December, 2016

Introduction to Data Science: Model evaluation and features selection

Zeev Waks, Intel

Advanced Analytics

We mine your business

Outline

- 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)

Agenda - Quick ML review

- 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)

How do we acquire knowledge?

- Explicitly
  - When you learn a second language you receive a grammar book with lots of rules and have vocabulary lessons.

- By learning
  - Who read the instruction manual for Candy Crush?

Types of learning

- Supervised – Making predictions from **labeled** examples

  Classification: Finite labels (i.e.- gender)

  Regression: Continuous labels (i.e. height)

- Unsupervised – Detecting patterns from **unlabeled** examples

- Semi-supervised – Learning from labeled and unlabeled

These are typically the 'highest' level of description of what you do when you talk to someone about machine learning

Features and labels

- **Features** (a.k.a covariates) are the variables that describe the data.
- **Label** (target variable/covariate): The feature we want to predict in supervised learning. So, a label is kind of like a feature

- **Examples:**
  - Supervised - Predict housing prices using:
    - Features: number of bathrooms, floor, number of windows, kitchen size
    - Label: House prices
  - Unsupervised - Find patterns in books using:
    - # of pages, # of printed copies, language, # of pictures, average words per page
    - Label: there is none
      - Maybe we will 'cluster' the books into genres? Maybe into authors? Etc.

More examples of supervised vs unsupervised

Supervised:

1. Regression: Predict housing prices given the prices of 1,000 previously sold houses, with various **features** describing the houses (size, # of windows, …)

2. Classification: Predict if a gift is a book, a toy, or something else (three discrete options only)

Unsupervised

3. Get data on sports players and see if there are specific patterns.
   - Maybe the data has 3 'clusters' that mean something– basketball players, soccer players, and sumo wrestlers?
   - Maybe it clusters into categories nobody has thought about?

Memorization and generalization

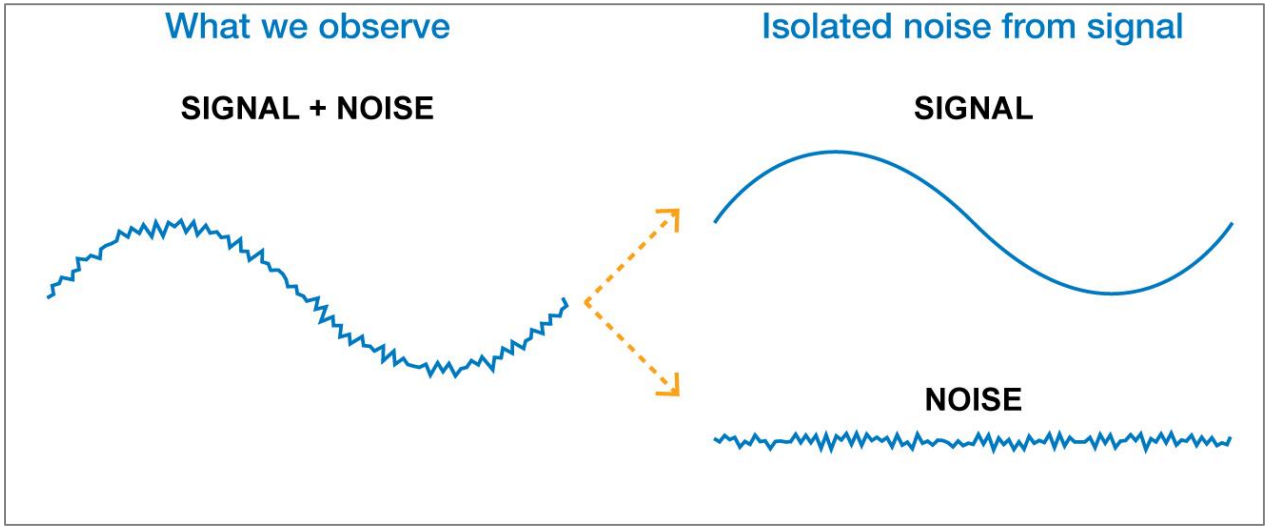
- Memorization:
  - Are we specifically only learning an algorithm on our data?
- Generalization
  - Refers to the capability of applying learned knowledge to previously unseen data
  - Without generalization there is no learning, just memorizing!

What's in the data we learn from

The data usually has two components:

- Signal: information that is relevant to pattern detection or prediction
- Noise: information that is irrelevant to the future

Finding which is which is the challenge



Agenda – Performance metrics

- ~~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)

Why do we evaluate a model's performance?

1. To know how well it works
   - Reporting results
   - Making scientific/business decisions
   - Can we use it in real-life?

2. To compare models
   - For choosing which models are best
   - For tuning a given model's parameters

Performance metrics

- Precision & Recall (and f1 score)
  - Precision: What percent of our predictions are accurate?
  - Recall: How many of the accurate predictions did we capture
  - F1 score: A single number that combines the two values above. Good for ranking/sorting…

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

  - Specificity/sensitivity – very similar to precision/recall. Often used in medicine.

- Precision at N
  - How many accurate examples did we capture in our top N ranked examples. This is often used in information (document) retrieval

- Area under curve (the ROC curve)
  - A Receiver Characteristic Curve (ROC) plots the True positive rate (TPR) vs. the False positive rate (FPR). The maximum area under the curve (AUC) is 1. Completely random predictions have an AUC of 0.5. The advantage of this metric is that it is continuous.

The confusion matrix

- Purpose: Evaluating binary classification
- Many performance terms: Precision, recall, sensitivity, specific, true negative rate, true positive rate, PPV, NPV, type 1 error, type 2 error

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	Positive predictive value = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Test Outcome Positive}}$
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Test Outcome Negative}}$
		Sensitivity = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Condition Positive}}$	**Specificity** = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Condition Negative}}$	

*Even this picture is not complete (precision/recall on next slide)

Confusion matrix – another look

Predictive Model: Evaluation

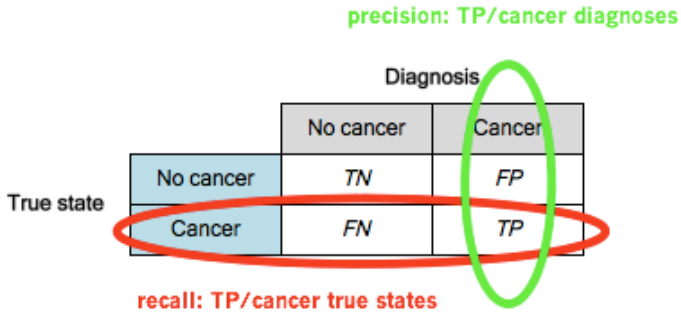
$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

		actual result / classification: yes	actual result / classification: no
predictive result / classification	yes	**tp** (true positive)	**fp** (false positive) ← Type 1 error
	no	**fn** (false negative)	**tn** (true negative)

