Data Preprocessing

11-485 / 685 / 785
Recitation 0J

Agenda

Part A:

1. What is data preprocessing?

Part B:

1. Why do we preprocess data?
2. How do we preprocess data?

Part C:

1. Preprocessing Speech | MFCCs

Agenda

Part A:

1. What is data preprocessing?

Part B:

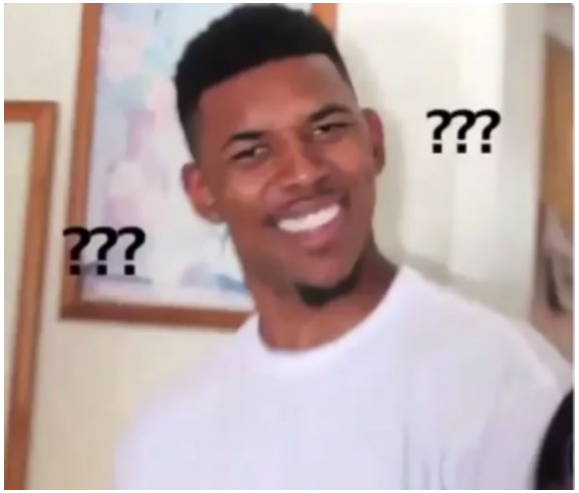
1. Why do we preprocess data?
2. How do we preprocess data?

Part C:

1. Preprocessing Speech | MFCCs

Defining with memes

When you feed raw data into your model



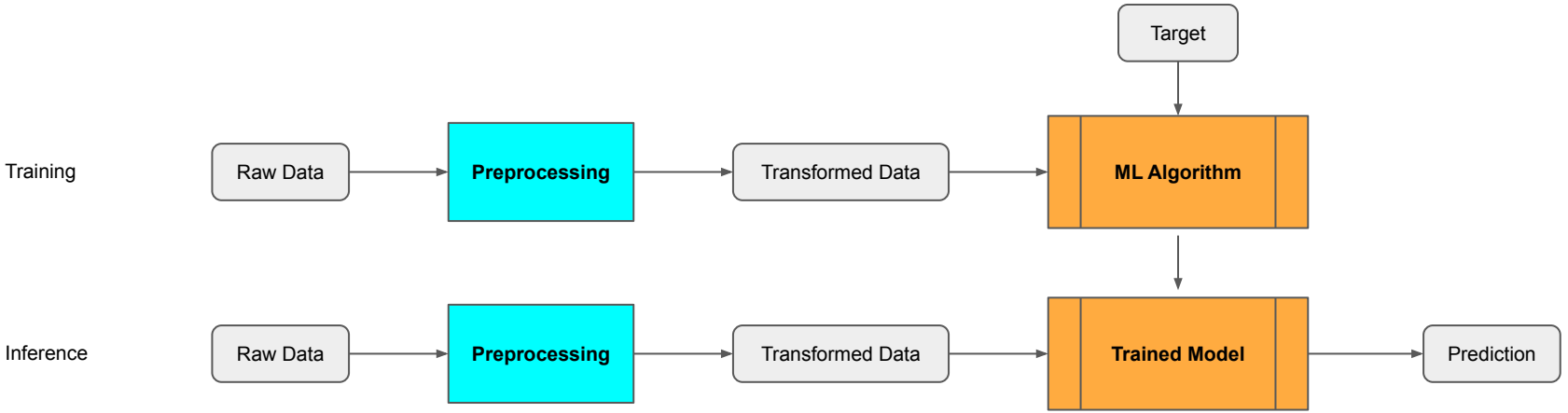
Raw
Data

Preprocessed
Data

So - What is it, in non-meme terms?

Data preprocessing is a step that takes raw data and transforms it in preparation for another processing pipeline, in our case, machine learning.

This is where the ‘pre’ comes from, it takes place before any kind of model training or inference.



Never Forget!

Always split your data into training and testing **before** any preprocessing happens.

Q: What would happen if we don’t do this?

A: Leakage!

More on this later.

Data Preprocessing

11-485 / 685 / 785
Recitation 0J - Part B

Agenda

Part A:

1. What is data preprocessing?

Part B:

1. Why do we preprocess data?
2. How do we preprocess data?

Part C:

1. Preprocessing Speech | MFCCs

Data can..

1. be full of outliers, and can have uneven ranges of values;
2. have redundant, unnecessary features;
3. be in a format that can’t be processed by machines;
4. have a severely uneven distribution of classes;
   (classes = labels; something that needs to be learned)
5. have incomplete rows;

And many other such problems that make it hard for the model to learn effectively.

