CHL7001H S1 Applied Deep Learning

Data Preprocessing

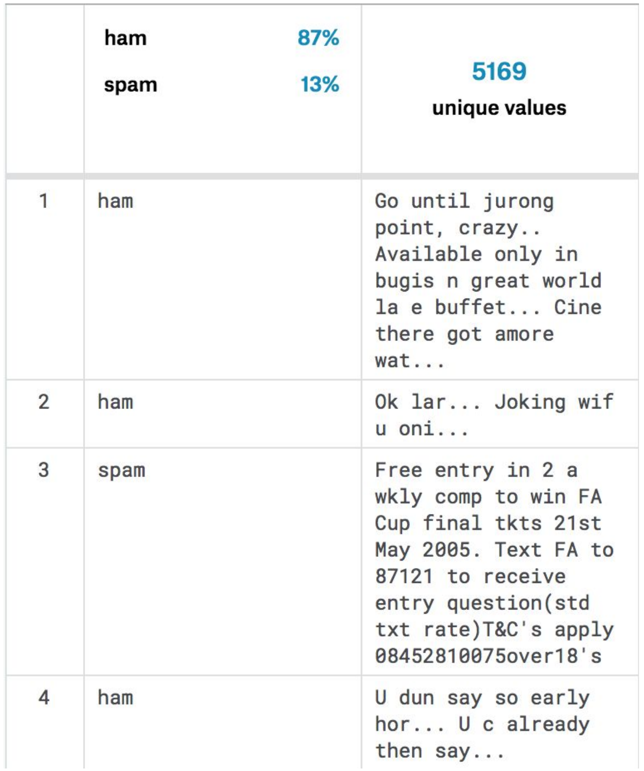
What is data preprocessing

Data preprocessing is a broad technique that involves:

1. **Feature extraction & selection** - convert raw data into a format which the machine learning model can digest and better.
2. **Cleaning** - remove outliers or mistakes
3. **Transformation**

And more……

Feature extraction: SMS spam detection

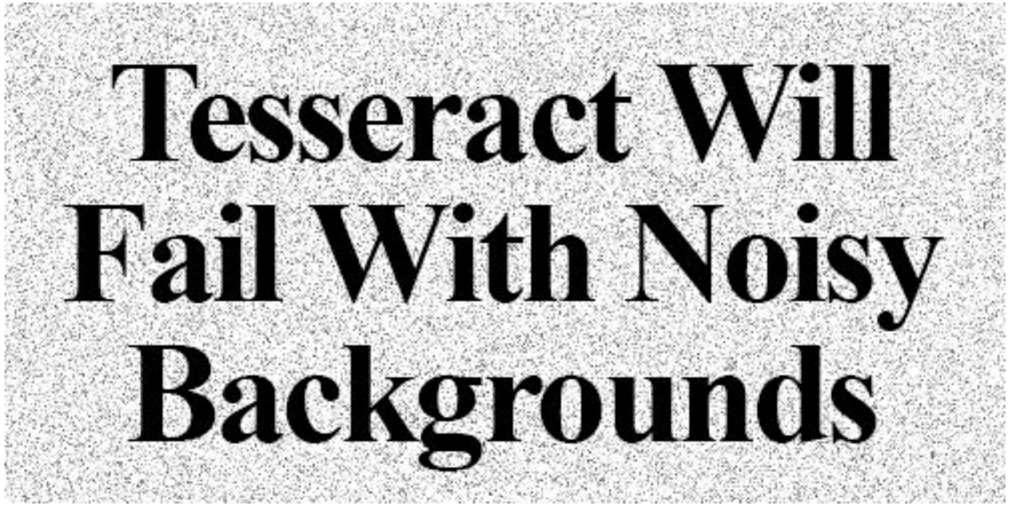


	ham 87% spam 13%	5169 unique values
1	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4	ham	U dun say so early hor... U c already then say...