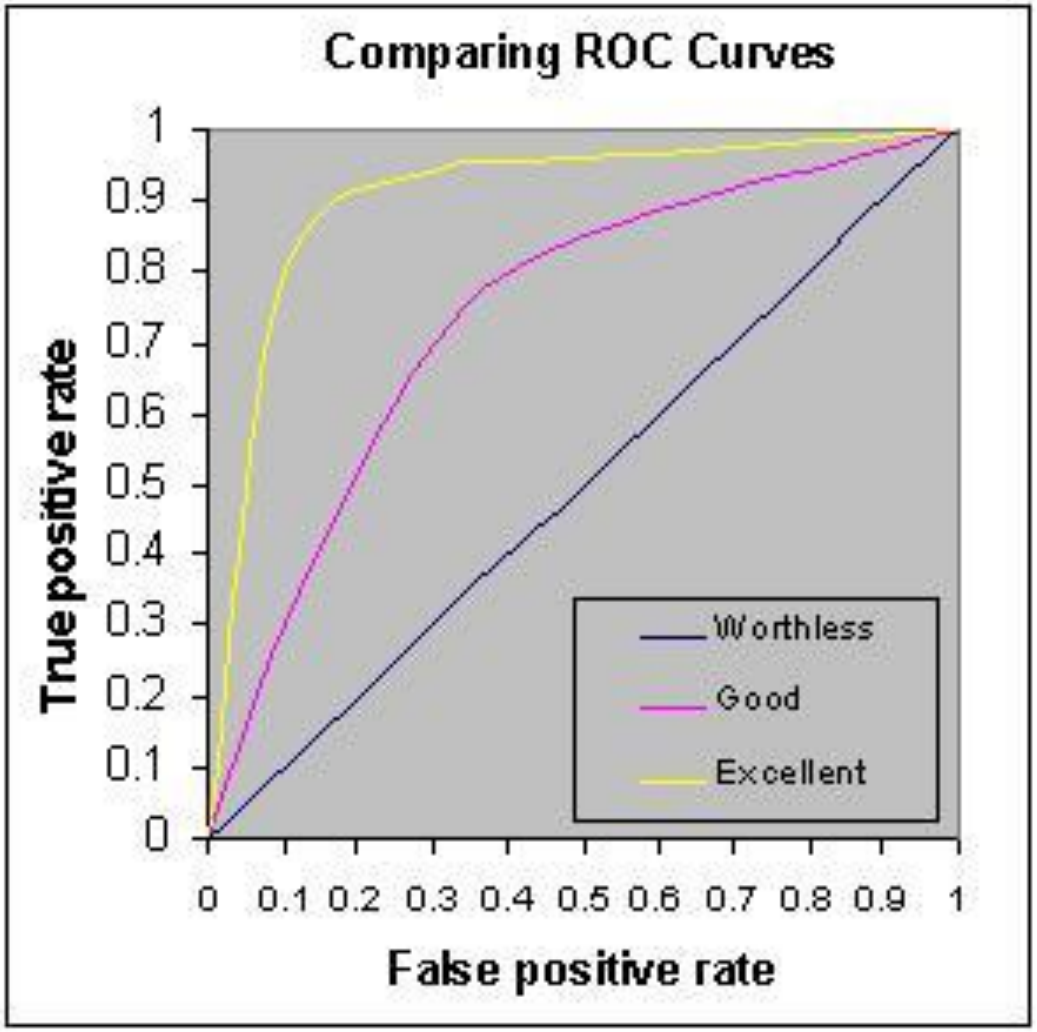
$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad \text{True Negative Rate} = \frac{tn}{tn + fp}$$

Precision/recall example



ROC curve


Comparing ROC Curves
True positive rate
False positive rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Worthless
Good
Excellent

Agenda – Hold-out evaluation

- ~~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)

The basic rule in model validation

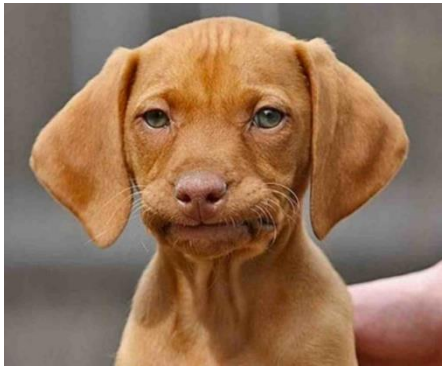
Don’t test your model on data it’s been trained with!

Testing on ‘realistic data’

- Can you evaluate the learned model on real-life/real use case data? → Ideal

- Sometimes you can only evaluate on simulated data

- Testing on biased datasets can lead to inaccurate estimation of performance



I thought our results were better

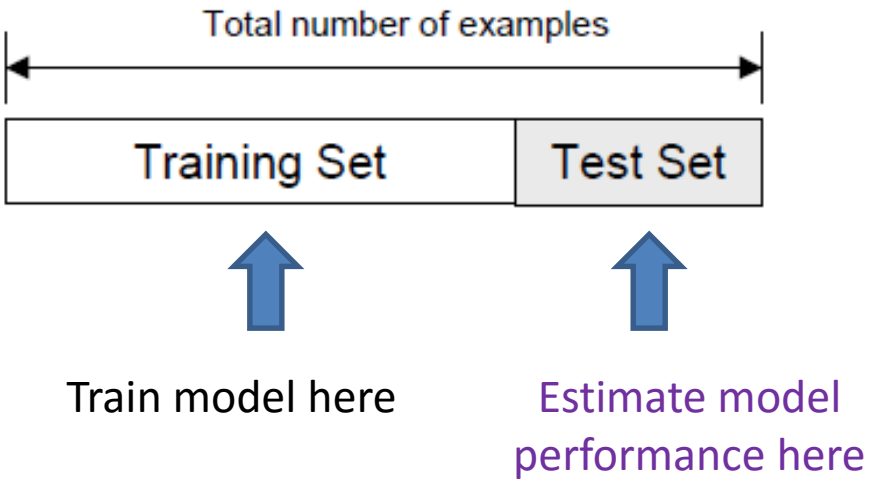
Random selection of data vs bias

- Unbiased data:
  - Data that represents the diversity in our target distribution of data on which we want to make predictions. I.e., it is a random, representative sample. This is a good scenario

- Biased data:
  - Data that does not accurately reflect our target population.
  - Example – we only asked people with glasses who they will vote for, instead of asking a random sample of the population.
  - Take home message – training on biased data can lead to inaccuracies.
  - There are methods to correct bias

Holdout test set: The naïve approach

- Randomly split the entire dataset into:
  - Training set: A dataset used for training the model
  - Test set (a.k.a validation set): Data only used for testing the model



Holdout test set – considerations

- Can the data be split randomly?
  - If not, it may lead to underfitting (remember me )
  - Test set and training set should be interchangeable, except for data size

- Do we have enough data?
  - Example: Guess if a car is American or Japanese based on 50 examples?

    Will the test set be big enough to accurately report our model performance?