Data Preprocessing can help us here!

Data Cleaning - Dealing with Data Quality

1. Some features might have values that are naturally very large, while others are naturally small, and this might result in the former getting undue importance over the latter;
   a. By **scaling** both to be restricted in a fixed range, usually [0,1], we can curb this;
   b. You can also **normalize** your data, i.e. impose a normal distribution over the values;

2. Data collected from different sources might have mismatched data types;
   Eg: financial data from different countries will have different currencies;

3. Missing data is a very common characteristic. It can be dealt with by either
   a. Imputing values from other rows
   b. Ignoring the row altogether (only when you have sufficient data)

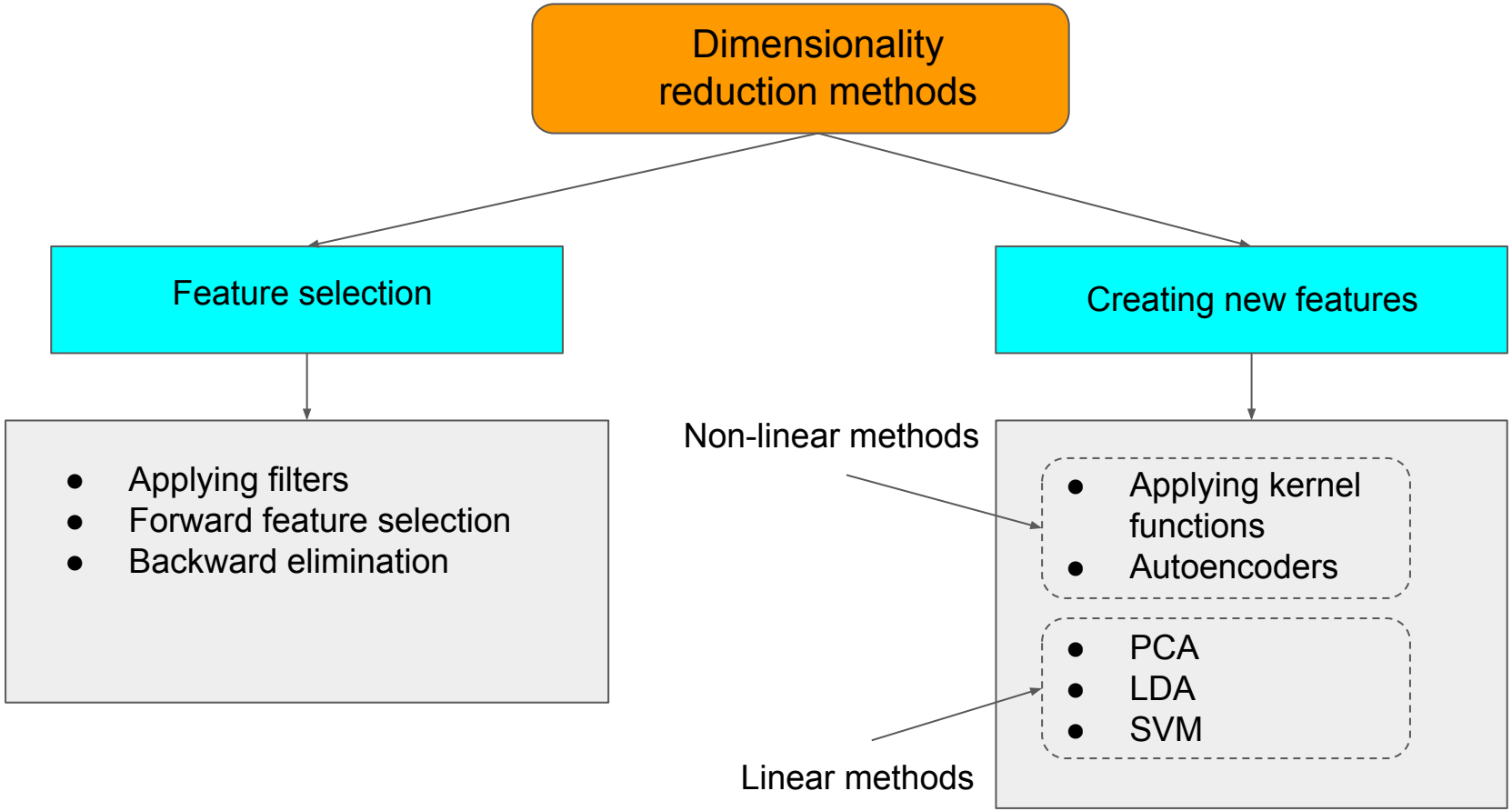
4. Text and speech often needs to be cleaned up
   URLs, Stopwords, “umm”, “err” etc

Dimensionality Reduction

Problems with high dimensional data

- Not all features are equally important
  a. Multicollinear features
  b. Noisy attributes
- If Data is too large (in dimensionality) it's worthwhile trying to keep only the features of most importance
  a. Curse of dimensionality → generally sparse data points → model does not generalize well
- Several Ways to do it:
  a. One of the most prevalent methods is to use feature correlation (with the target variable) as a proxy for "importance" and remove features below some threshold.

