March 14, 2018

Introduction to Data Science

Zeev Waks, Intel

Feature selection

Advanced Analytics

We mine your business

Agenda – Curse of dimensionality

- ~~Quick ML review~~
- ~~Model evaluation~~
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross validation~~
  - ~~Training with imbalanced classes~~
  - ~~Overfitting/underfitting~~
- Dimensionality and feature selection
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Curse of Dimensionality (Bellman 1961)

- What is it?
  - A name for **various problems that arise** when analyzing data in high dimensional space. Dimensions = independent features in ML
  - Occurs when # dimensions is large in relation to number of samples.

- Real life examples:
  - We have about 10^6 possible genomic features, but our human sample sizes are often in the 100s or 1000s of different genomes.

So what is this curse?

- **Sparse data:**
  - When the dimensionality d increases, the volume of the space increases so fast that the available data becomes **sparse, i.e. a few points in a large space**
  - Many features are not balanced, or are 'rarely occur' – sparse features

- **Noisy data**: More features can lead to increased noise □ it is harder to find the true signal

- **Less clusters**: Neighborhoods with fixed k points are less concentrated as d increases.

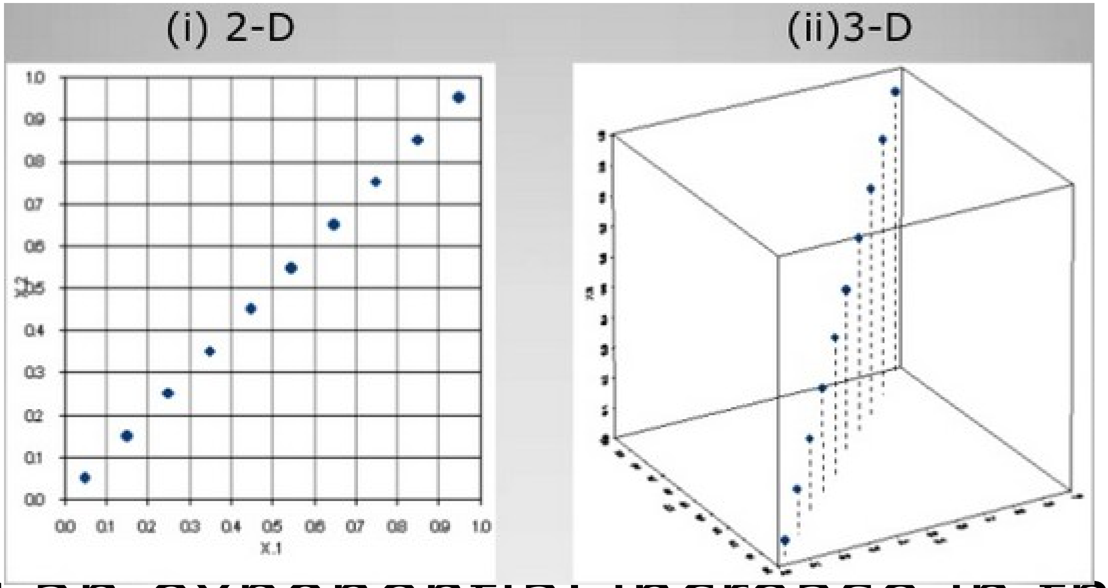
- **Complex features**: High dimensional functions tend to have more complex features than low-dimensional functions, and hence harder to estimate

Curse of dimensionality – Runtime complexity

- Complexity (running time) increase with dimension **d**
- A lot of methods have at least $O(n*d^2)$ complexity, where n is the number of samples
- As d becomes large, this complexity becomes very costly ($).