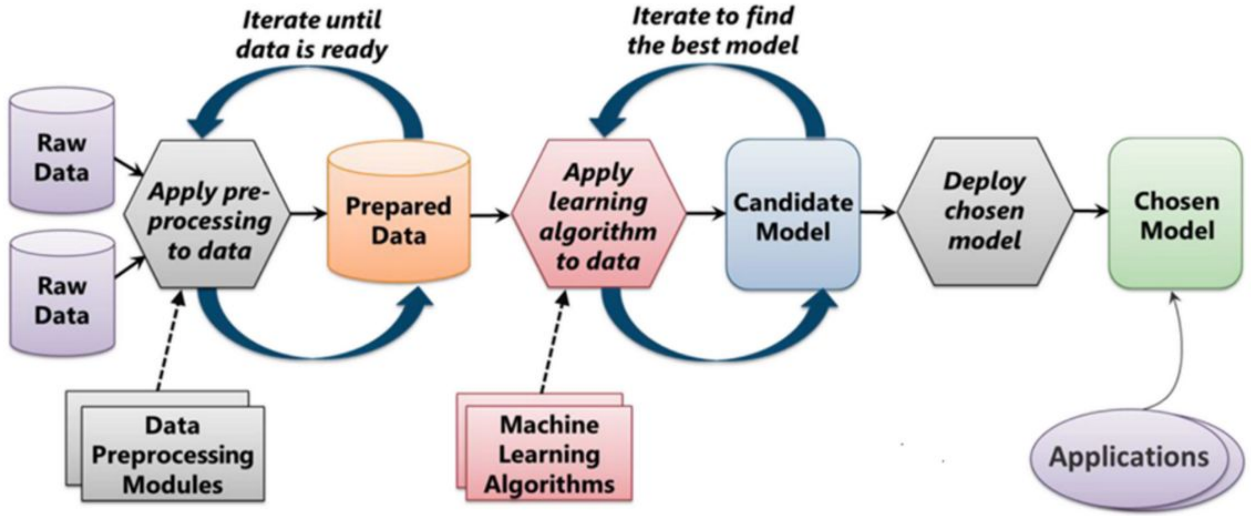
Bag of words (BoW)

neighborhood	7
unit	6
this	4
is	4
a	3
prestigious	1
amenity	1
...	...

Feature cleaning - smoothing images


Tesseract Will Fail With Noisy Backgrounds

The Machine Learning Process


Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

From "Introduction to Microsoft Azure" by David Chappell

Steps in data preprocessing

1. Splitting the data into training and test set before any preprocessing!!!
2. Check out the missing values and outliers
3. Feature scaling
4. See the categorical values

Data splitting

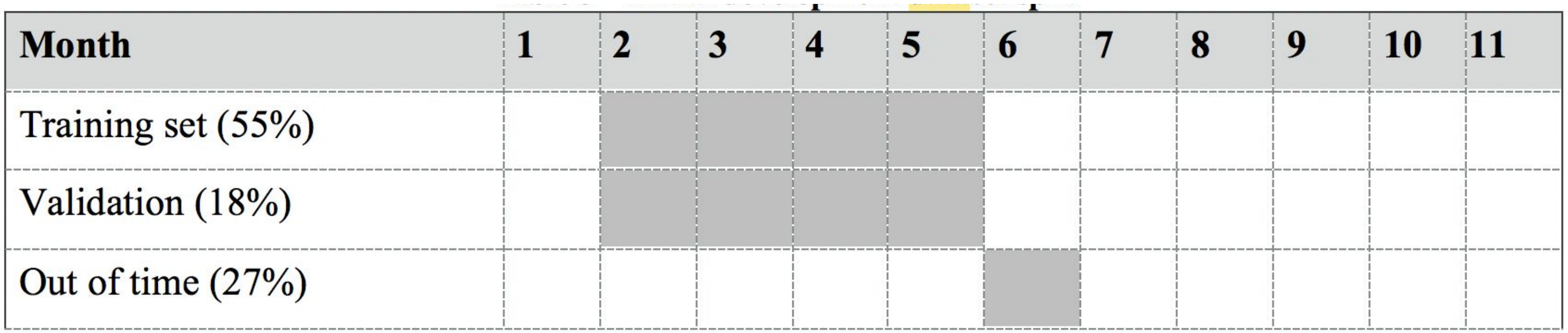
Training, validation, test and maybe more

- Training data - is used to train models.
- Validation data - is used to tune hyperparameters.
- Test data - works like a dry-run to simulate the real-world data performance.
- (Optional) Out-of-time data - to capture unusual time behavior.

And more ……

In production, this Splitting can be varies according to the problems and data accessibility.

