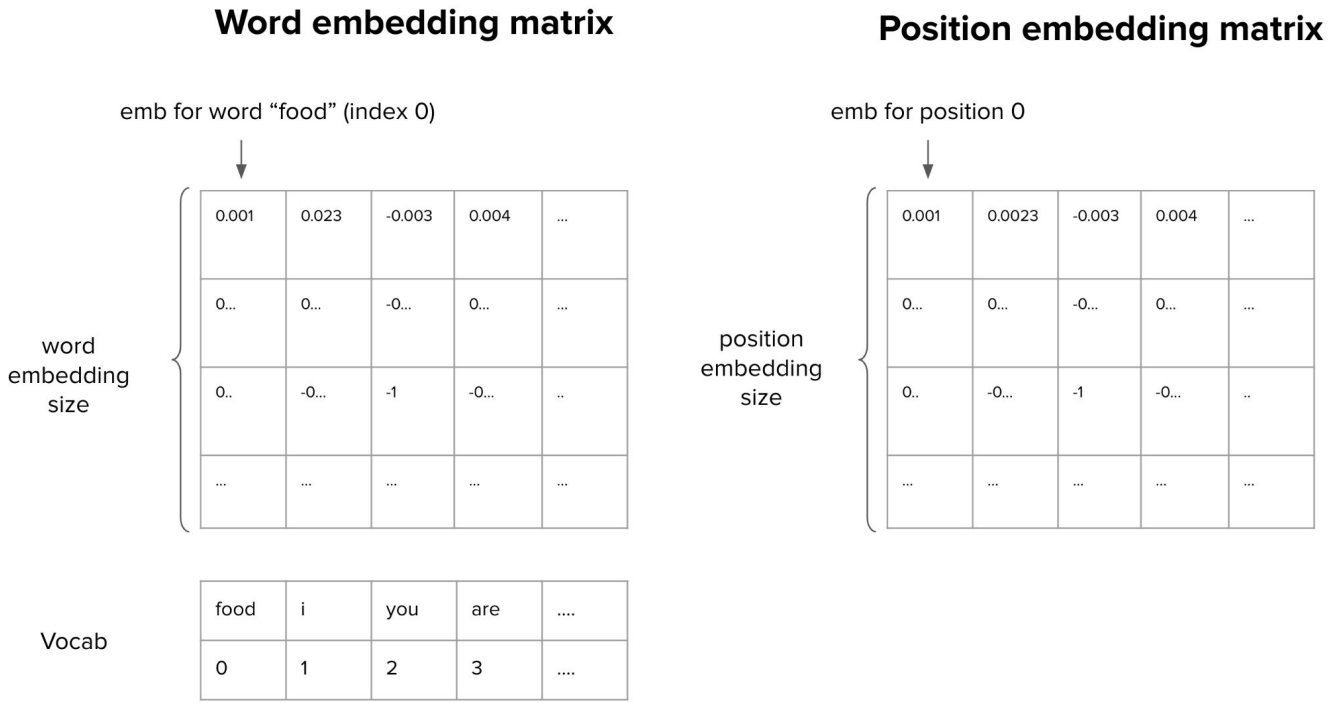
- Choose a hash space large enough to reduce collisions
- Choose functions with properties beneficial to your use case
    - Locality-sensitive hashing

Hashing Trick Takeaways

- Hashing trick considered “hacky” by academics
- Widely used in industry and in machine learning frameworks
- Useful in practice for continual learning in production

Positional Embeddings

- Popularized in *Attention is All You Need* paper
- Similar to word embeddings
  - Can be either learned or fixed



Why do we need position embeddings?