The three-way split

- **Training set**

  A set of examples used for learning

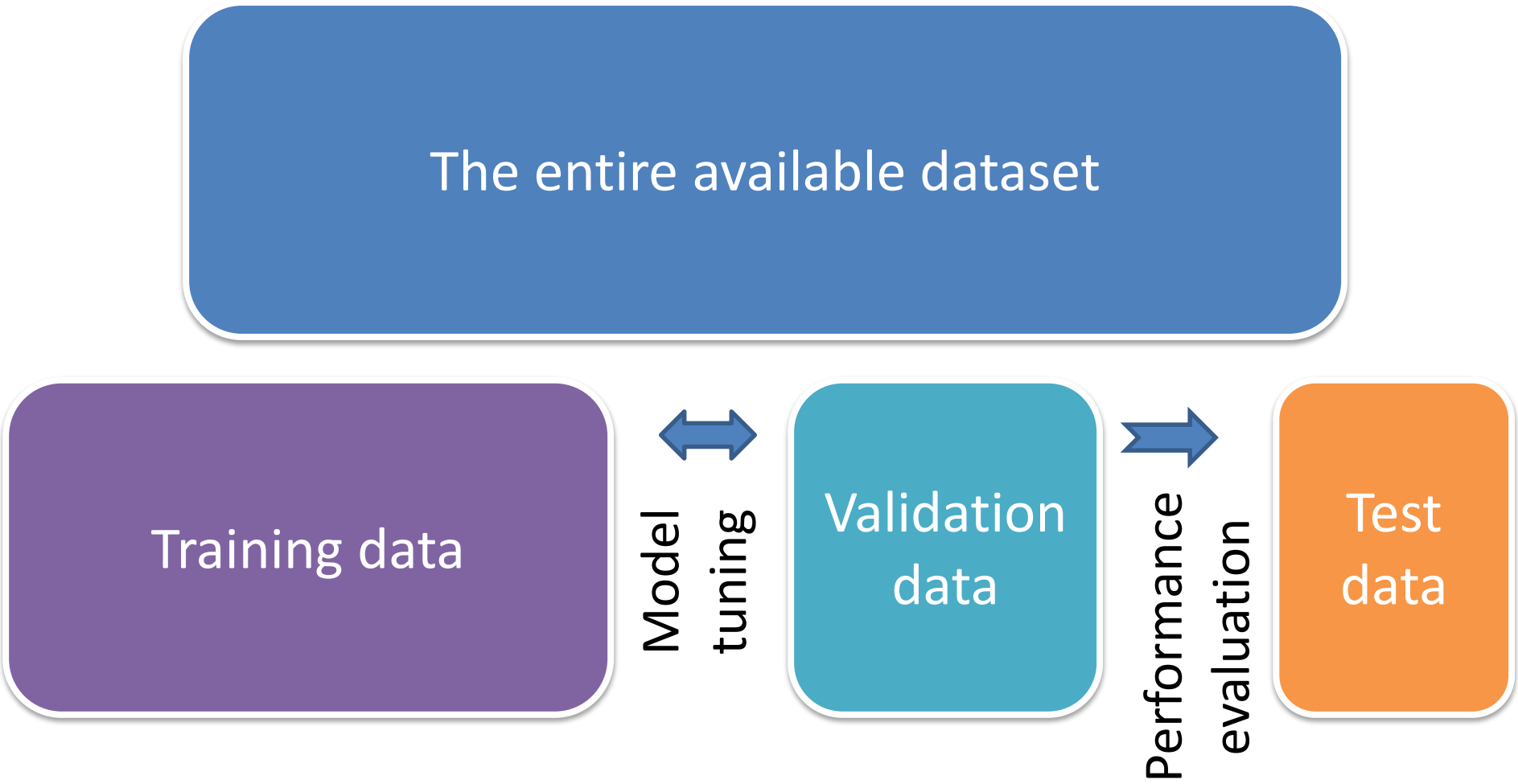
- **Validation set**

  A set of examples used to tune the parameters of a classifier

- **Test set**

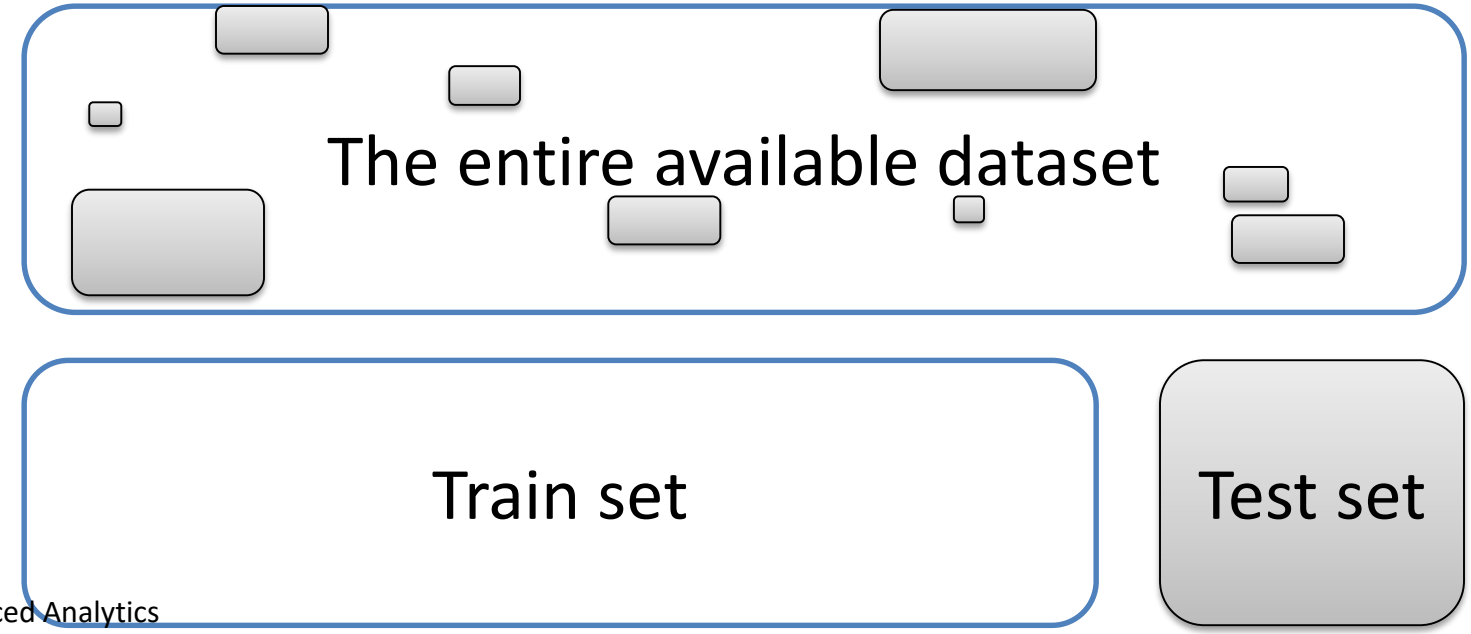
  A set of examples used only to assess the performance of fully-trained classifier. After assessing the model with the test set, YOU *MUST NOT* further tune your model (that's the theory anyway… - in order to prevent 'learning the test set' and 'overfitting')

The three-way split



How to perform the split?