Example: recommendation system



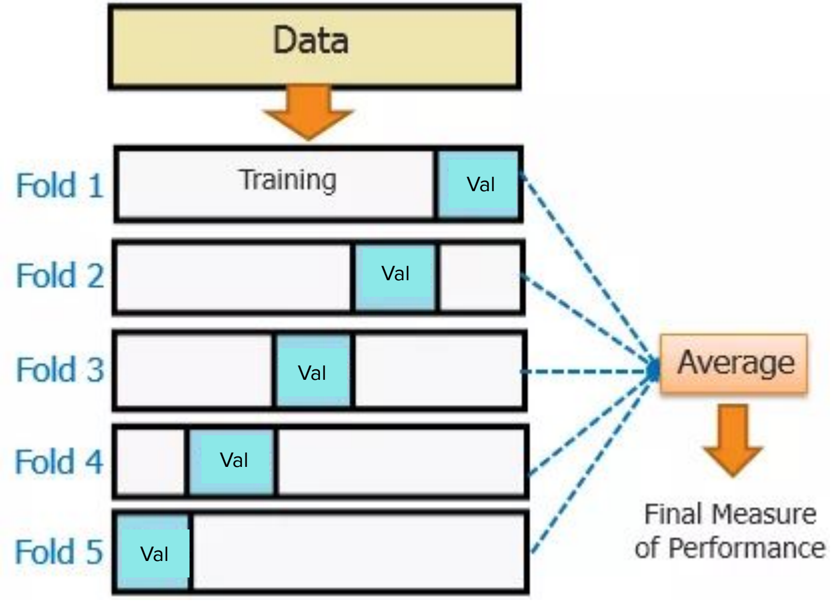
Month	1	2	3	4	5	6	7	8	9	10	11
Training set (55%)		▨	▨	▨	▨						
Validation (18%)		▨	▨	▨	▨						
Out of time (27%)						▨					

Data splitting should be done before any feature cleaning, transformation and scaling!!!!!

Missing data

K-fold cross-validation

- Divide data into K roughly equal-sized parts (K = 5 here).
- In the end, every block of data is used for validating and the results/errors are evaluated based on all folds.
- Extreme case: leave-one-out-cross-validation
- In practical, K = 5/10.



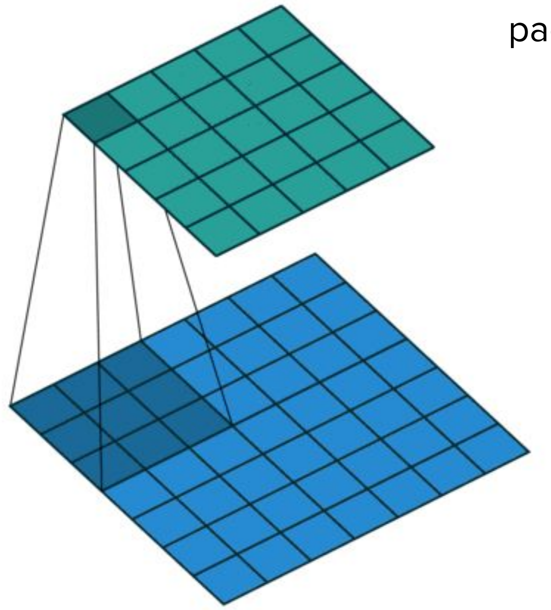
Missing data

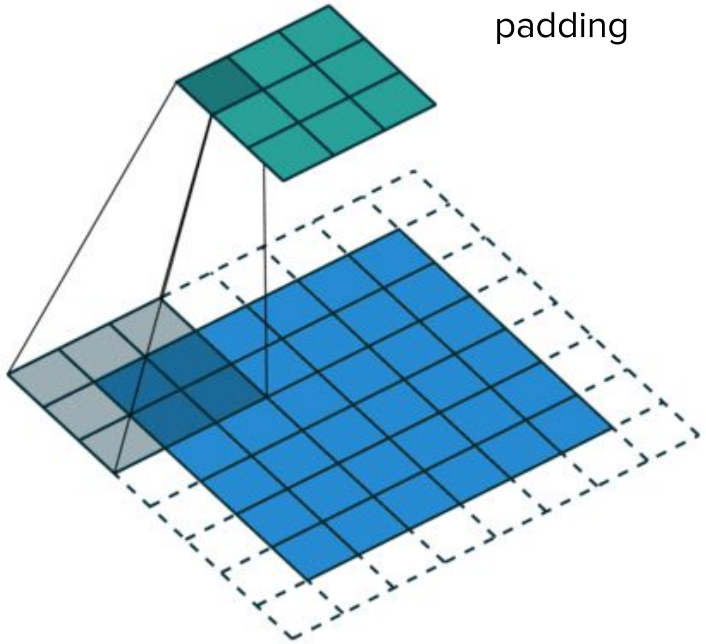
Ways to handle:

Calculate the mean, median of numerical feature or mode of categorical feature and replace it with the missing values.

Don’t delete missing feature!!!! Because you cannot just ignore any incomplete inference data in real production.

Missing data - images or sequences


Stride: 1, no padding


Stride: 2, with padding

Missing data - images or sequences

Sequential data to predict True/False:

[101, 99, 103, 97] >> True		[101, 99, 103, 97, 999] >> True
[102, 98, 105, 93, 111] >> True	$\rightarrow$	[102, 98, 105, 93, 111] >> True
[94, 105, 110, 97, 103] >> False		[94, 105, 110, 97, 103] >> False

After padding ↙ 999

Feature scaling

Feature scaling

Is used to normalize the range of independent variables or features of data.

