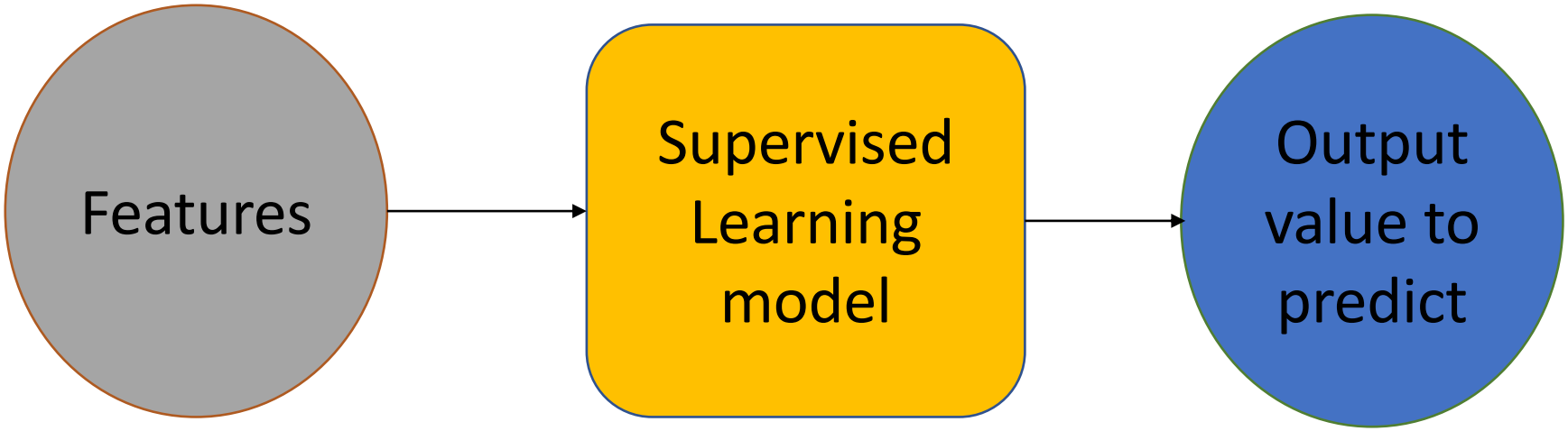
Machine Learning
CSCE 5215

Feature Engineering

Learning Objectives

- Feature engineering strategies
- Steps for feature engineering
- Summary
- Python implementation of Feature extraction
- Python implementation of Feature transformation

Feature Engineering



Feed into …

Algorithm learns …

Predict output …

Which features?

Choose features? How?
Create new features? How?

House pricing

Size	# of lizards	Price
3500 sft	3	$ 400,000
2200 sft	0	$ 150,000
1000 sft	12	$ 300,000

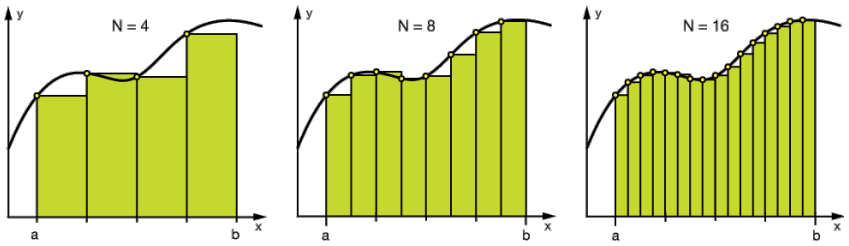
Feature engineering strategies

Normalization

$$x_{normalized} = (x - \mu_{training\ set}) / \sigma_{training\ set}$$

$$x_{normalized} = (x - min_{training\ set}) / (max_{training\ set} - min_{training\ set})$$

Binning



One-hot encoding

User	Color
1	Yellow
2	Green
1	Green
3	Orange
2	Orange
1	Orange
1	Yellow

User	Green	Orange
1	0	0
2	0	1
1	0	1
3	1	0
2	1	0
1	1	0
1	0	0

Feature engineering definition

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

- Creating appropriate features for learning through
  - massaging current features (remove noise, normalization…)
  - combining current features (dimensionality reduction…)
  - extracting more relevant features from source data (frequency spectrum...)