Dimensionality Reduction



Dealing with Data Leakage

Data leakage is observed when some input features are correlated with output or it has part of output while training which won’t be available during inference

Methods

- Remove features that contain information about the target that normally wouldn’t be available during inference
- Have a separate validation dataset to check the model performance while training
- Use k-fold cross validation when data is less
- Normalized training data separately and then use those parameters for test dataset

Categorical to Numerical

Machine can only interpret numerical values so mapping categorical data to numeric values before training is important

Methods

- Directly mapping classes to unique numbers
  (NOTE: might introduce undesired bias due to ordinality)
- Create One Hot Encoded Vector
- Use Feature Hashing → for large dataset or data encryption

Dealing with class imbalance

Uneven distribution of classes eg. 10 instances of class A and 90 instances of class B

Method

- Oversampling or undersampling
- Try to have realistic (sometimes uniform) distribution of classes over data
- Have weighted loss if data is highly imbalanced

How to Preprocess?

Various python libraries have support for data preprocessing

PyTorch

Data Preprocessing

- Random sampling, weighted random sampling
- Filters for cleaning data (using lambda functions)
- Random splitting and shuffling of data

Data Augmentation

- Torchvision - for image data
- Torchaudio - for audio data

Implemented while creating Dataloader class (Pytorch document)

Scikit-learn, Pandas, Numpy

- Random data splits, applying filters, normalizing features, etc.

Agenda

Part A:

1. What is data preprocessing?

Part B:

1. Why do we preprocess data?
2. How do we preprocess data?

Part C:

1. Preprocessing Speech | MFCCs

Let’s “talk” about speech.

MFCC

Mel Frequency Cepstral Coefficients

A little detour into signal processing

- Speech signals can be understood as a mixture of signals at various frequencies

- Raw speech signals aren’t always the best source of data for models to work with
  - Contain noise
  - Naturally continuous data that needs discretization at the minimum

- Reckless discretization can lead to noisy frequency bands
  - especially the high ones