Data becomes sparse as dimensions increase

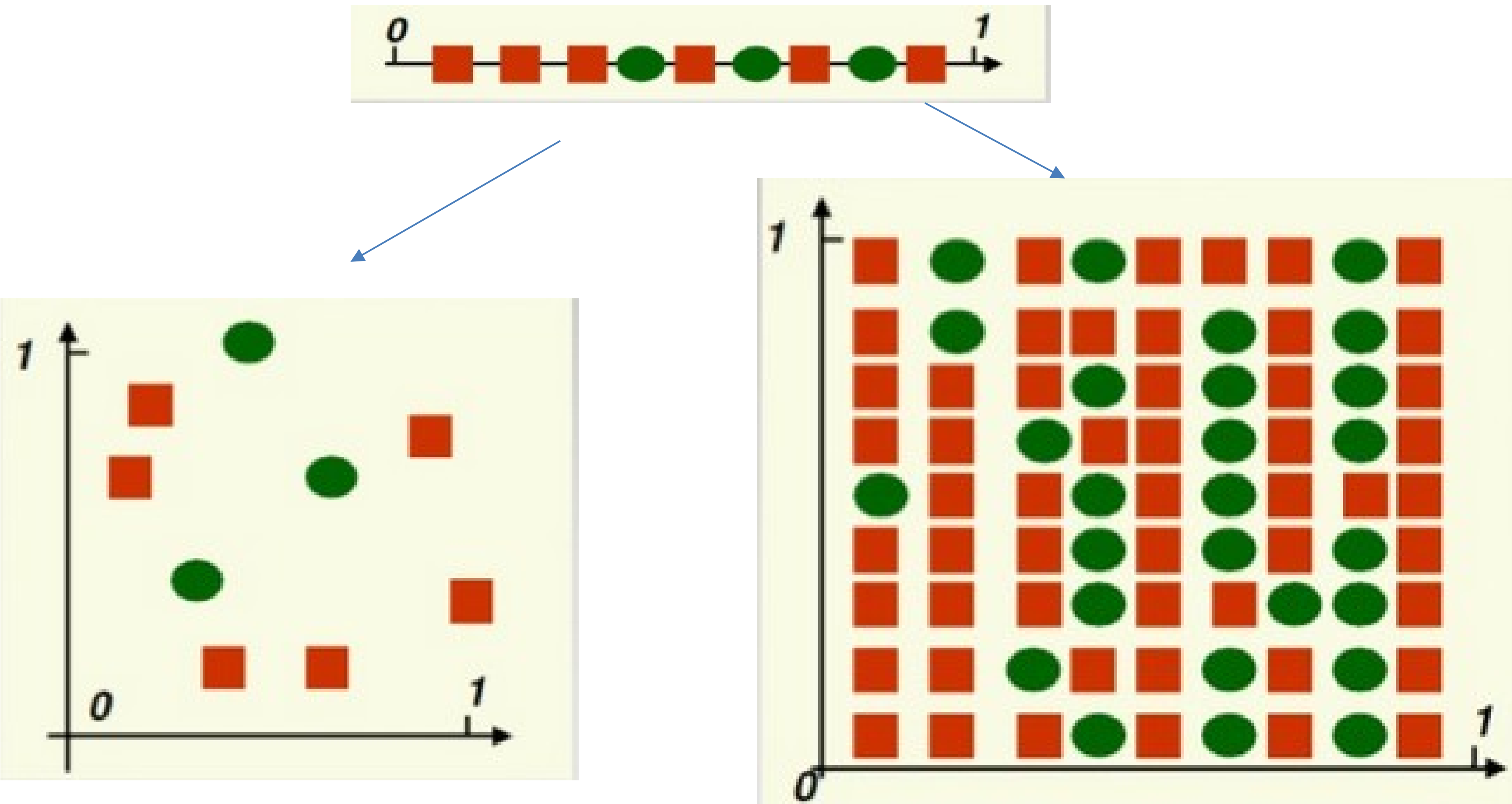
- A sample that maps 10% of the 1x1 squares in 2D represent only 1% of the 1x1x1 cubes in 3D



- There is an exponential increase in the search-space

Visual example 2

- 9 samples in 2D look sparse. Need 81 to keep same density



Some mathematical (weird) effects

- Ratio between the volume of a sphere and a cube for **d**=3:

$$\frac{(\frac{4}{3})\pi r^3}{(2r)^3} \approx \frac{4r^3}{8r^3} \approx 0.5$$

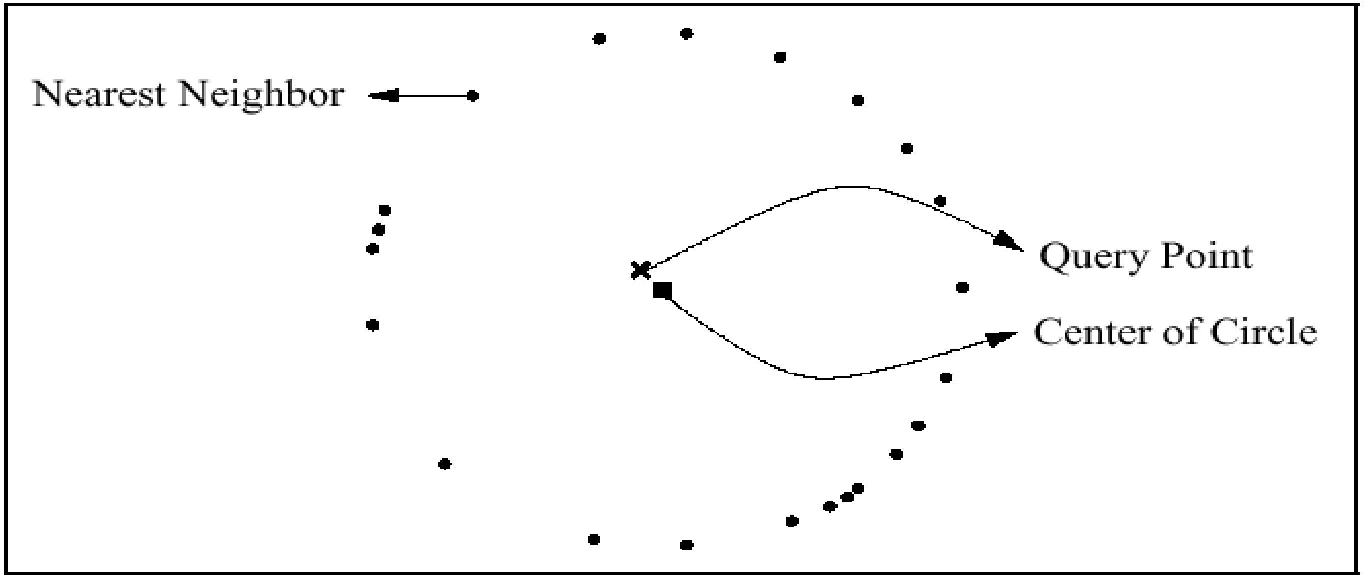
- When **d** tends to infinity the volume tends to zero