1. min-max normalization, [0,1]

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. Standardization, mean = 0, std closed to 1

$$x' = \frac{x - \bar{x}}{\sigma}$$

Neural networks are very sensitive to the feature scales. But some models like trees don’t need to bother with various scales.

Categoricals

Categorical values

For example:

- The feature CITY have possible values of { “Toronto”, “Montreal”, “Vancouver”, “Kingston”}
- The feature PROVINCE have possible values of {“Ontario”, “Quebec”, “Alberta”, ...}.

Values are often nominal: no inherent ordering.

Categorical feature encoding

One-hot-encoding (dummy variables)

BUILDING_TYPE	v1	v2	v3	v4
Toronto	1	0	0	0
Montreal	0	1	0	0
Vancouver	0	0	1	0
Kingston	0	0	0	1

CITY =“Toronto” => four columns [1, 0, 0, 0]

Categorical feature encoding - problems

- The feature space can be **extremely sparse** if there are many possible values for those categorical features.
- Consider the postal code as a feature:
    - As of 2014, there are 855,815 postal codes in total in Canada.
    - A naïve one-hot encoding would yield 855,815 dummy variables to encode this one single feature.
- **Extremely high-dimensional feature space**

  => extremely high capacity models

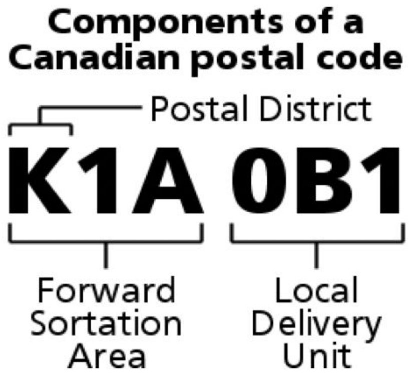
  => overfitting on training data

  => less model generality.

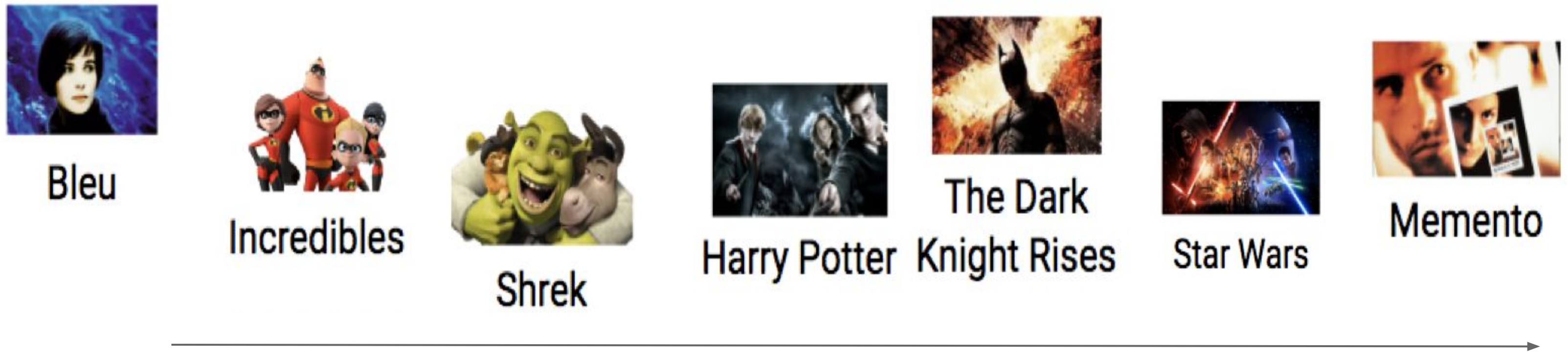
Categorical feature encoding - solutions

Strategies to overcome the sparsity issue with one-hot encoding:

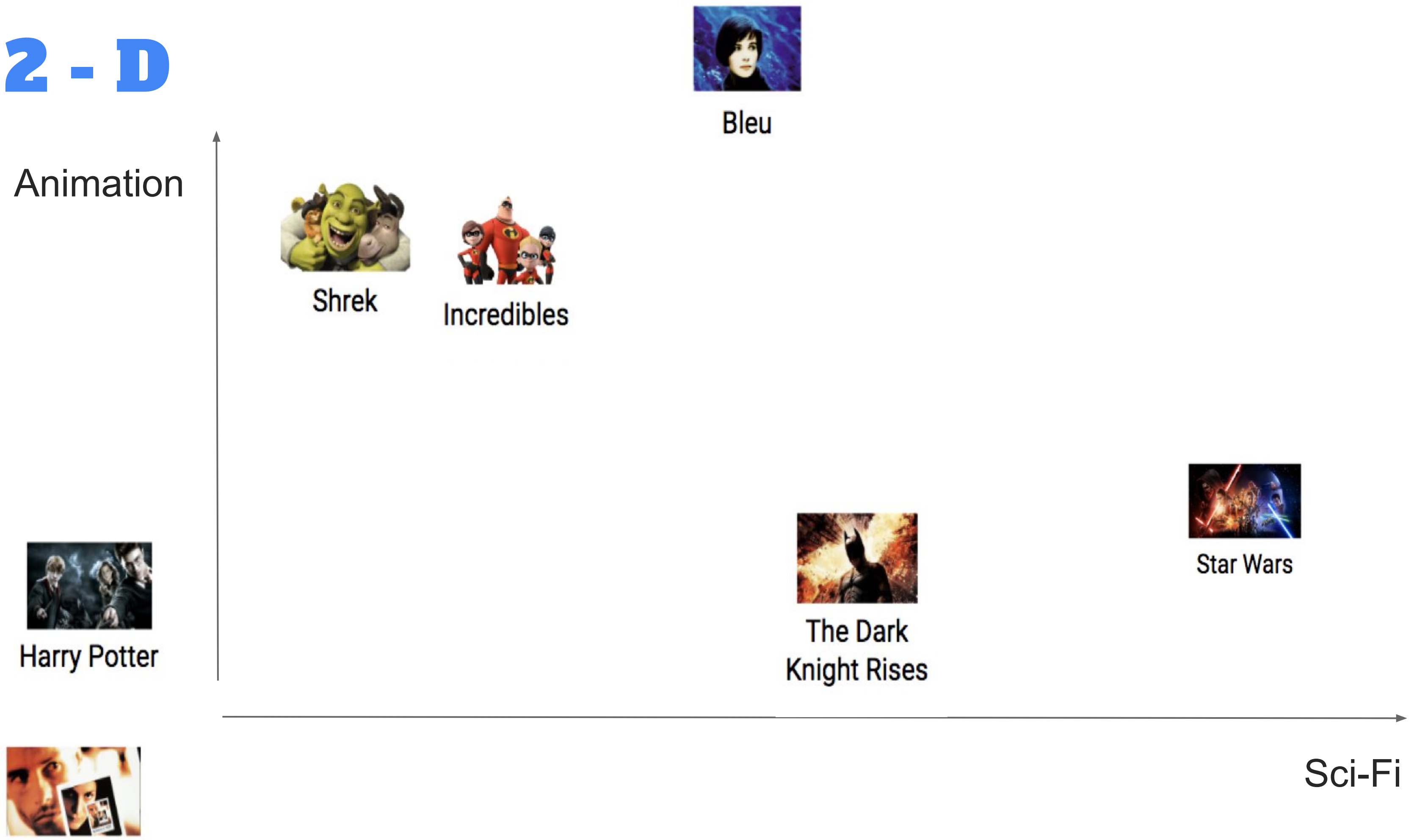
1. Encode frequent features only.
   - Example: group any postal codes with occurrences fewer than 100 times into one big group "**OTHERS**".
2. If values have natural hierarchy, **reduce the granularity** by rolling up low-level values to higher levels in the hierarchy.
   - Example: group postal codes by their first three digits.
3. Embeddings



1-D example


Bleu
Incredibles
Shrek
Harry Potter
The Dark Knight Rises
Star Wars
Memento

2 - D


Bleu
Animation
Shrek
Incredibles
Star Wars
Harry Potter
The Dark Knight Rises
Sci-Fi

3 - D vector values

	Shrek	Harry Potter	Memento	Star Wars
[i,_,_]	0.335	-0.121	-0.482	0.277
[_,i,_]	-0.109	0.152	0.241	0.803
[_,_,i]	0.115	-0.261	-0.263	-0.167