Feature engineering steps

1. Basic preprocessing
    a) feature extraction
    b) normalization
    c) introduce nonlinearities / transformations
    d) products of features

- Testing features
    - What is a “good” feature

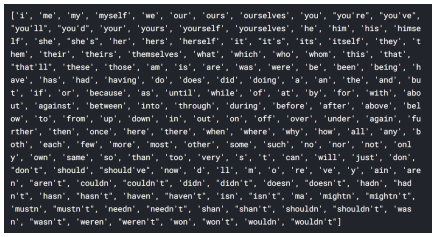
2. Domain-specific features
    a) Signal processing
    b) Audio
    c) Images

3. Hierarchical ML
    a) PCA/ICA
    b) Classifier output as input

1.Basic preprocessing

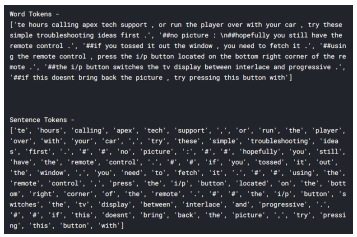
- E.g: text processing application

  -Eliminate stop words (extra words which don’t have useful meaning)




```
word length with stopwords 12593
word length without stopwords 7190
```

  -Tokenize (single entity of a whole entity)



  -Stemming (removes redundancy by bringing everything in its simple form)
  'dancing' & 'dancer' becomes 'dance' in this.

1a) Feature extraction

- Data often comes in raw and useless in a direct way
  - Images as array of pixel intensities
  - Sounds as air pressure variations
  - Movement of a sensor as variations in accelerations or torques
- Extract meaningful and potentially useful values
  - we will discuss signal processing techniques for those above
- e.g. from a time stamp: “2014-09-20T20:45:40Z”
  - Time of day?
  - Week, month, year?
  - M-F? Weekend?
  - Holiday?

1b) Normalization

Without normalization, some features would dominate the distance metric

Similar problem in other classifiers. Because of this, many methods may normalize automatically and fix the normalization parameters from the training set

Examples of typical normalizations:

- standardized = $(raw - \mu_{training\ set}) / \sigma_{training\ set}$
- standardized = $(raw - min_{training\ set}) / (max_{training\ set} - min_{training\ set})$

1c) Transformations

- New features can provide flexibility to simpler prediction algorithms
  - Linear algorithms can do the work on non-linear ones by a simple transformation of features
  - e.g. linear regression can fit quadratics with the addition of squared features
- Some algorithms assume particular data distributions to meet their assumptions and transformation can help meet that assumption
  - linear regression assumes normally distributed data
  - though it can still be robust to deviations from those assumptions
  - e.g. Reaction time data
    - log-normal → normal distribution by log transform
- If you care more about % differences than magnitude of error
  - Log transform your predictor!
- Also, can distribute feature values more sensibly
  - log (friends on twitter) to lessen impact of celebrities
  - log (revenue of companies) to lessen impact of Walmart, ExxonMobile, Sinopec, Apple...

1d) Products of features

- An easy way to automatically generate new features
- Particularly relevant if dependencies are expected between features
- e.g. holiday weekends are not well represented as linear combinations
  - e.g. holiday weekend bike ridership not equal to holiday + weekend bike ridership
  - holiday (1 or 0)
  - weekend (1 or 0)
  - holiday weekend = holiday * weekend
- Often done concurrently with nonlinear individual features
- $y = a_0 + a_1x_1 + a_2x_2$ becomes...
- $y = a_0 + a_1x_1 + a_2x_2 + a_{12}x_1x_2$ (+ $a_{11}x_1^2 + a_{22}x_2^2$)

2.Domain specific feature engineering

But sometimes you just have to know the domain and come up with some features based on your domain knowledge

- Trying to distinguish daytime indoor and outdoor visual scenes?
  - amount of blue is more useful than an individual pixel's intensity
- Trying to distinguish a male from a female voice?
  - a frequency decomposition would be useful
- Trying to distinguish walking from running
  - standard deviation of torque or accelerometer reading would work well

Each of those used our "common sense" or domain specific knowledge to extract useful features. Some require complex math. Others - domain specific experience.