- Most of the data is in the corner of the cube
  - Thus, Euclidian distance becomes meaningless, most two points are “far” from each others

➢ Very problematic for methods such as k-NN classification or k-means clustering because most of the neighbors are equidistant

The K-NN problem: visualization

- If all the points are pushed on the outer shell of the sphere then all potential NN (nearest neighbors) appear equidistant from the query point



Just a second…What is a dimension anyway?

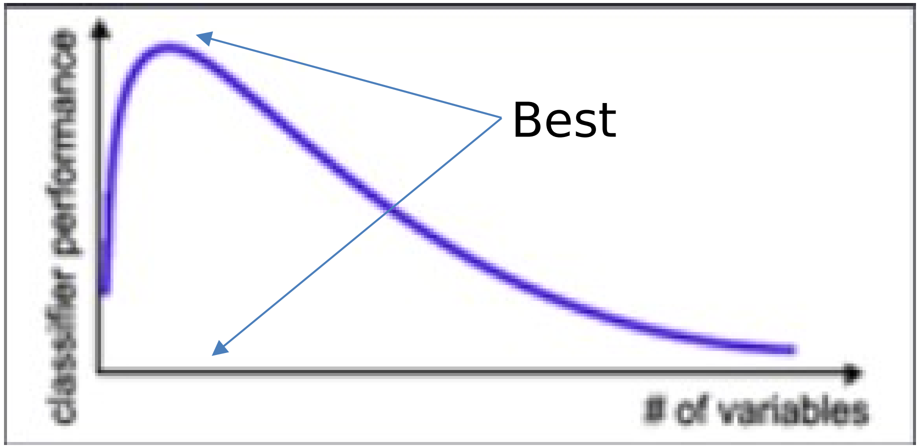
x1	x2	x3	x4
1	2	1	1
2	4	0.5	1
3	6	17	1

- How many dimensions does the data intrinsically have here?
  - Two!
    - x1 = ½ * x2 (no additional information)
    - x4 is constant

How to avoid the curse?

- Reduce dimensions
  - Feature selection - Choose only a subset of features
  - Use algorithms that transform the data into a lower dimensional space (example – PCA)

*Both methods often result in information loss



Agenda – Filter methods

- ~~Quick ML review~~

- ~~Model evaluation~~
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross validation~~
  - ~~Training with imbalanced classes~~
  - ~~Overfitting/underfitting~~

- Dimensionality and feature se
  - ~~Curse of dimensionality~~
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Feature selection goals

- **Benefits** of feature selection
  - Reduce overfit risk (by reducing model complexity)
  - Reduce dimensionality
  - Improve compute speed

- Feature selection is **not always necessary**

Feature selection challenges

- It is a search/optimization problem
  - Exhaustive testing of feature combinations is often unrealistic

- Goal of feature selection methods is to find a ‘good enough’ feature set
  - Requires score for ranking
  - A heuristic to prune the space of possible feature subsets, and will guide the search

Feature selection types

- **Filter method:** Ranks features or feature subsets independently of the classifier
  - Low computational power
  - Independent of model type

- **Wrapper method:** Uses a predictive model (machine learning) to score feature subsets.
  - Often better than filter method
  - Requires training a model for each feature set

- **Embedded method:** Performs variable selection (implicitly) in the course of model training (e.g. decision tree, WINNOW)

Filter methods

- Select subsets of variables as a pre-processing step, by ranking according to some scoring metric, **independently of the learning model**

- Relatively fast & not tuned by a given learner
- Very commonly used

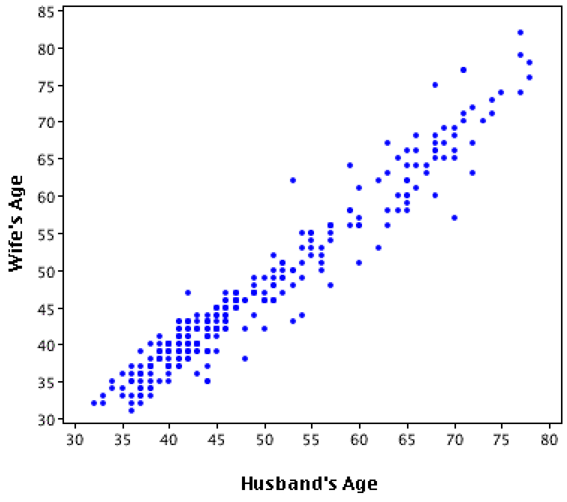
Feature ranking – common examples

- Label association:
  - Example: Choose the top 10 features correlated with the label
- Low variation features:
  - Remove features with little variation in their value
- Correlated features (redundancy):
  - Keep only one of two highly correlated features