Positional Embeddings

- Traditional architectures (RNNs, LSTMs) process tokens sequentially
- Transformers process tokens in parallel → need to communicate sequential nature of human language to model

Positional Embeddings

The following criteria should be satisfied:

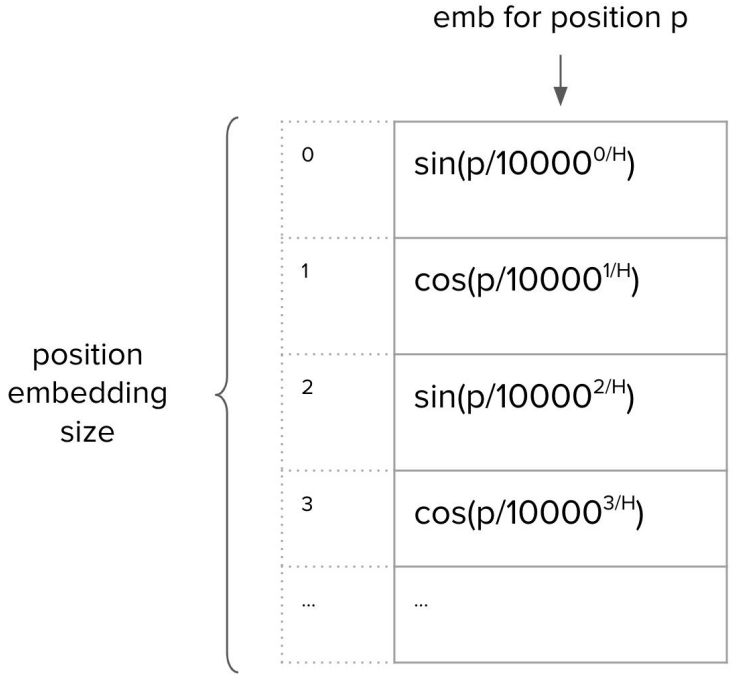
- It should output a unique encoding for each time-step (word's position in a sentence)
- Distance between any two time-steps should be consistent across sentences with different lengths.
- Our model should generalize to longer sentences without any efforts. Its values should be bounded.
- It must be deterministic.

Positional Embeddings

- Fourier features

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$



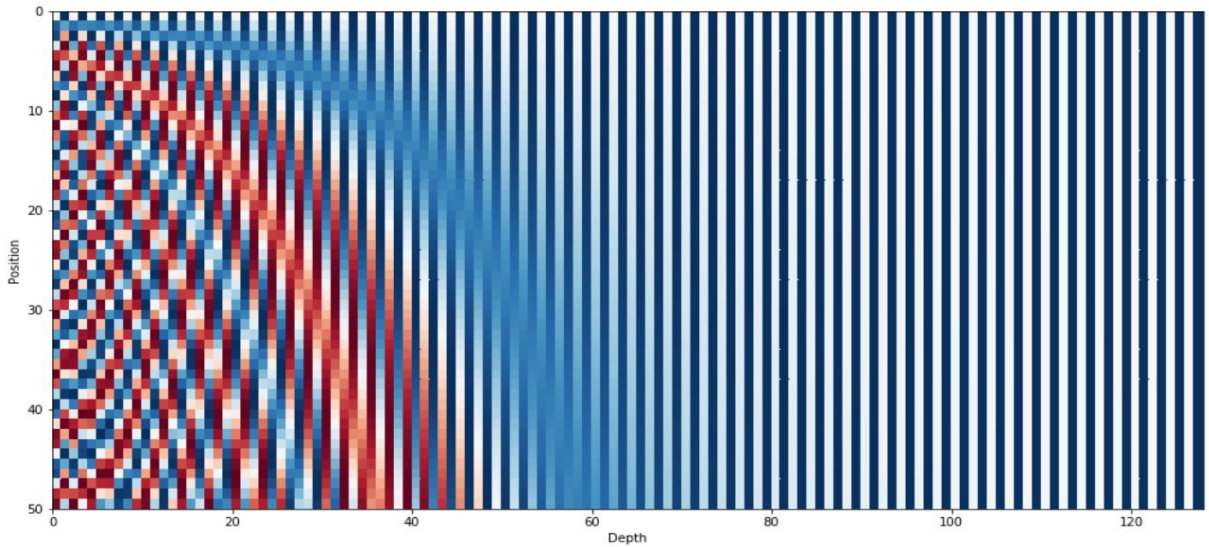
Positional Embeddings

What the Sinusoidal functions

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

Positional Embeddings

What the Sinusoidal functions



Question

How to encode ZIP code?