MFCC

Mel Frequency Cepstral Coefficients

MFCC

Mel Frequency Cepstral Coefficients

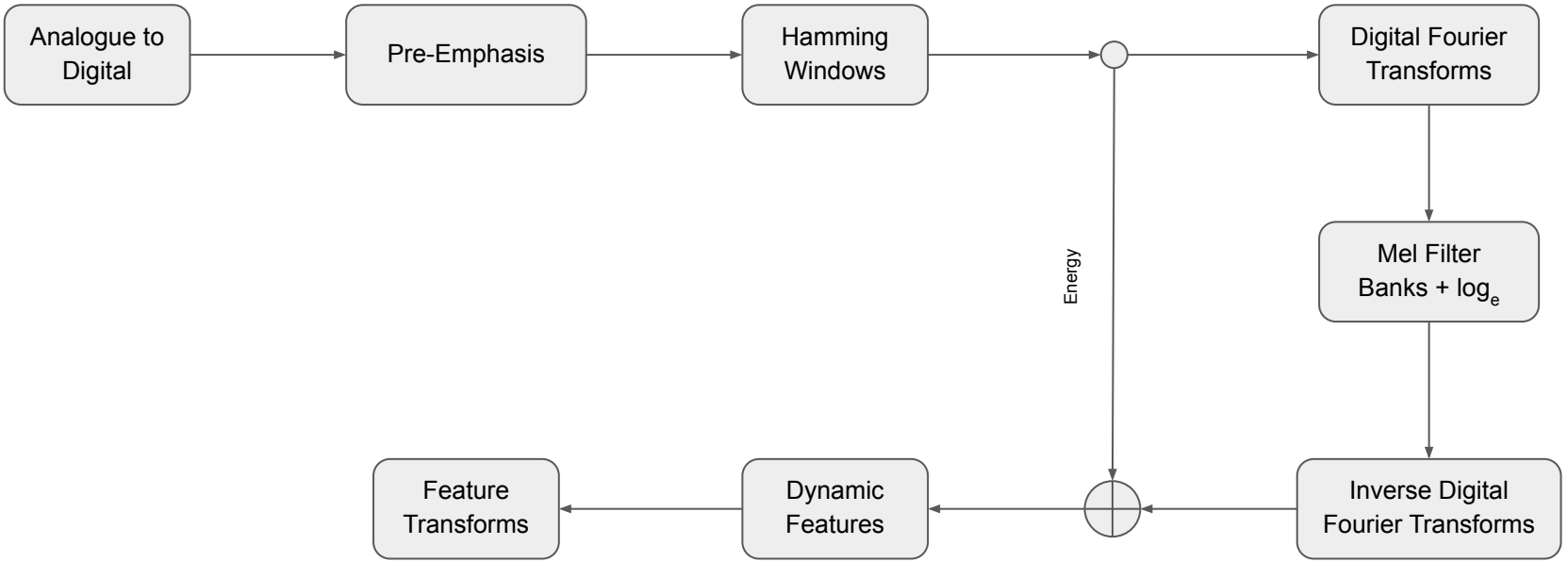
MFCC

Mel Frequency Cepstral Coefficients

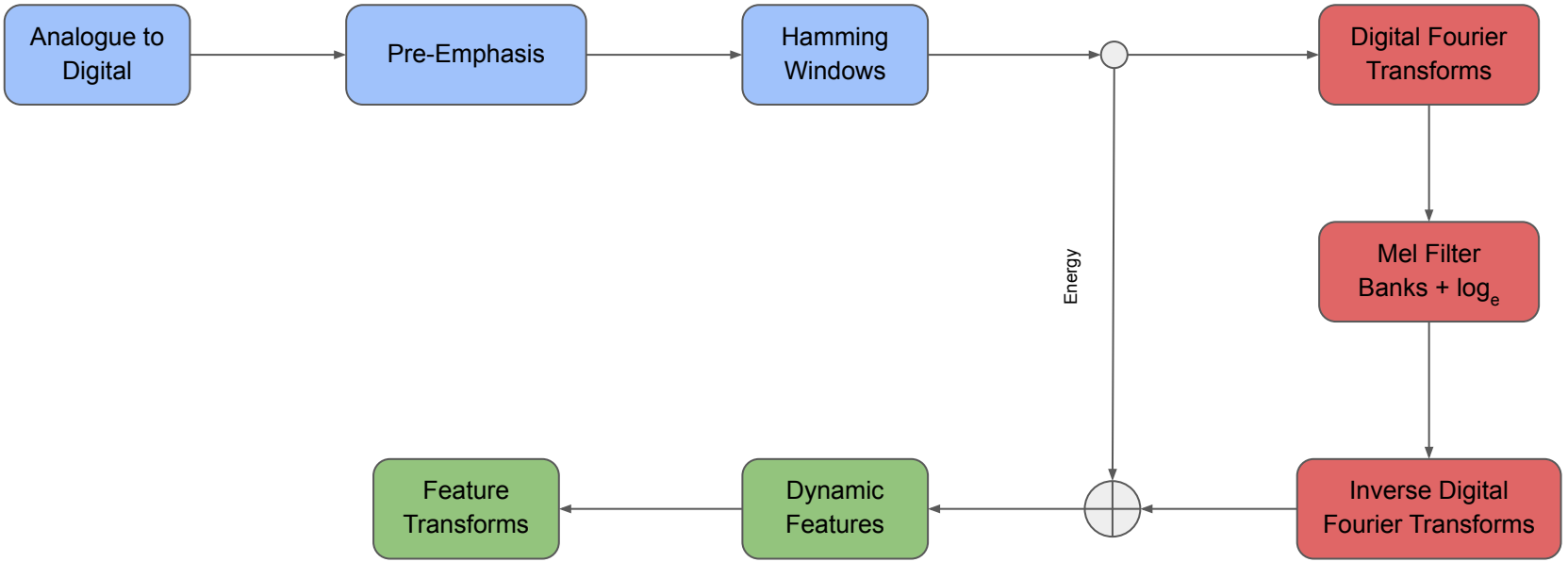
Cepstrum ↔ Spectrum

$C(x(t)) = F^{-1}[\log(F[x(t)])]$

Generating MFCCs



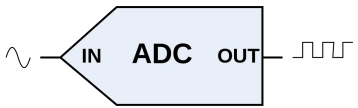
Generating MFCCs



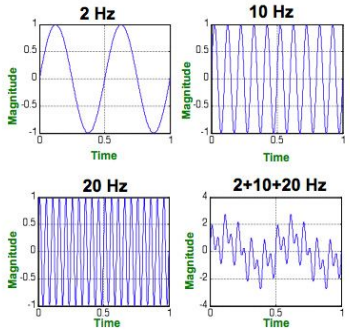
Stage 1

Analogue to Digital → Pre-Emphasis → Hamming Windows

Convert continuous analogue signals into discrete digital signals



Giving a boost to higher frequency signals



Segmenting signal into windows 25ms in width and 10ms apart

Cosine window instead of a rectangular one

Stage 2 - The Cepstrum