Common scoring functions

Pearson Correlation

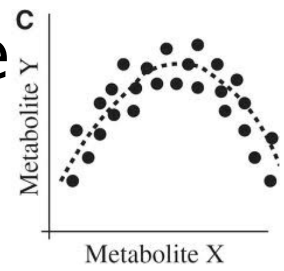
- Can only detect linear dependencies

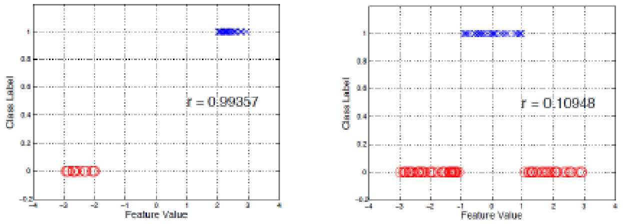


Mutual information (MI)

"How much information two variables share"

- Can detect any form of statistical depe





Multivariate filter methods - Types of search strategies

Background

- Choose search strategy – Feature subset combinatorial space is often very large
- Choose ranking/evaluation method for feature subsets
  - Provide a feedback to the search strategy for the relevancy of the candidate subsets

Approaches

- Sequential algorithms (forward selection, backward selection)
  - Add or remove features sequentially, but have a tendency to become trapped in local minima

- Randomized algorithms (Genetic algorithms, simulated annealing)
  - Incorporating randomness into their search procedure to escape local minima

Agenda – Wrapper methods

- ~~Quick ML review~~

- ~~Model evaluation~~
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross validation~~
  - ~~Training with imbalanced classes~~
  - ~~Overfitting/underfitting~~

- Dimensionality and feature se
  - ~~Curse of dimensionality~~
  - ~~Filter feature selection~~
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Wrapper methods

- Definition: Use of a learning model to choose features. The model is NOT 'the model' that is eventually used.

- Features and feature subsets are ranked based on their contribution to model performance

- Benefit: Often good selection of feature subsets

- Drawbacks
  - Computational requirement: Requires training a model on each feature subset
  - Variation: Result vary for different learning models

Wrapper methods in practice

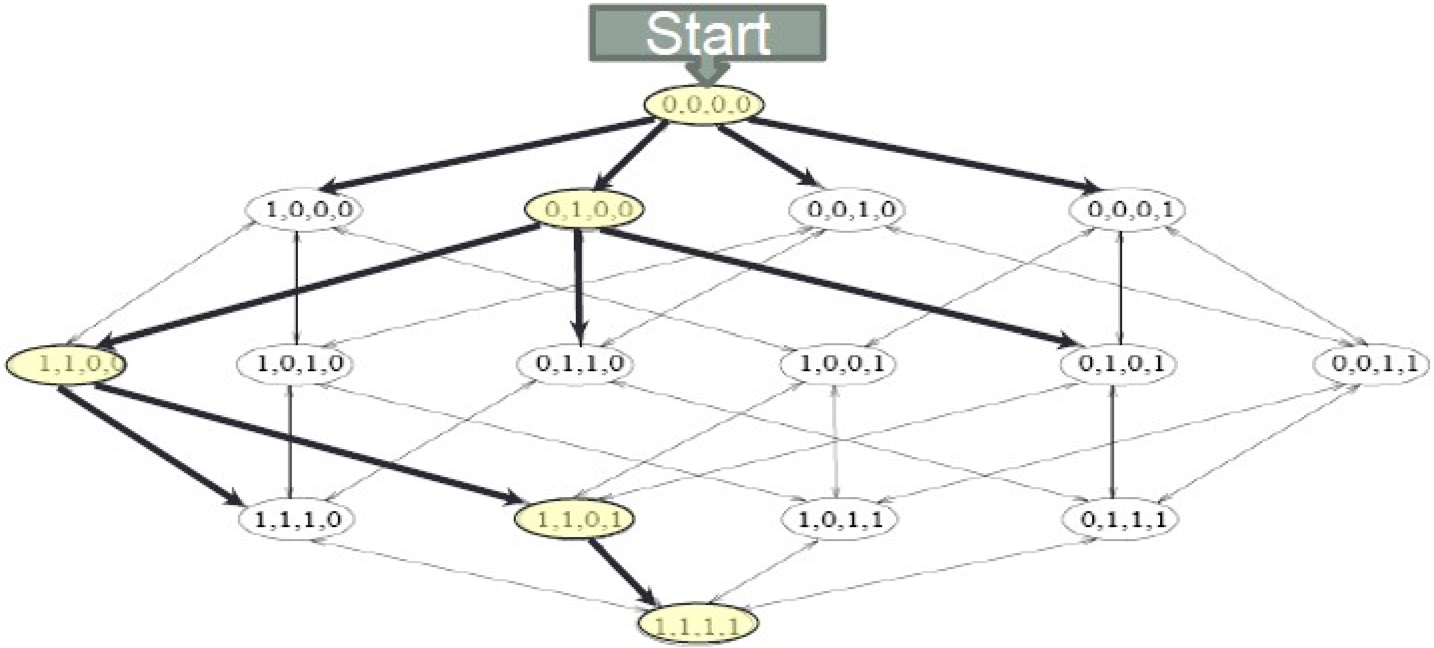
- Often preceded by filter methods to reduce computational cost