Answer

- Hashing
- Geocoding
- Target encoding

	Hash Integer Value
belvedere tiburon	582753783
berkeley	1166288024
martinez	-157684639
mountain view	-1267876914
san leandro	1219729949
san mateo	986716290
south san francisco	-373608504

Target encoding

For example, if you have a feature called city and a target variable called salary, you can replace each city name with the average salary of all data points belonging to that city[1]. This way, you can capture some information about the relationship between city and salary in a single number.

Target encoding can help improve the performance of machine learning models that work better with numerical features than categorical features. However, it can also introduce some problems such as overfitting or leakage if not done properly.

target-encoding

Target encoding

- It can cause overfitting or leakage if not done properly
- It can lose some information about the frequency or distribution of categorical features

How to avoid overfitting?

Target encoding

One way is to use cross-validation or nested folds, where you split your training data into several subsets and fit the encoder on one subset and apply it on another subset.

Discretization/bucketizing/binning

- Turning a continuous feature into a discrete feature (quantization)

Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
    - Incorporate knowledge/expertise about each variable by constructing specific buckets

Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
  - Incorporate knowledge/expertise about each variable by constructing specific buckets
- Examples
  - Income
    - Lower income: x < $35,000
    - Middle income: $35,000 < x < $100,000
    - High income: x > $100,000

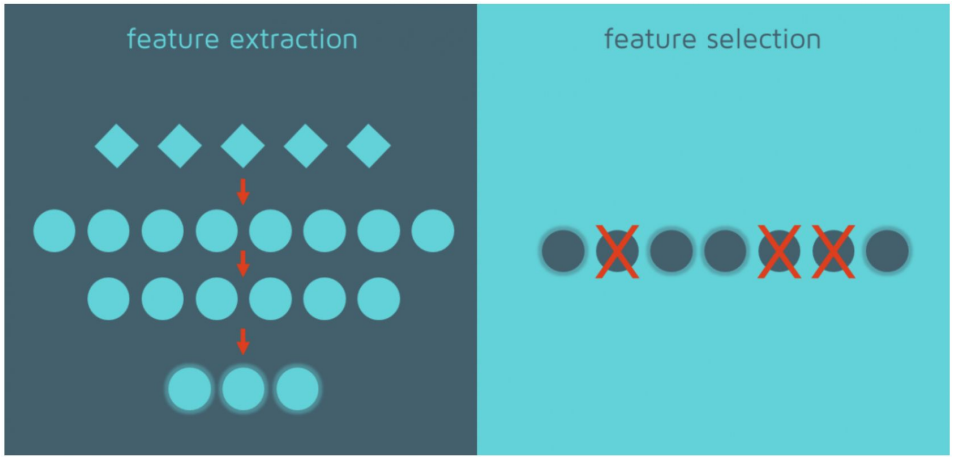
Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
  - Incorporate knowledge/expertise about each variable by constructing specific buckets
- Examples
  - Income
    - Lower income: x < $35,000
    - Middle income: $35,000 <= x < $100,000
    - High income: x >= $100,000
  - Age
    - Minors: x < 18
    - College: 18 <= x < 22
    - Young adult: 22 <= x < 30
    - 30 <= x < 40
    - 40 <= x < 65
    - Seniors: x >= 65