- How many examples in each data set?
  - Training: Typically 60-80% of data
  - Test set: Typically 20-30% of your trained set
  - Validation set: Often 20% of data
- Examples
  - 3 way: Training: 60%, CV: 20%, Test: 20%
  - 2 ways: Training 70%, Test: 30%



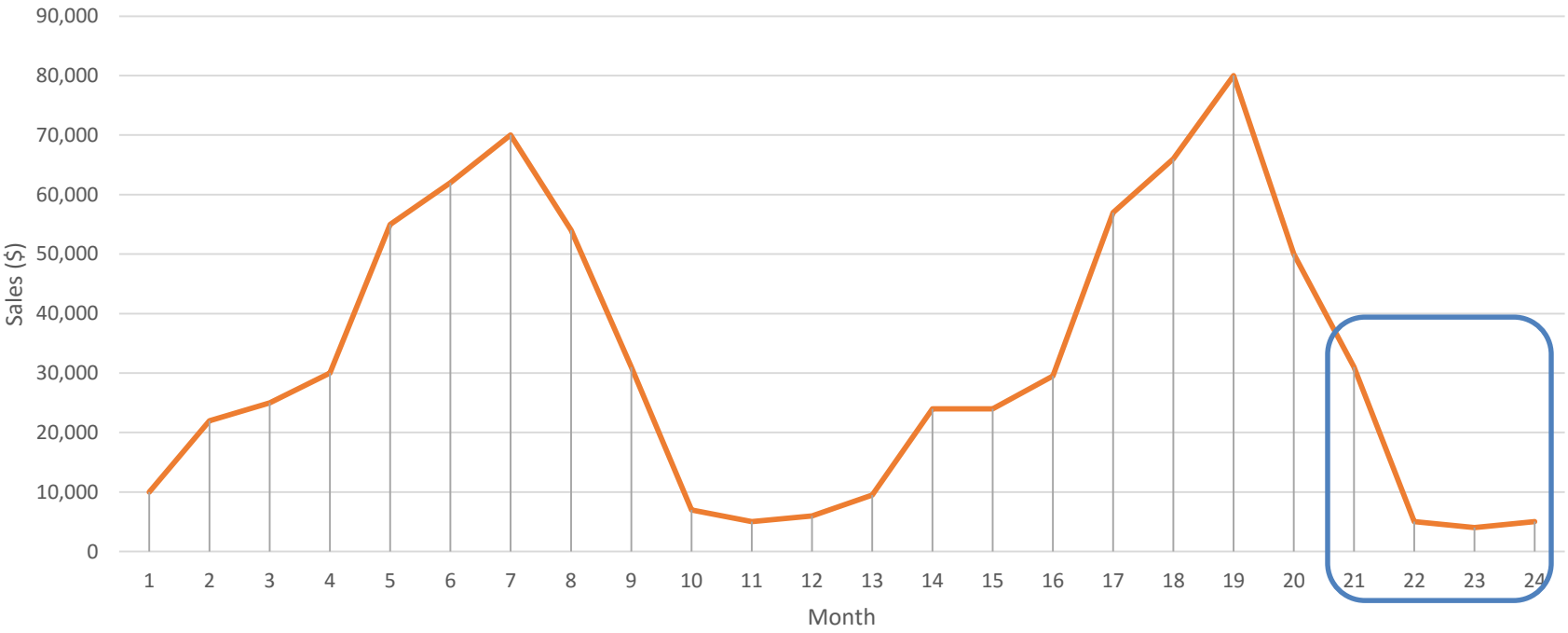
How to perform the split?

- Time based – If time is important

Can you use Jul 2011 stock price to try and predict Apr 2011 stock price?

How to perform the split?

- Some datasets are effected seasonality



Is it a good test set?

How to perform the split?

- Many more possible scenarios…
- Key messages:
  - Evaluation plan: Very important at the beginning of the project
  - Realistic evaluation: Such plan should reflect the real life scenarios your model should work on
  - External evaluation by third party is the gold standard (but not always realistic)

Holdout summary

- Good news:
  - Intuitive
  - Usually easy to perform
  - Considered the ideal method for evaluation

- Drawbacks:
  - In small datasets you don't have the luxury of setting aside a portion of your data
  - The performance will be misleading if we had unfortunate split

Agenda – Cross validation

- ~~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)

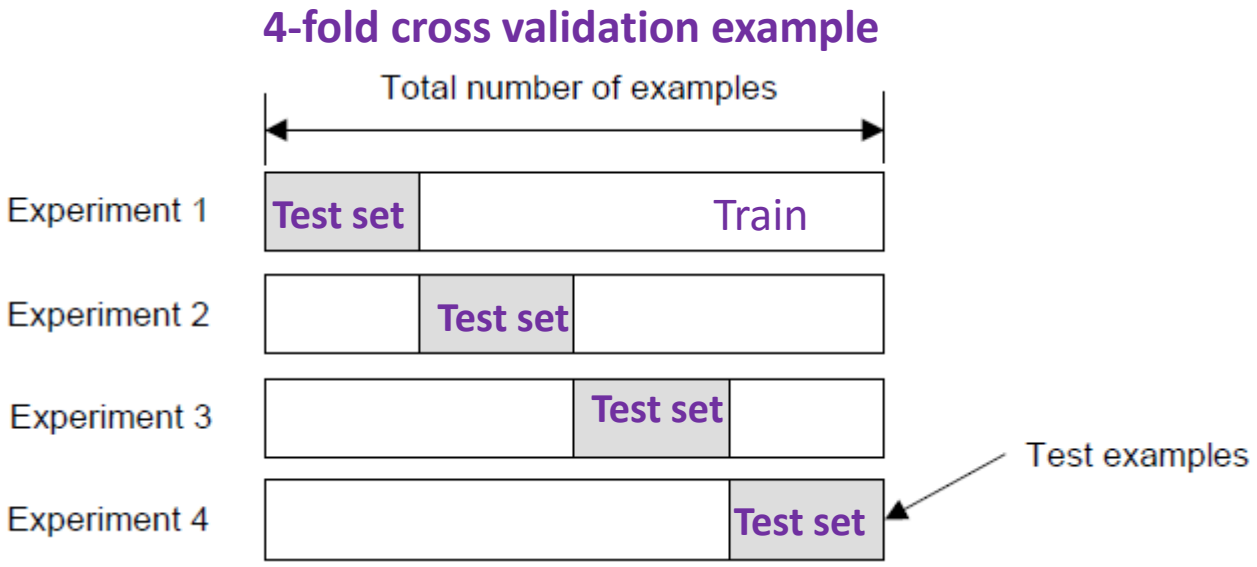
Cross validation

- Popular method for validating models

- Cases when it is often used
  - When a hold-out plan is difficult to design or was not designed at the beginning
  - When the dataset is small, and you want to use all the data for good model training

Cross - Validation

- Create a K-fold partition of the dataset
  - For each of the K experiments, use K-1 folds for training and the remaining one for testing