- Various heuristic search strategies are used. Most common are:
  - Forward selection – Start with an empty feature set and add features at each step
  - Backward selection – Start with a full feature set and discard features at each step

*Both are greedy since exhaustive search is often not feasible

- Evaluation is usually done on validation (development) set

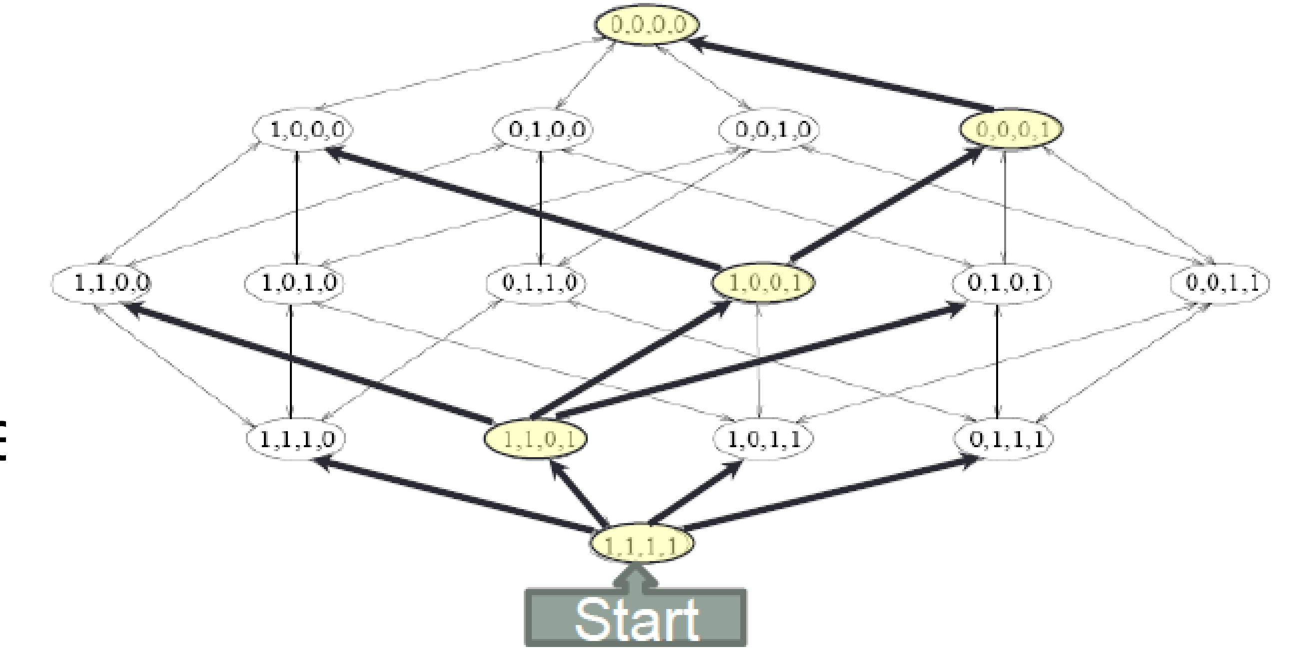
Sequential Forward Selection (SFS)



SFS ... s a

Sequential Backward Selection (SBS)

SBS pe



set has

Forward vs. Backward

- **Forward selection** considered computationally more efficient and has an advantage detecting the strongest single feature
- **Backward selection** can detect “stronger” subsets because the importance of features is assessed in the context of other features
- **Hybrid techniques** attempt to enjoy both approaches

Feature selection summary

- Feature selection is usually good practice
- Filter methods are:
  - Fast, model independent, tend to select large subsets
  - Frequently used examples:
    - Correlation with label, correlation between features, features with little variation
- Wrapper methods are:
  - Accurate, avoid overfitting, slow, model dependent

Agenda – PCA

- ~~Quick ML review~~

- ~~Model evaluation~~
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross validation~~
  - ~~Training with imbalanced classes~~
  - ~~Overfitting/underfitting~~

- Dimensionality and feature se
  - ~~Curse of dimensionality~~
  - ~~Filter feature selection~~
  - ~~Wrapper feature selection~~
  - Principal Component Analysis (dimensionality reduction)