6. Feature Extraction

What

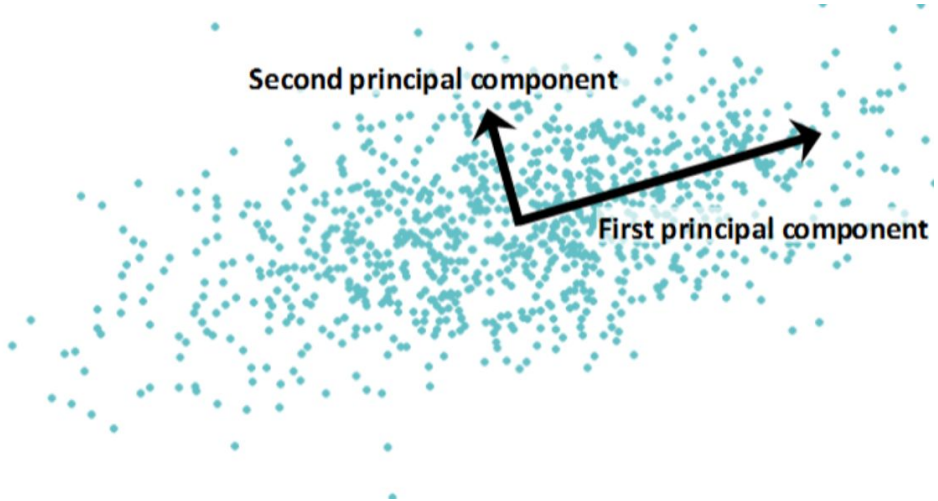
- A process of transforming the original data into a new representation that can be more suitable for machine learning tasks (informative & non-redundant)
- A way of reducing the dimensionality and complexity of data by selecting or creating relevant features.

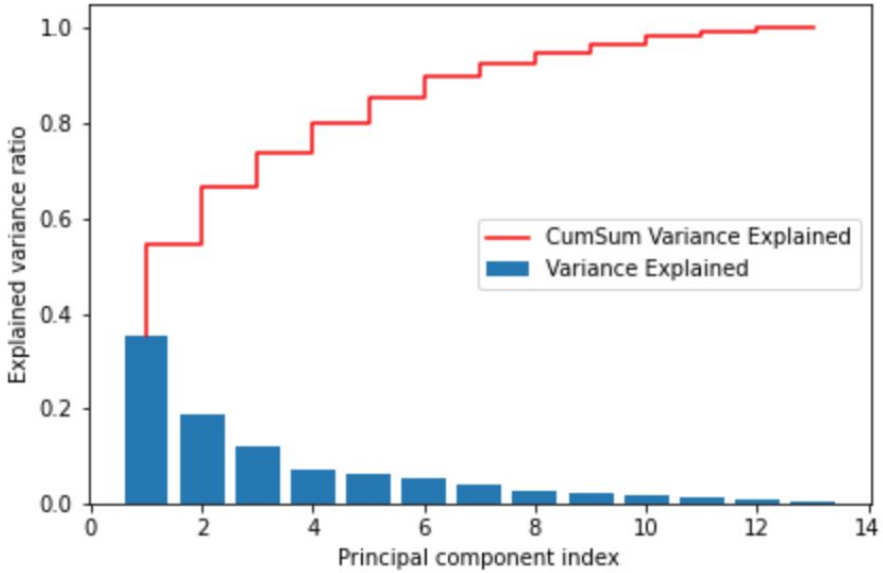


Why

- To improve the performance, efficiency and interpretability of machine learning models.
- To overcome the problems of high dimensionality, such as noise, redundancy, sparsity and curse of dimensionality.