Note, in many cases you have to segment out what you want to identify first using simple cues (words in speech, objects in images, …)

2a) Signal processing, basic - 1D

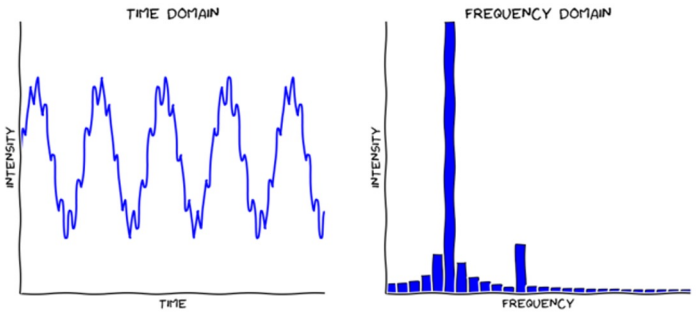
Let's cover a few simple signal processing tricks that may come in handy. You have a 1D signal... what can you extract from it. Let's brainstorm:

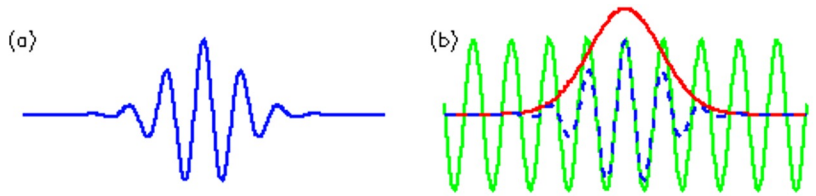
- Moments and their derivatives (e.g. absolute value, squares)
  - 1st order: mean, 2nd order: standard deviation, variance
  - 3rd&4th: skew and kurtosis
- Frequency components: Fourier analysis
- extremes: max/min
- Quantiles: median, 25%, 75%...
- Histograms (% in each range)
- smoothed (low-pass filtered) versions of above.

But remember, many signals are not just 1D (3 axis accelerometer, 2D images...)

2b) Audio

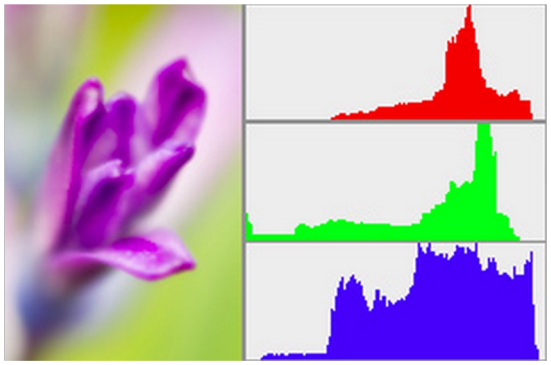
- A great example of a 1D signal (though sound localization and source separation is possible with multiple microphones for better processing)
- Frequency spectrum is critical
  - But frequency spectra have no temporal precision! solution...
- Wavelets: localized “frequency” detectors
- Speech: many more appropriate features possible, but must be localized (perhaps through segmentation of words)

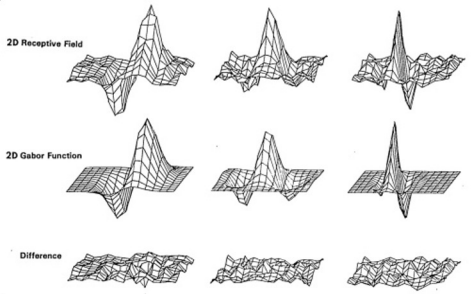




2c) Images - (2D signal, often with color)

- Object recognition: much harder than you would think (why?...)
  - e.g. Dog vs. Cat is a challenging ML problem
  - Face recognition is even harder
- More critical to segment appropriately first (if you can)
- Example features
  - Color and intensity histograms
  - Orientation / edges
  - Spatial frequency
  - match to simple templates (T-junctions, L-junctions…)
  - Conjunctions of the above
- Note on how the brain does this…
  - similar to 1D wavelets, but that's just the first step





3.Hierarchical ML

Feature engineering sounds cyclical…

“find/create/combine the best features to help your learning algorithm (to pick/combine the best features)”

...isn’t that what ML is supposed to do for you?

Yes. You can use ML to do the job of feature selection. In some cases this is necessary.

- Two common cases:
  - Too many features for too few samples → dimensionality reduction
  - Weak features or to maximize accuracy → Use the output of one classifier to aid in prediction