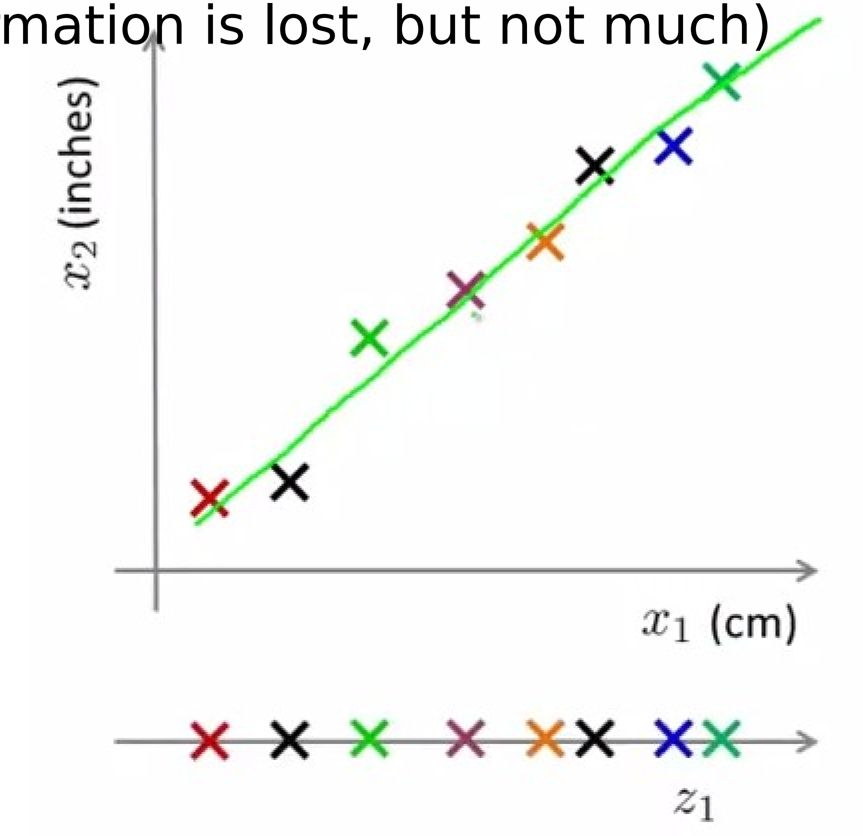
Dimensionality reduction goals

- Improve ML performance
- Compress data
- Visualize data (you can’t visualize >3 dimensions)
- Generate new features

Example – reducing data from 2d to 1d

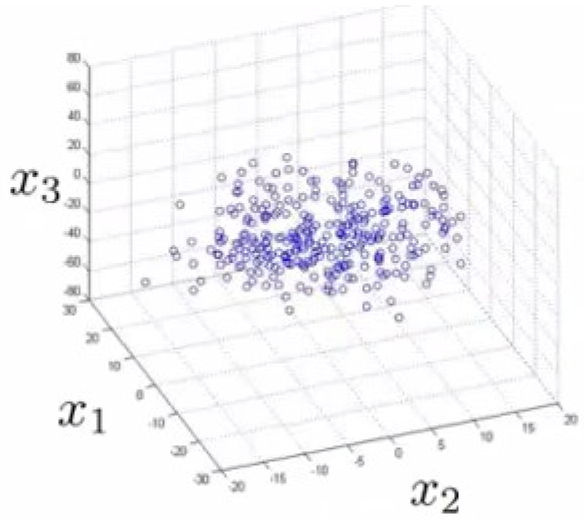
- X1 and x2 are pretty redundant. We can reduce them to 1d along the green line

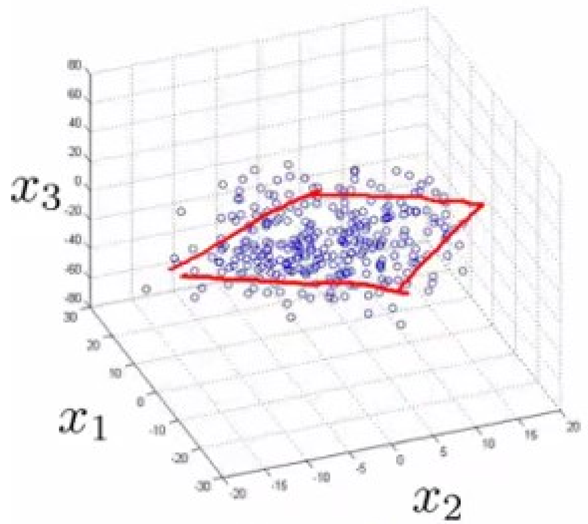
- This is done by projecting the points to the line (some information is lost, but not much)



Example – 3D to 2D (1)