Digital Fourier Transform → Mel Filter banks + $\log_e$ → Inverse DFT

Bring signal from time-domain to frequency domain

Decompose signal into series of sinusoids

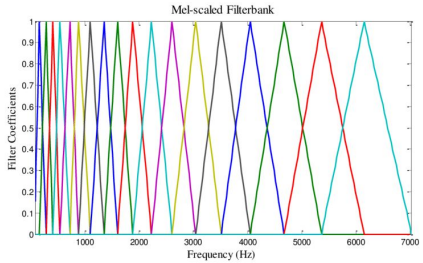
Mapping the signal onto separate frequency bands at the mel-frequency scale

Close to the non-linear human perception of sound

Put signal back into the time-domain - *Selectively*

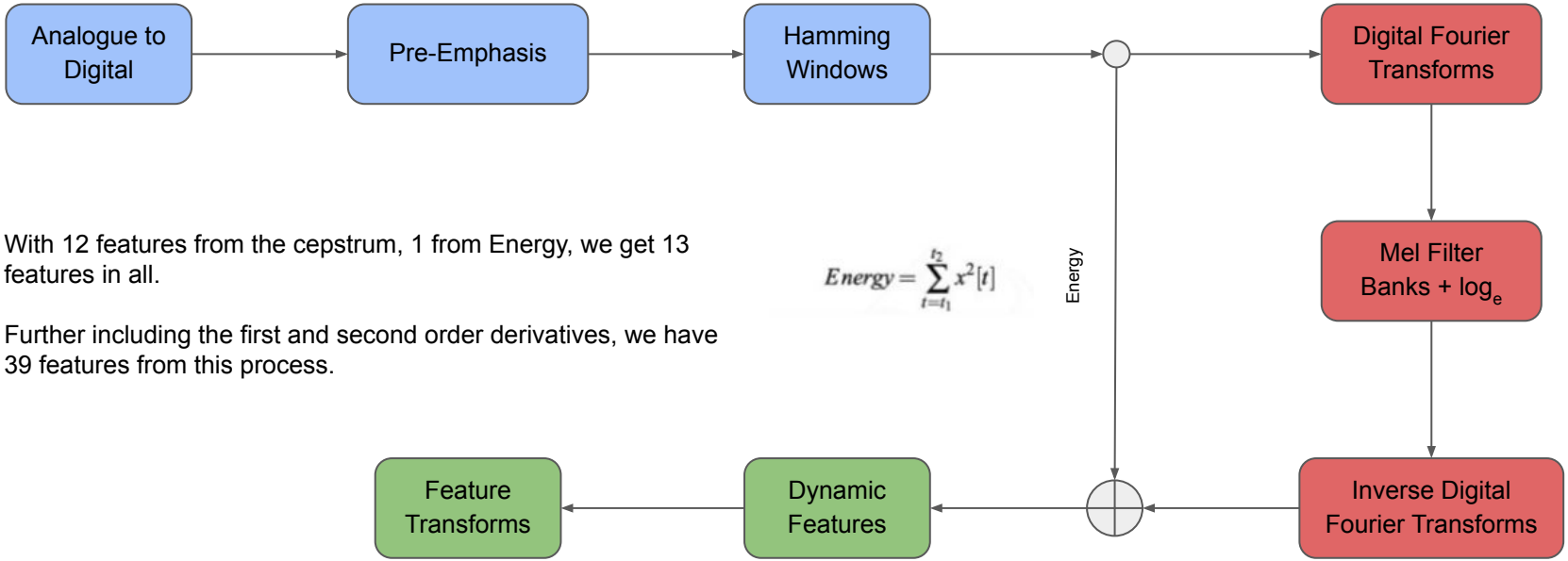
Cepstrum ↔ Spectrum

$C(x(t)) = F^{-1}[\log(F[x(t)])]$



$$mel(f) = 1127 \ln(1 + \frac{f}{700})$$

Stage 3 and Energy



With 12 features from the cepstrum, 1 from Energy, we get 13 features in all.

Further including the first and second order derivatives, we have 39 features from this process.