Cross - Validation

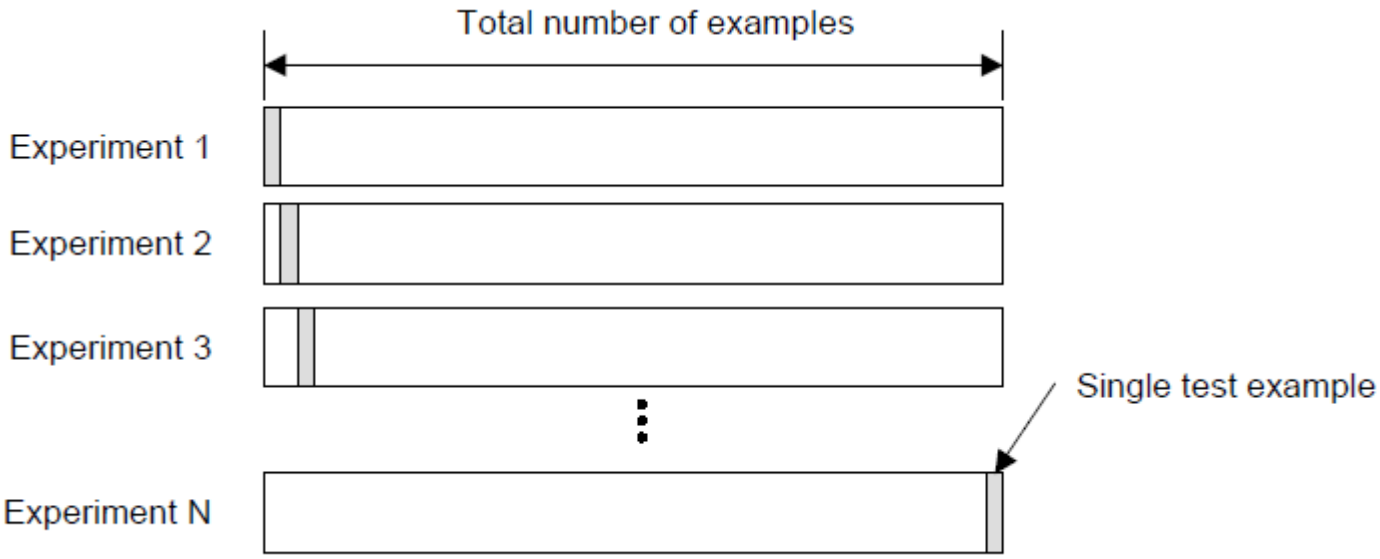
- How do you summarize the performance?
  - **Average:** Usually average of performance between experiments

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

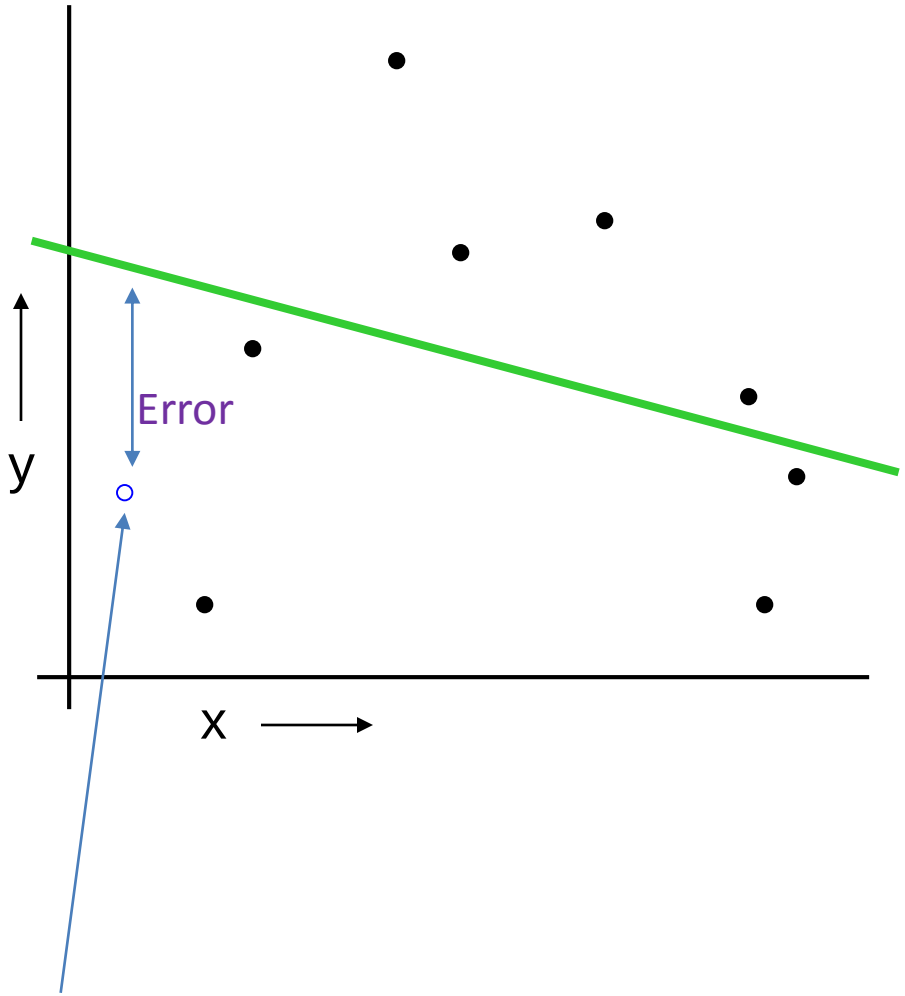
- How many folds are needed?
  - Common choice: 5-fold or 10-fold cross validation (probably since these numbers sound nice)
  - Large datasets → even a 3-Fold cross validation will be quite accurate
  - Smaller datasets → we will prefer bigger K
  - Leave-One-Out approach (K=n). In this case the number of folds K is equal to the number of examples n. Used for smaller datasets

Leave-One-Out Cross Validation (LOOCV)

- Leave-One-Out is the degenerate case of K-Fold cross validation, where K is chosen as the total number of examples



LOOCV in action



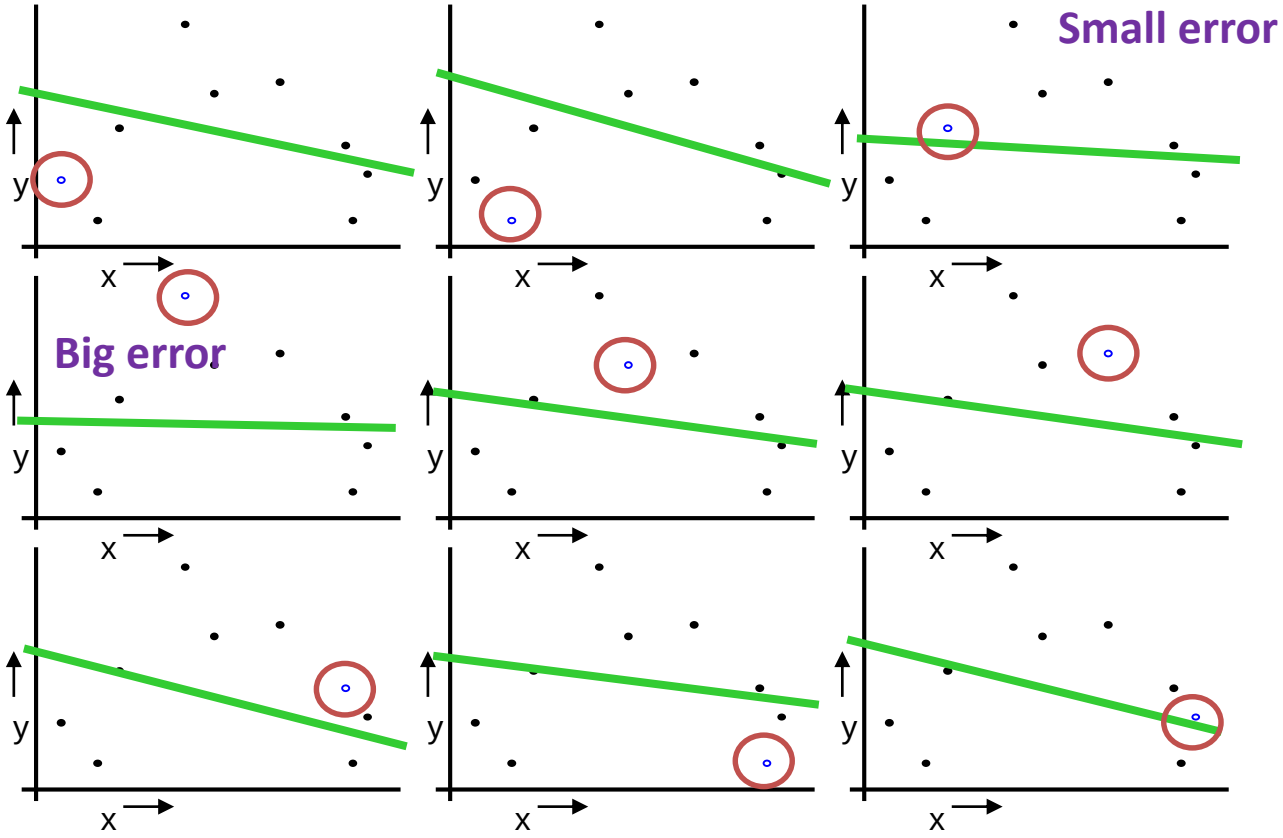
For k=1 to R, do:

1. *Let (x_k,y_k) be the k'th record*
2. *Remove (x_k,y_k) from the dataset*
3. *Train on the remaining R-1 datapoints*
4. *Measure your error (x_k,y_k)*

When you've done all points, report the mean error.

I was left out ☹. You trained without me

LOOCV in action



$$MSE_{LOOCV} = 2.12$$

Cross-Validation summary

- Good news:
  - Good for small datasets - training on all the data
  - No need to plan hold-out dataset(s), which can be challenging sometimes

- Drawbacks:
  - Partitioning is not always easy to do
  - Can be less believable than hold-out data (especially 3rd party holdout).

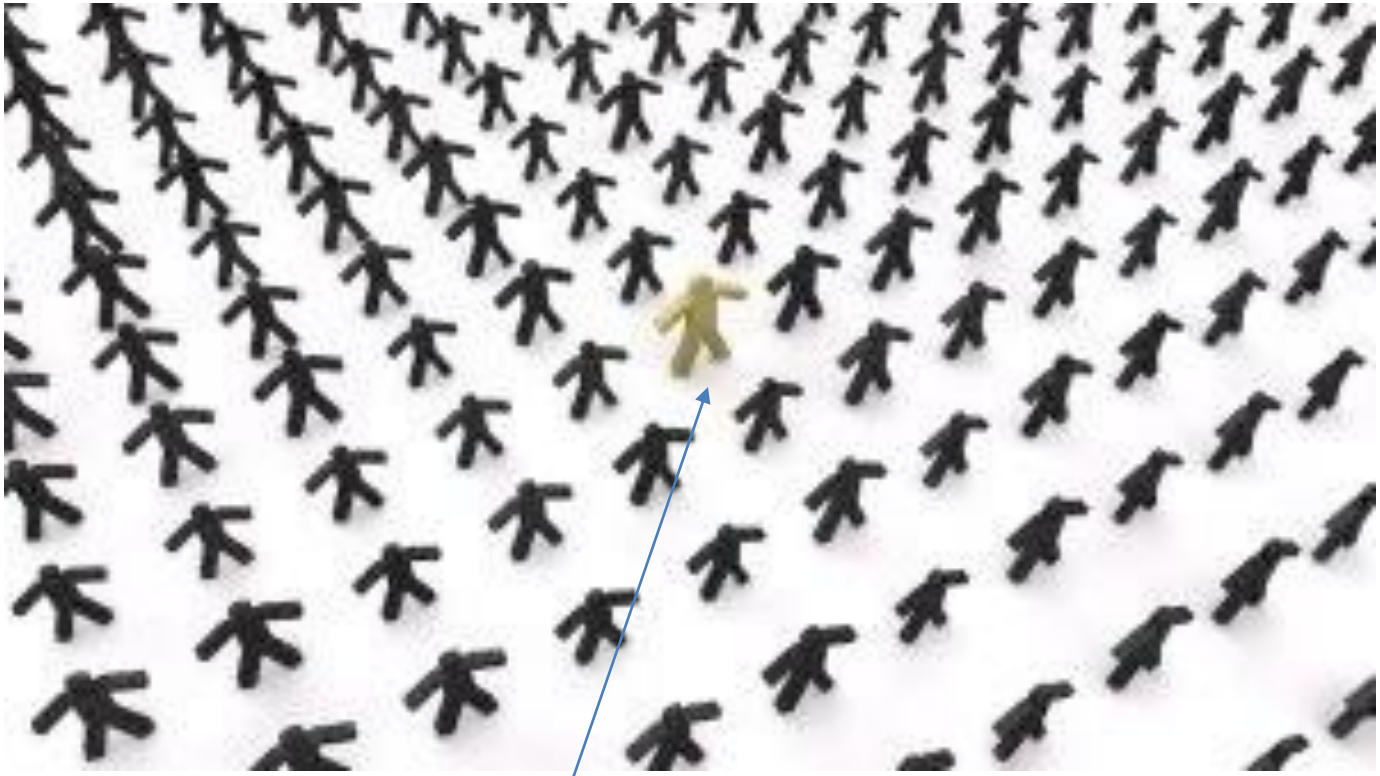
Agenda – Imbalanced classes

- ~~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)

Dealing with imbalanced classes



Class 1: Short people

Class 2: Tall people

Can we build a classifier that can find the needles in the haystack?

Imbalanced Class Sizes

- Sometimes, classes have very unequal frequency
  - Cancer diagnosis: 99% healthy, 1% disease
  - eCommerce: 90% don't buy, 10% buy
  - Security: >99.99% of Americans are not terrorists

- Similar situation with multiple classes

- This creates problems for training and evaluating a model

Evaluating a model with imbalanced data

- Example: 99% of people do not have cancer
- If we simply create a 'trivial classifier' – predict that nobody has cancer, then 99% of our predictions are correct! Bad news! – we incorrectly predict nobody has cancer
- If we make only a 1% mistake on healthy patients, and accurately find all cancer patients
  - Then ~50% of people that we tell have cancer are actually healthy!