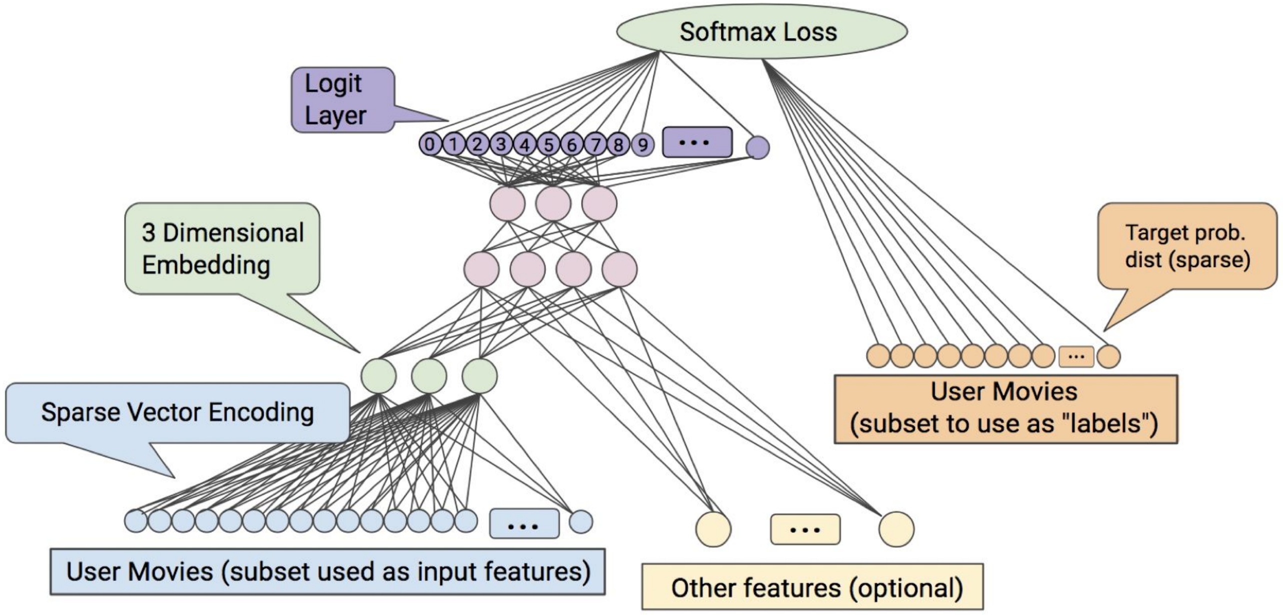
3 - D vector values

	Shrek	Harry Potter	Memento	Star Wars
[i,_,_] Comedy	0.335	-0.121	-0.482	0.277
[_,i,_] Sci-Fi	-0.109	0.152	0.241	0.803
[_,_,i] Animation	0.115	-0.261	-0.263	-0.167

Embeddings - translate to low-dim space

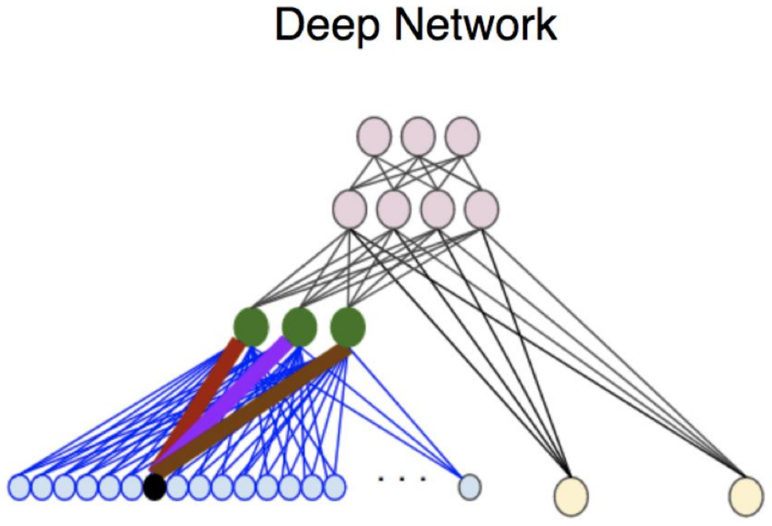
Another strategies to overcome the sparsity issue with embeddings: mapping of a discrete — categorical — variable to a vector of weights.