- Let’s have a look at a 3D dataset

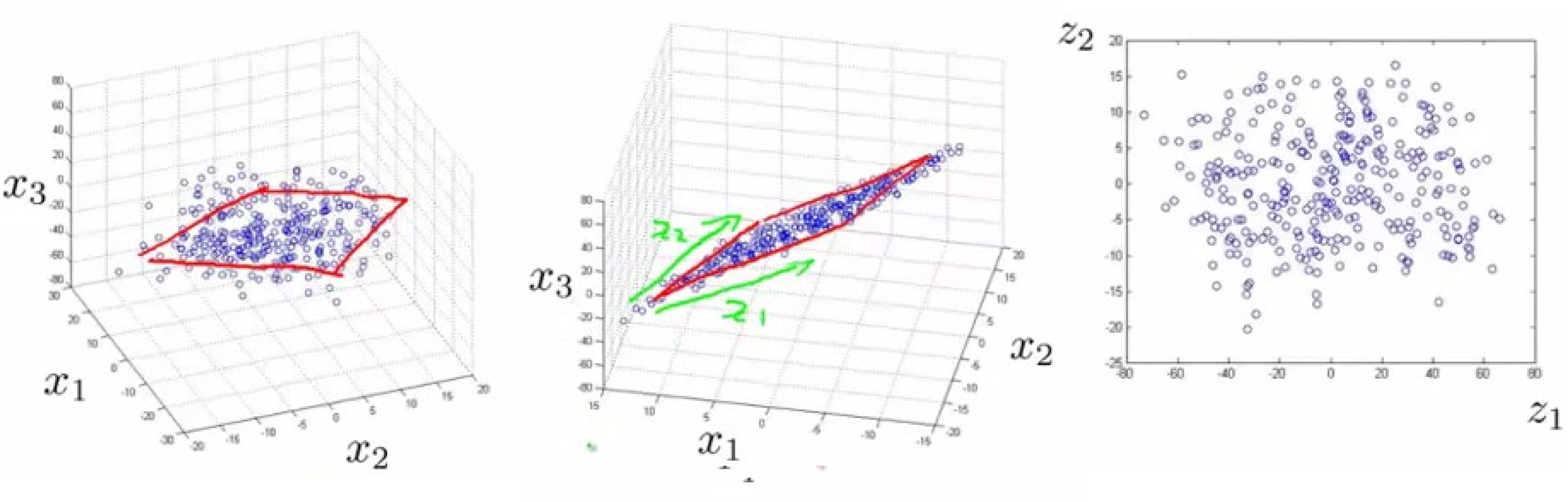




- Despite having 3D data most of it lies close to a plane

Example – 3D to 2D (2)

- If we were to project the data onto a plane we would have a more compact representation



- So how do we find that plane without loosing too much of the **variance** in our data? PCA is a linear method for doing this

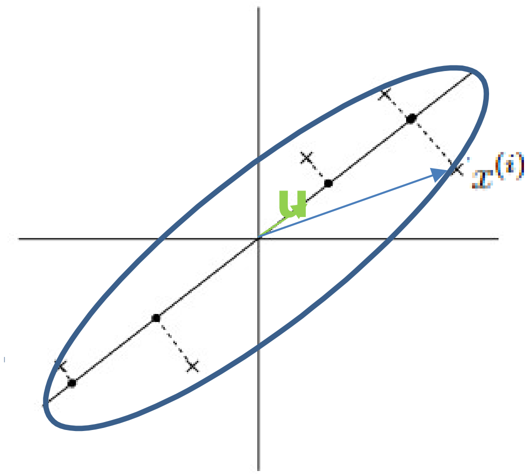
Principal Component Analysis (PCA)

- The idea is to project the data onto a subspace which compresses most of the variance in as little dimensions as possible.
- Each new dimension is a **principle component**
- The principle components **are ordered** according to how much **variance in the data** they capture
  - Example:
    - PC1 – 55% of variance
    - PC2 – 22% of variance
    - PC3 – 10% of variance
    - PC4 – 7% of variance
    - PC5 – 2% of variance
    - PC6 – 1% of variance
    - PC7 - ….

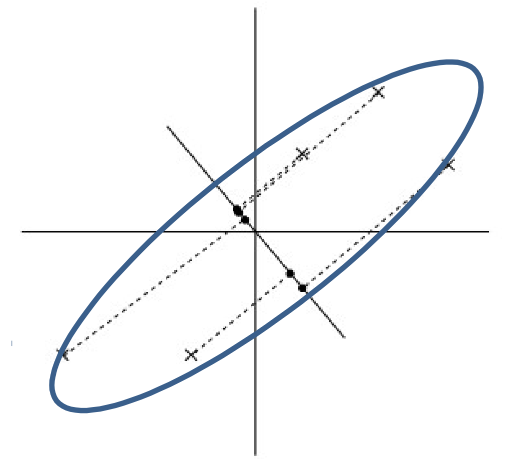
Geometrical intuition

- We want to find **new axis** in which the variance is maximal.

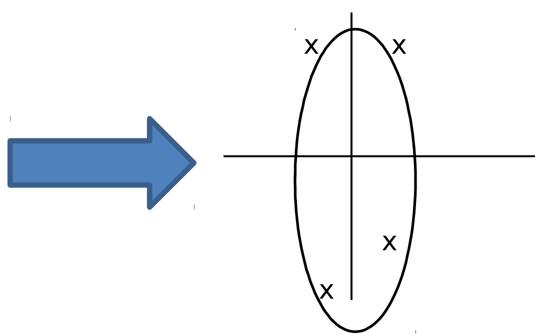
Large variance axis

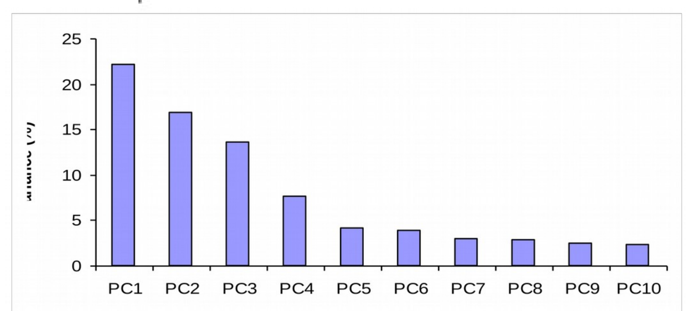


Small variance axis



- PCA is geometrically equivalent to the rotation of the axis.