Learning with imbalanced datasets is often done by balancing the classes

- **Important: Estimate the final results using an imbalanced held-out (test) set**

- How to create a “balanced” set
  - Treat the majority class
    - Down-sample
    - Bootstrap (repeat downsampling with replacement)
  - Treat the small class
    - Up-Sample the small class
    - Assign larger weights to the minority class samples

Down-sample and Up-sample

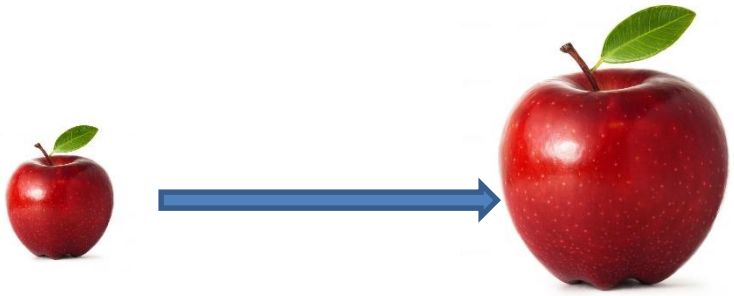
- Down-sample:
  - Sample (randomly choose) a **random subset of points** in the majority class
- Up-sample:
  - Repeat minority points
  - Synthetic Minority Oversampling Technique (SMOTE)

Bootstrapping

- Sample **m** sets from the majority so that each sets size is equal to the minority set.
- The **downsampling** is done with replacement (repeats allowed).
- Train a model of minority vs. bootstrapped sample for each bootstrap iteration
- This gives us **m** different models this is the basis of ensemble models like Random forest

Reweighting

- Idea: Assign larger weights to samples from the smaller class



- A commonly used weighting scheme is linearly by class size: $w_c = \frac{n}{n_c}$
  - Where n_c is the size of the class c and $n = \sum_c n_c$ is the total sample size

Imbalanced data can be harnessed

- The Viola/Jones Face Detector

Faces

Non-Faces

Training data

5000 faces 10^8 non faces

- P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features”, CVPR, 2001
- P. Viola and M. Jones, “Robust real-time face detection”, IJCV 57(2), 2004

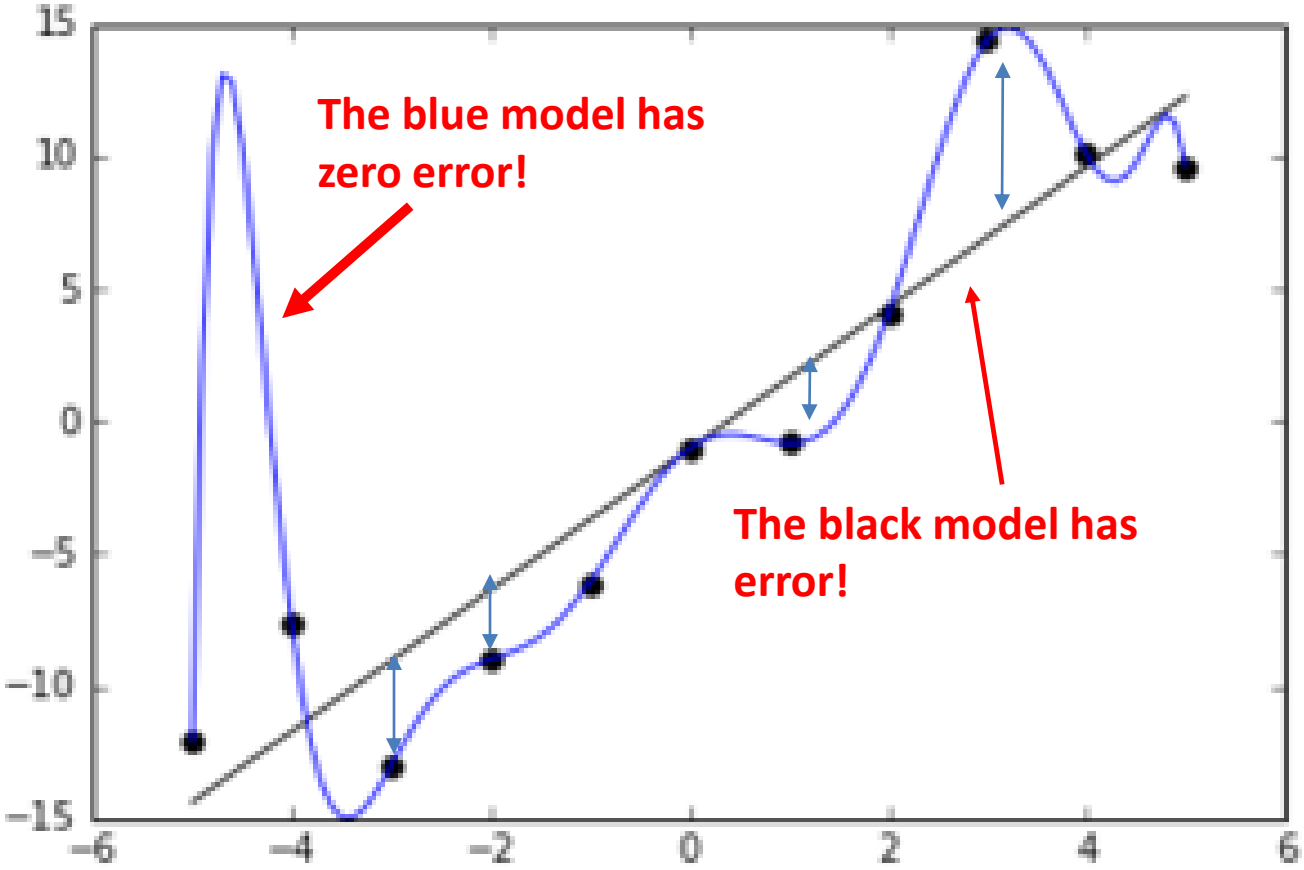
Agenda – Overfitting/underfitting

- ~~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)

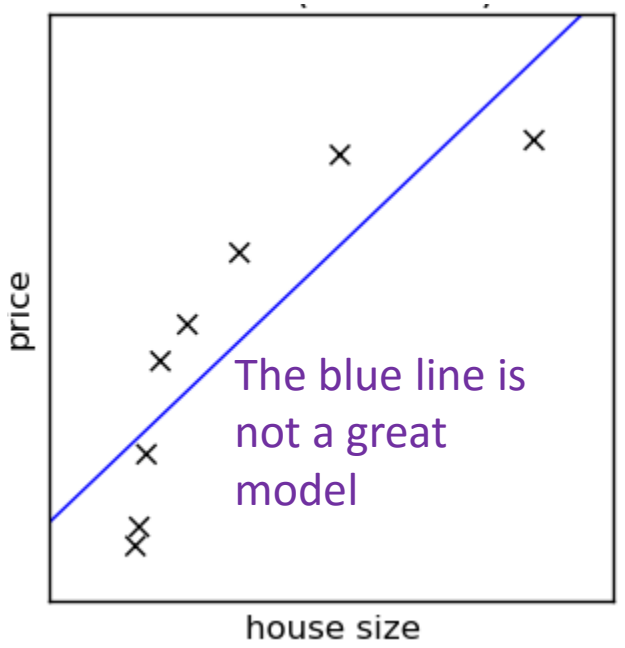
Overfitting and underfitting



So which model is better : blue or black?