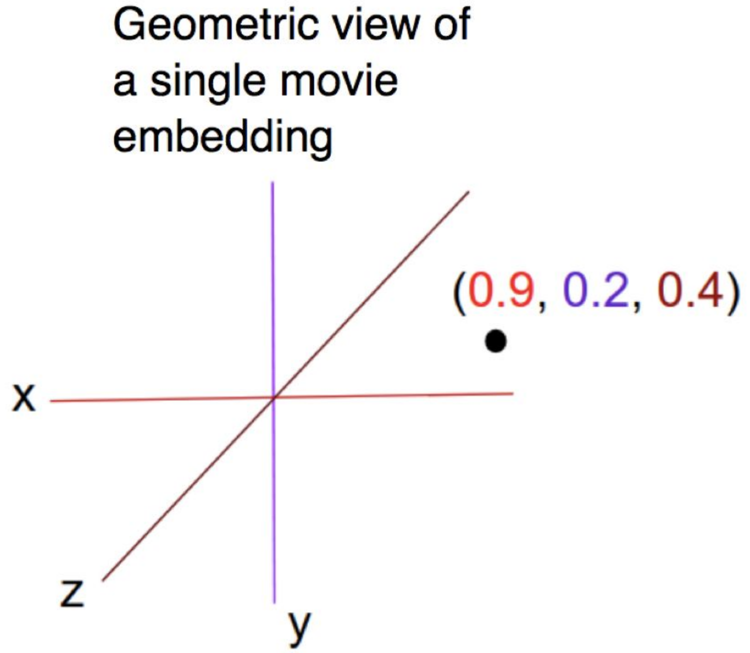
Embedding layer in a network


Softmax Loss
Logit Layer
0 1 2 3 4 5 6 7 8 9 ...
3 Dimensional Embedding
Target prob. dist (sparse)
...
User Movies (subset to use as "labels")
Sparse Vector Encoding
...
User Movies (subset used as input features)
...
Other features (optional)

You can train them inside a neural network as a weight matrix separate from the hidden layers

Embedding layer in a network





Learn Embedding weights in a network

- No separate training process needed – the embedding layer is just a hidden layer
- Supervised information tailors the learned embeddings for the desired task
- Intuitively the hidden units discover how to organize the items in the d-dimensional space in a way to best optimize the final objective

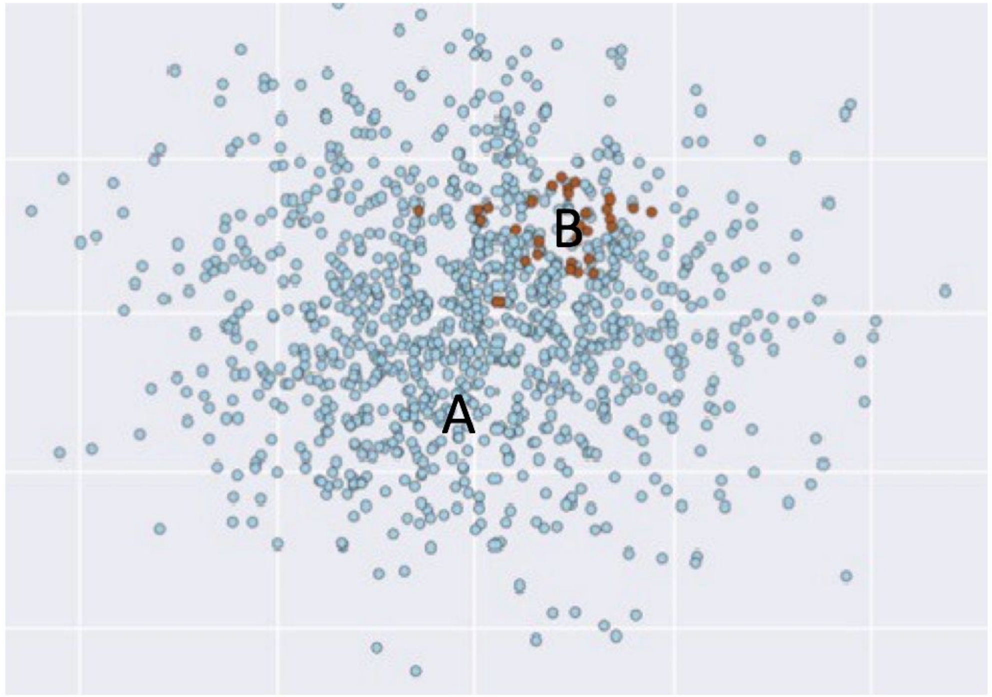
Empirical rule-of-thumb

- Higher dim embeddings can more accurately represent the relationships between input values
- But more dims increase the change over **overfitting** and leads to **slow training**
- Should experiment in each task

