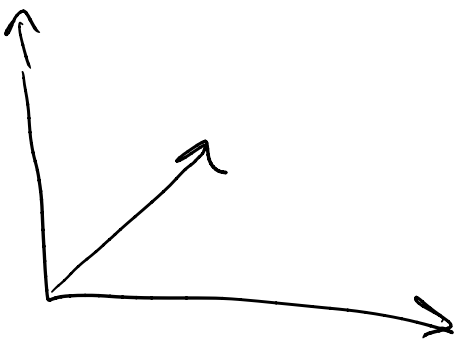
Dimensionality Reduction

Curse of Dimensionality

- what happens with data with 1000s
- each feature "dimension"



- data gets significantly more sparse in higher dimensions

* need exponentially more data as dimensionality increases

sparse data => data doesn't really look similar

Dimensionality Reduction

- <u>feature selection</u> → identify features to keep, which features to discard (the rest)
  keep k features

- feature extraction → find k new features (dimensions that are combinations of the original features)
  ↳ aka "feature engineering"

Feature Selection - Why?

- reduce overfitting
  - redundant data makes it easier for noise to be modeled
- reduces training time
  - ↳ probably should take into account time for feature selection
- in addition to curse of dimensionality

Feature Selection

Subset Selection

For both, need to check error on validation set (adding more features ⇒ training set error goes down)

A) Forward selection

start with no variables
add in one at a time (choosing one that decreases error the most)
stop after certain # of features
stop after adding one makes minimal difference

B) Backward selection

start with all
removes one by one (choosing one to remove that decreases error the most or increases slightly)
stop when removing causes a significant increase the error

Issues

Which is more costly? backward selection

- costly
  - 1st time train d-models
    Then train d-1, then d-2, ..., d-k
    (as opposed to just training one model)
- greedy algorithms → no guarantee of best subset
  * some features together can make a big difference
- not always appropriate
  Example: image face recognition
  individual pixels don't carry discriminative info
  ↳ combination of pixels that matter

Variance Threshold:

- a feature that doesn't vary much has little predictive power
- remove it

Regularization:

- option for many of models L_1
- add in penalty term
  - ↳ penalize non-zero weights
  (Ex: penalty term in regression)

Statistical Test Scores

Ex: look at χ^2 scores to identify k best features

Feature Importance

* weights in regression
* random forests: mean decrease in impurity or gini importance measure (how effective was feature at reducing uncertainty)
  - ↳ biased: inflates importance of continuous and high-cardinality categorical variables

* Permutation Importance
  1) train model
  2) predict on data
  3) randomly shuffle a single feature
  4) predict again
  5) compare difference in predictions
  6) If difference is low, feature isn't really important (random value from distribution did just as well at predicting)

beds	baths	price
5	3	300k
3	3	400k
4	3	200k
2	2	120k
3	1	200k

Feature Extraction

Idea: find smaller number of new features

Two sides:

1) mathematically find low dimensional representation that closely approximates true data

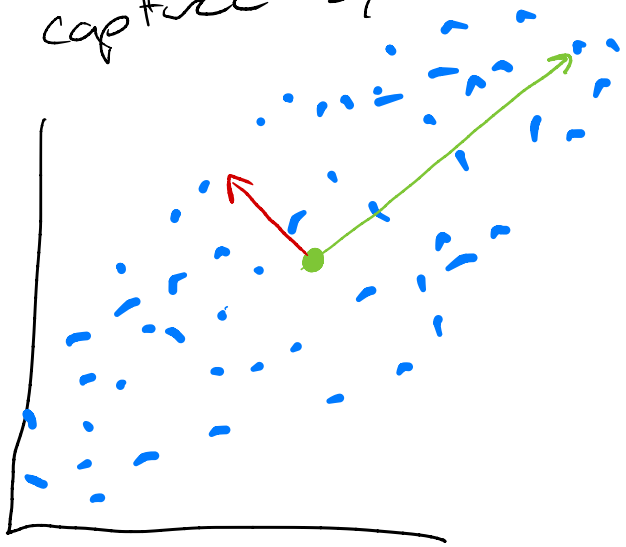
2) use problem knowledge to identify computed features that may be more useful

   Ex: compute curvature for trajectories to anomalous trajectory analysis

Mathematical Approaches

PCA: represent data as PCs
orthogonal directions that capture most of the variance

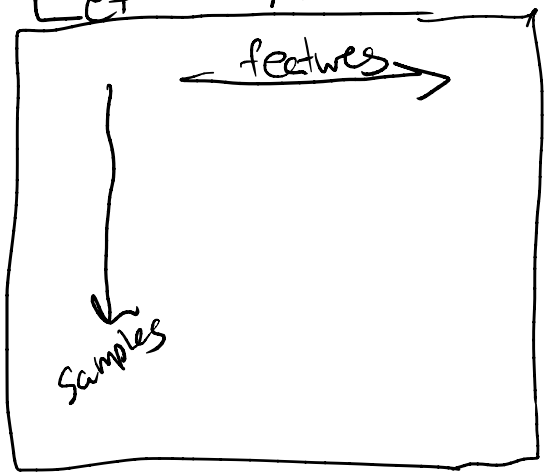
- first PC captures most variance
- second PC captures most variance in directions not captured by first PC



Process:

a) compute covariance matrix

Let X be matrix of data, centered (mean subtracted)



$$\Sigma = \frac{1}{N} X^T X$$ covariance matrix

b) Compute U and D s.t.

$$\Sigma = UDU^T$$

↳ U = matrix of eigenvectors of Σ

↳ D = diagonal matrix of eigenvalues of Σ

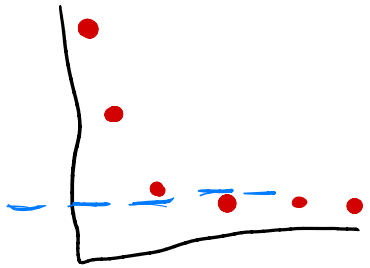
↳ Σ is symmetric so we know we can do this (and get orthogonal eigenvectors)

Typically, a few eigenvalues are large, many are quite small

- eigenvectors for large values are directions of greatest variance
- represent in these dimensions
  ↳ need fewer dimensions → lower dimensional representation of the data

c) Projection onto lower dimensional subspace

- truncating columns of U to get rid of columns corresponding to the small eigenvalues
  - ↳ both for elbow in plot of eigenvalues



$$\hat{X} = U U^T X^T$$

- $U^T X^T$: scalars = coefficients of basis vectors
- U: columns of U are basis vectors

$\hat{X}$: will have columns as samples → need to transpose to get back to rows as samples

↳ need to add back in mean

PCA (cont)

- Projected Representation Error

$$\frac{1}{N}\sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2 = \sum_{j \in \text{truncated cols}} \lambda_j$$

- Practical Comments:
  - never actually want to compute covariance matrix and don't want to compute its eigenvalues
    - ↳ computationally unstable (bad things happen numerically)
    - ↳ computationally expensive
  - instead compute SVD of data matrix
    - ↳ let a package compute it for you

ICA: Independent Component Analysis

- cocktail party problem
  - signal is linear combination of seperate independent sources
- independent → does not mean orthogonal
- cares about higher order statistics (than PCA)
- fundamentally not actually dimensionality reduction
  - doesn't seperate which sources are most important
  - can be if signal is filtered after