3a) Dimensionality Reduction / Factor Analysis

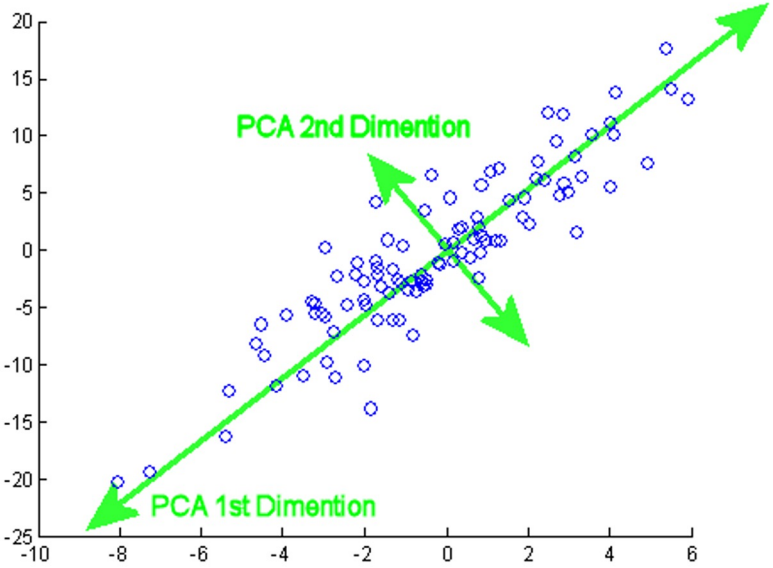
Many regression / classification techniques perform poorly with large feature sets

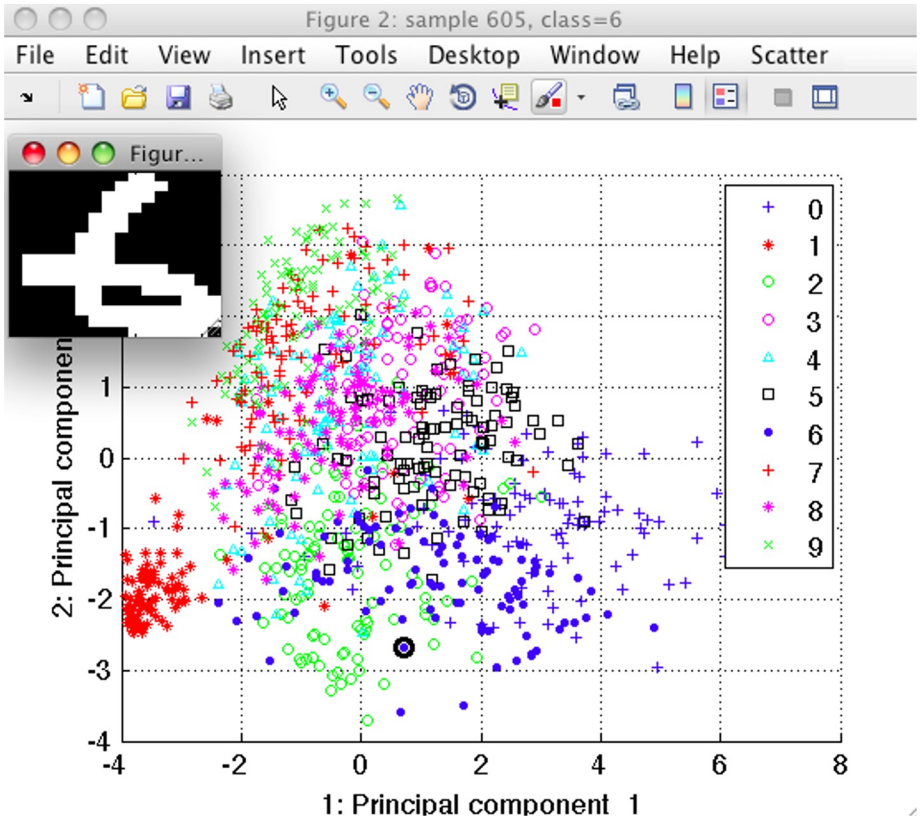
Simple solution: use unsupervised ML to reduce the number of features, without regard to what you are predicting.

e.g. used in psychology to take 100 personality questions and boil it down to, say, 5 personality characteristics

This allows us to view (by approximation) a larger dimensional space than the 2-3 that we are used to graphically...

3b) Principal Components Analysis (PCA) example





Hierarchical ML using prediction models as input

Useful when approximated classes can help, but no operational definition will work

Examples:

- Easier to predict office vs non-office scenes than individual objects. Use that classifier as a feature in detecting monitors, mice, staplers…
- Create a prediction model when certain, useful features are available for only some of the samples (note: missing values bias concerns here)

Feature Engineering Summary

- Basic preprocessing
  - feature extraction
  - normalization
  - introduce nonlinearities
  - products of features
- Testing features
  - What is a “good” feature
- Domain-specific features
  - Signal processing
  - Audio
  - Images
- Hierarchical ML
  - PCA/ICA
  - Classifier output as input