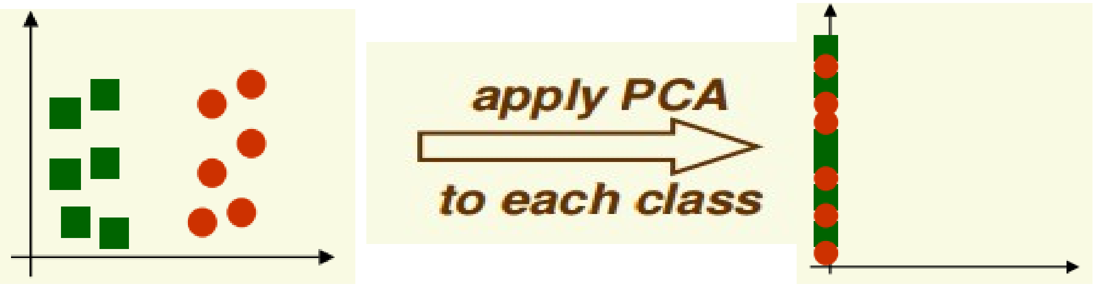
PCA algorithm

1. Mean normalization: For every value in the data, subtract its mean dimension value. This makes the average of each dimension zero.
2. Covariance matrix: Calculate the covariance matrix
3. Eigenvectors and eigenvalues: Calculate them
   – Note: Each new axis (PC) is an eigenvector of the data. The standard deviation of the data variance on the new axis is the eigenvalue for that eigenvector.
4. Rank eigenvectors by eigenvalues
5. Keep top k eigenvectors and stack them to form a feature vector
6. Transform data to PCs:
7. New data = featurevectors(transposed) * original data

$$\begin{pmatrix} y1 \\ \vdots \\ yK \end{pmatrix} = \begin{pmatrix} u11 & \cdots & uK1 \\ \vdots & \ddots & \vdots \\ u1n & \cdots & uKn \end{pmatrix}^T \begin{pmatrix} x1 \\ \vdots \\ xn \end{pmatrix}$$

When not to use PCA?

- PCA is completely unsupervised. It is designed for **better data representation** **not** for **data classification**
- Projecting the data on the axis of maximum variance can be disastrous for classification problems



- In case of Labeled multiclass data, it is better to perform Linear Discriminant Analysis (LDA)

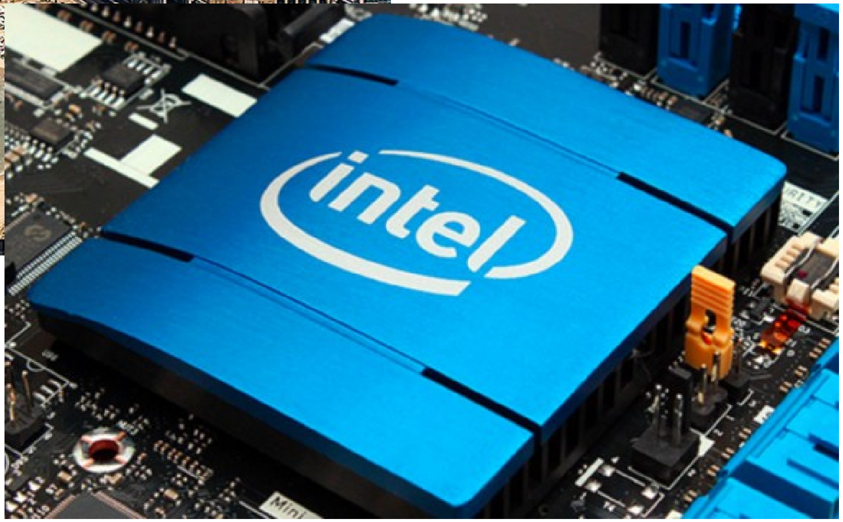
Agenda – Done

- ~~Quick ML review~~

- ~~Model evaluation~~
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross validation~~
  - ~~Training with imbalanced classes~~
  - ~~Overfitting/underfitting~~

- ~~Dimensionality and feature se~~
  - ~~Curse of dimensionality~~
  - ~~Filter feature selection~~
  - ~~Wrapper feature selection~~
  - ~~Principal Component Analysis (dimensionality reduction)~~

Thank you and enjoy life



Zeev Waks, zeev.waks@intel.com
Data Science Lead
Intel – Advanced Analytics