$$\text{dim} \approx \sqrt[4]{possible\ values} + 1$$

Resamplings

Imbalanced data



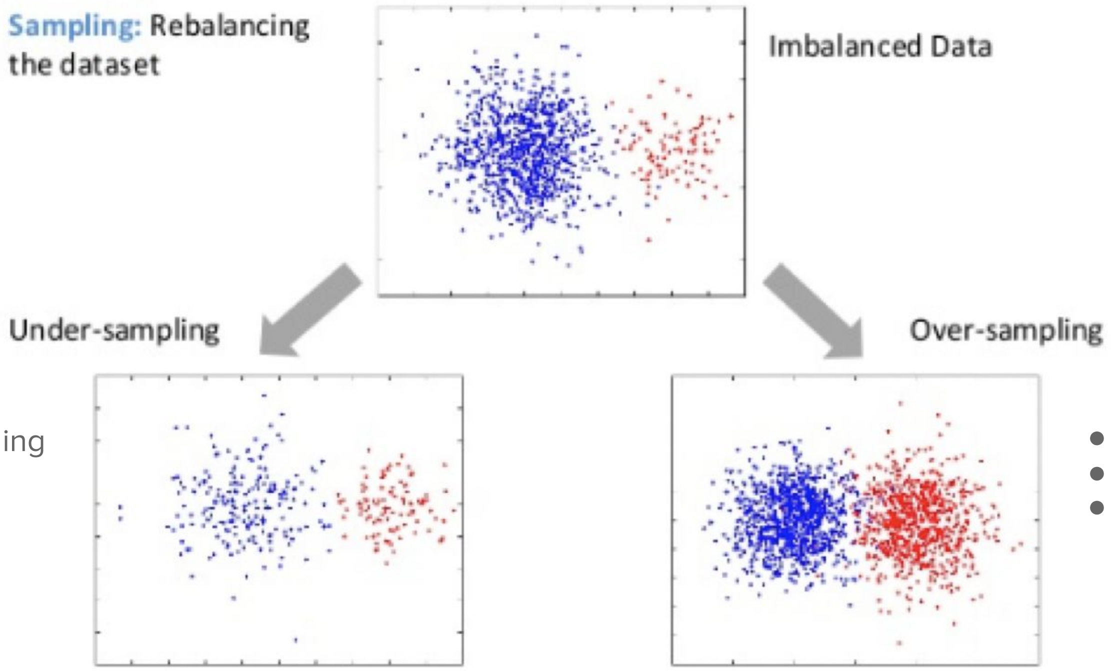
Scenarios:

- rare diseases diagnosis
- credit card frauds
- user churn

Challenges:

- biased predictions
- misleading accuracy

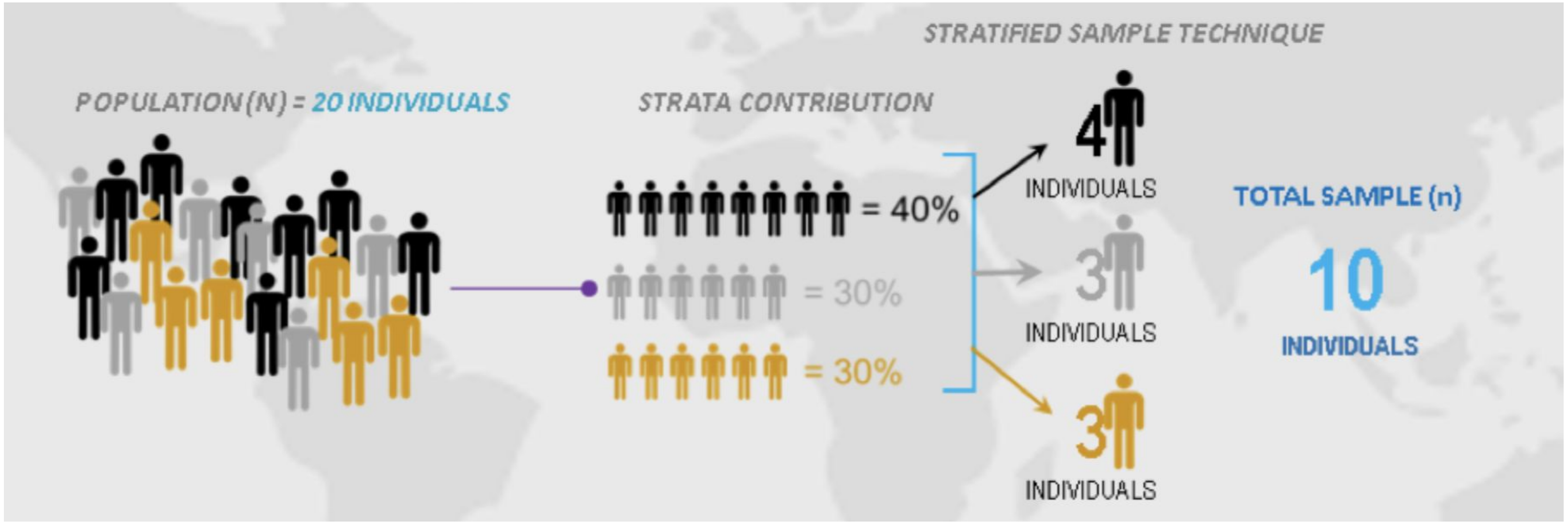
Resampling strategy



- Random undersampling
- Cluster centroid under-sampling
- Tomek's links

- Random oversampling
- SMOTE
- ADASYN

Stratified sampling



SOURCE: towardsdatascience

Lessons I have leanred

Bias introduced from feature selection
Data leakage - future facing data