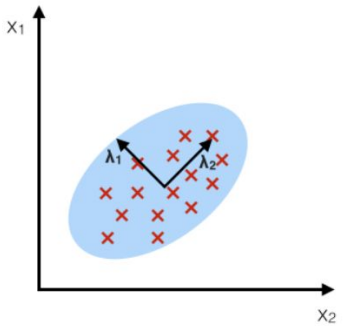
PCA: Principal Component Analysis



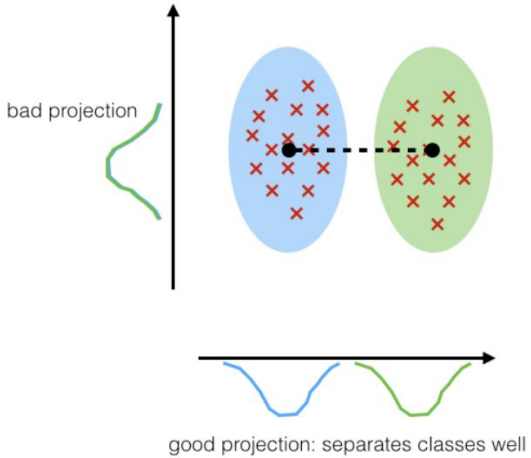


LDA: Linear Discriminant Analysis

PCA:
component axes that maximize the variance



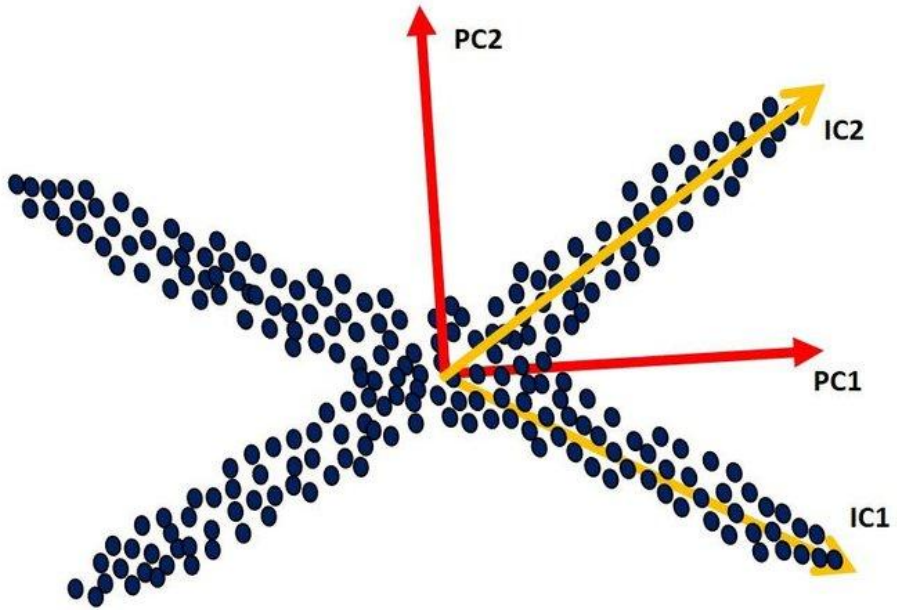
LDA:
maximizing the component axes for class-separation



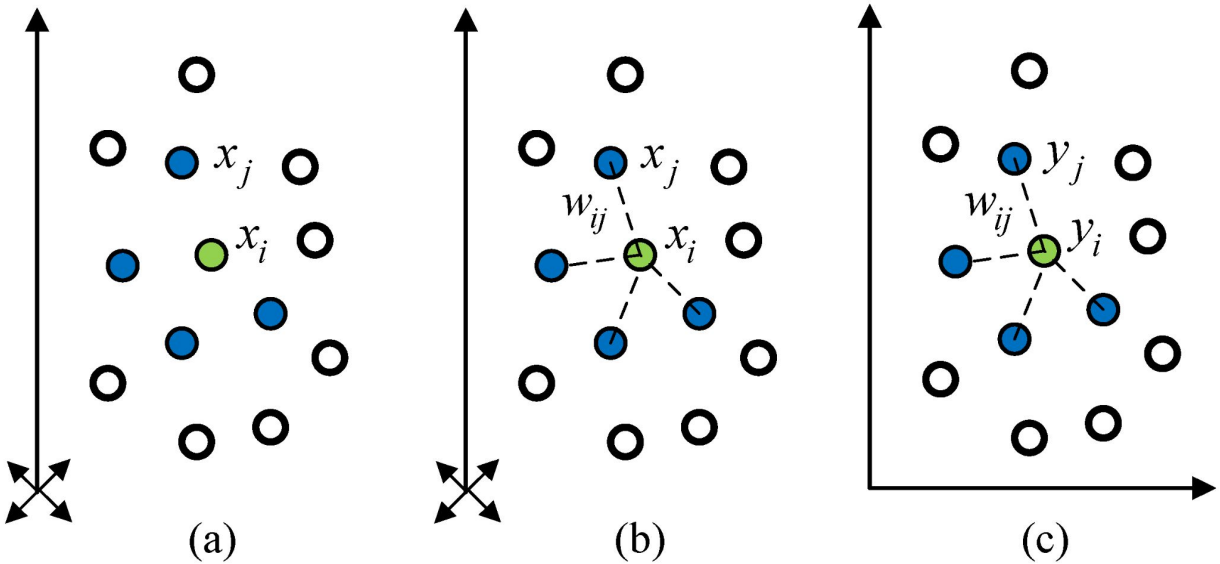
ICA: Independent Component Analysis

PCA: finds directions of maximal variance in gussian data

ICA: Finds direction of maximal independence in nonguassian data

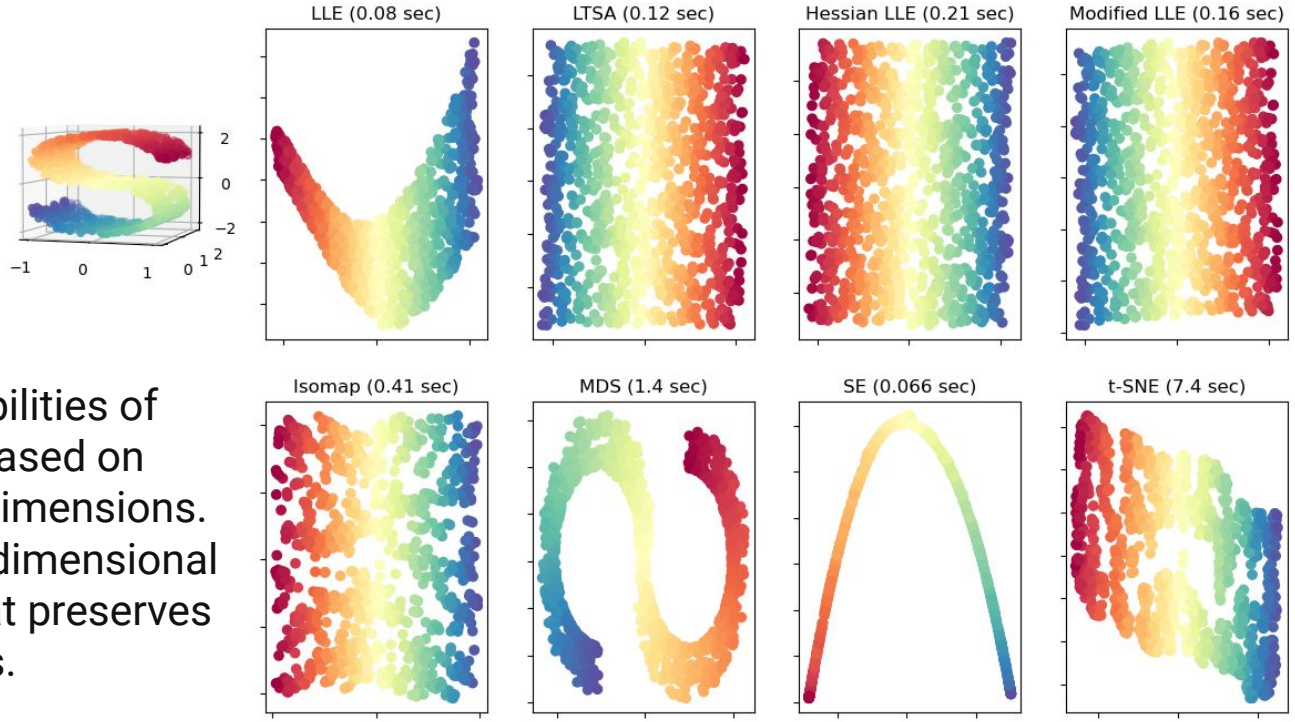


LLE: Locally Linear Embedding



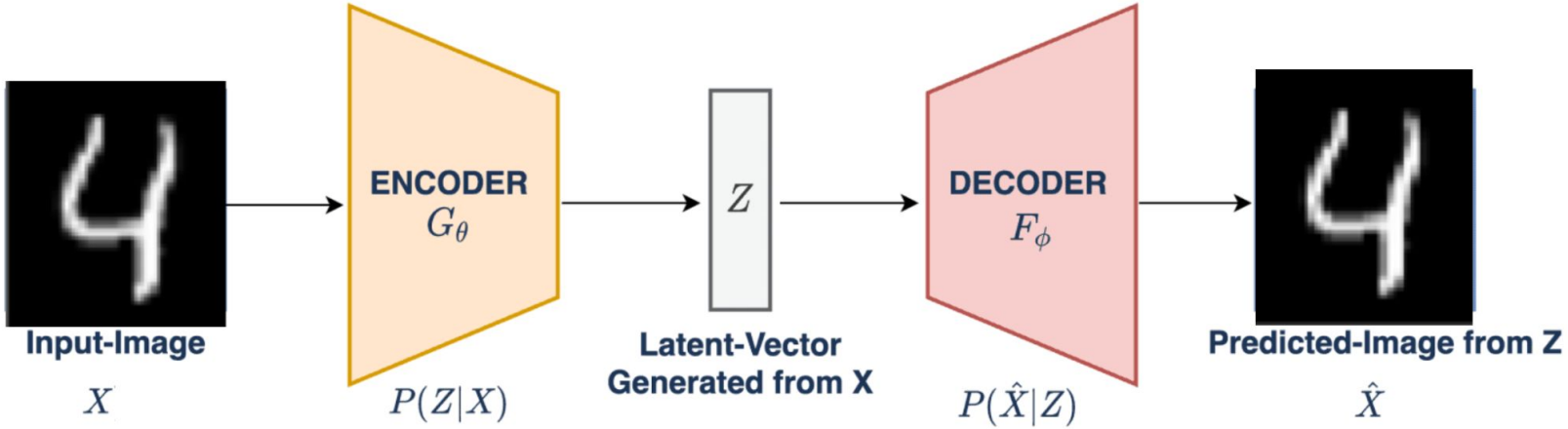
Reducing the dimensionality of data while preserving the local structure of the data

t-SNE: t-distributed Stochastic Neighbor Embedding



Constructs probabilities of choosing points based on similarity in high dimensions. Then, finds a low-dimensional representation that preserves these probabilities.

Autoencoders



Feature Crossing

- Combine two or more features to create a new feature

Marriage	Single	Married	Single	Single	Married
Children	0	2	1	0	1
Marriage & children	Single, 0	Married, 2	Single, 1	Single, 0	Married, 1

Feature Crossing

- Helps models learn non-linear relationships between variables
- Warning – feature crossing can blow up your feature space
  - e.g. Feature A and B both have 100 categories → Feature A x B will have 10,000 categories
  - Need even more data to learn this new feature space
  - Blowing up feature space can increase risk of overfitting

Very common in RecSys & CTR with models like DeepFM and xDeepFM

Machine Learning Systems Design

Data Lifecycle

Next Lecture: Model Development and Training

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io