Simple models cause underfitting (a.k.a high bias)

- In underfitting, the training error and test error are high



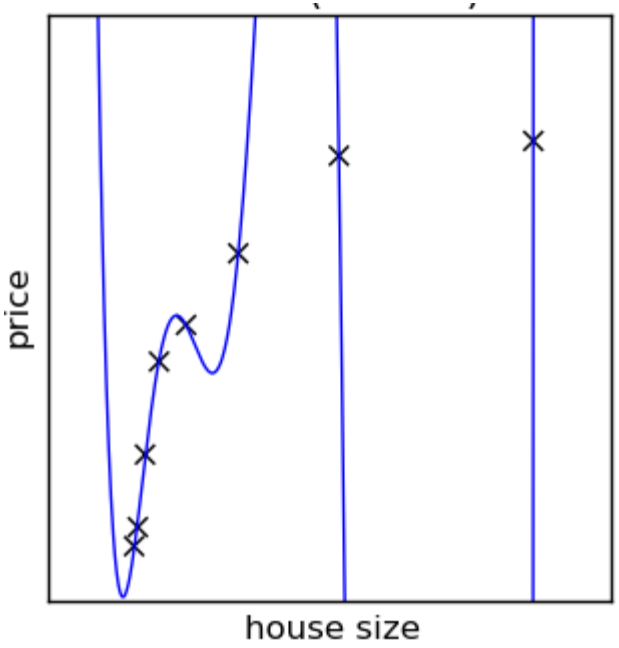
- What does too simple mean?
  - Too few features
  - Use of features is not 'complex'
- Examples
  - Can you predict well if an email is spam using only the word 'free'?

    Probably not. More features (words) are needed)
  - Can you predict well housing prices using only the year it was built?

    Probably not - More features are needed: size, location, #windows, etc.

Overfitting (high variance) is when the model learns the noise and signal

- If the model overfits
  - It cannot generalize well to new data
  - It memorizes the training data
  - It has a low training error, and high test error



The model has no error, but it does not seem to represent the data well

Overfitting is caused by:

- Too much model complexity
  - Typically too many features

- Too little data or not enough data diversity
  - Big data: The more data we have, the more complex a model we can use
  - Diversity: Redundant data does not add information and thus does not improve the model

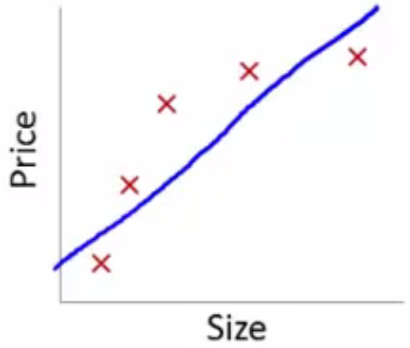
Overfitting due to the choice of too many features

- We can use many features to try and predict the price of a house

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}

Overfitting due to poor model choice

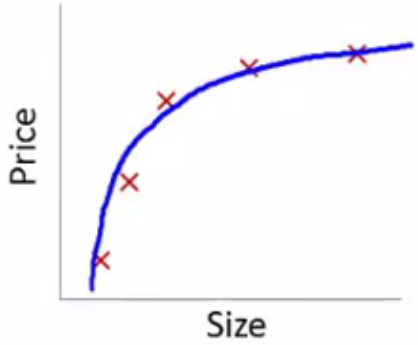
Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$

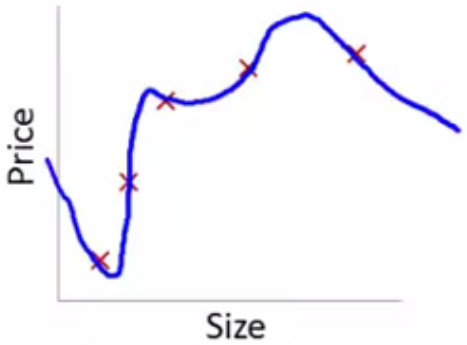
"Underfit" "High bias"

Not many features (only x1)



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



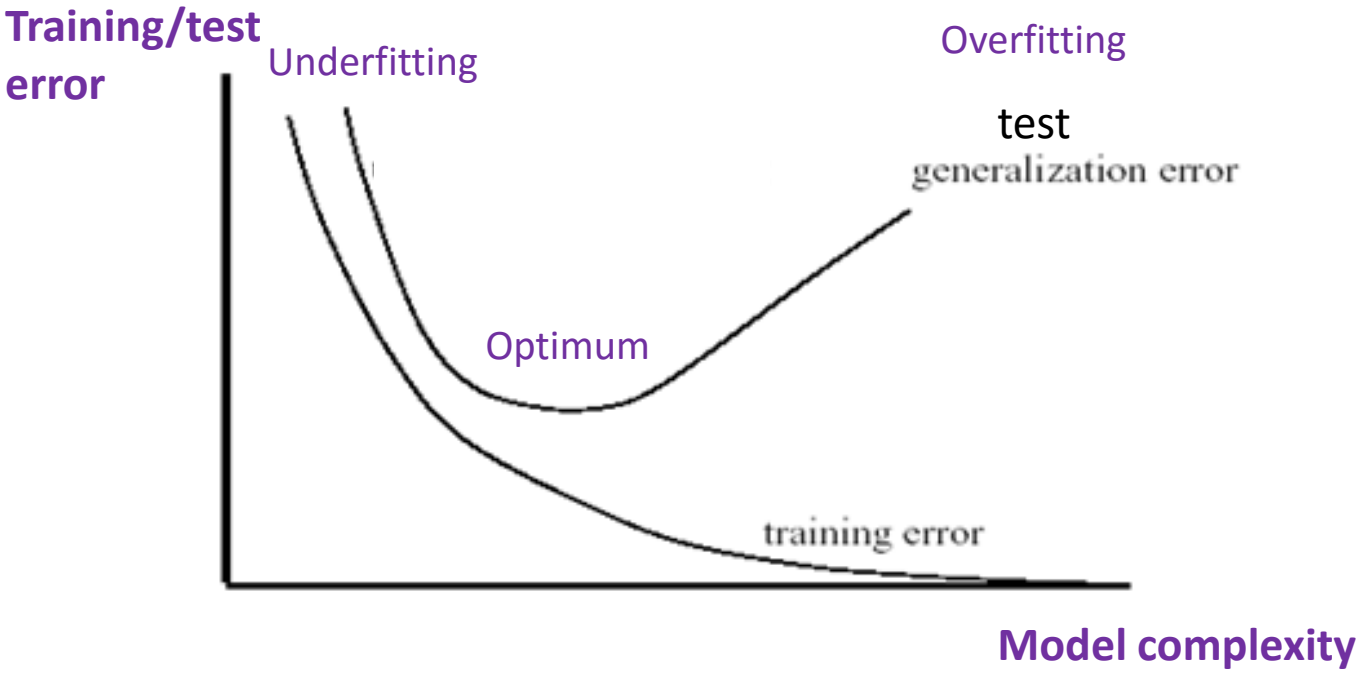
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit" "High variance"

Too many features (x1 to x4)

Andrew Ng

We can plot learning curves to spot overfitting



Addressing Overfitting

- More Data → a problem that comes up many time

- Reduce # of features or dimension
  - Select which features to keep
  - Reduce dimension
  - Select which model to use

- Regularization
  - Keep all the features but penalize some features/ values of parameters
  - This is particularly useful when we have a lot of features, each contributing a bit to the prediction

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 d (# dimensions) is large in relation to n (number of samples).

- Real life examples:
  - Genomics
    - We have ~20k genes, but disease sample sizes are often in the 100s or 1000s.

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

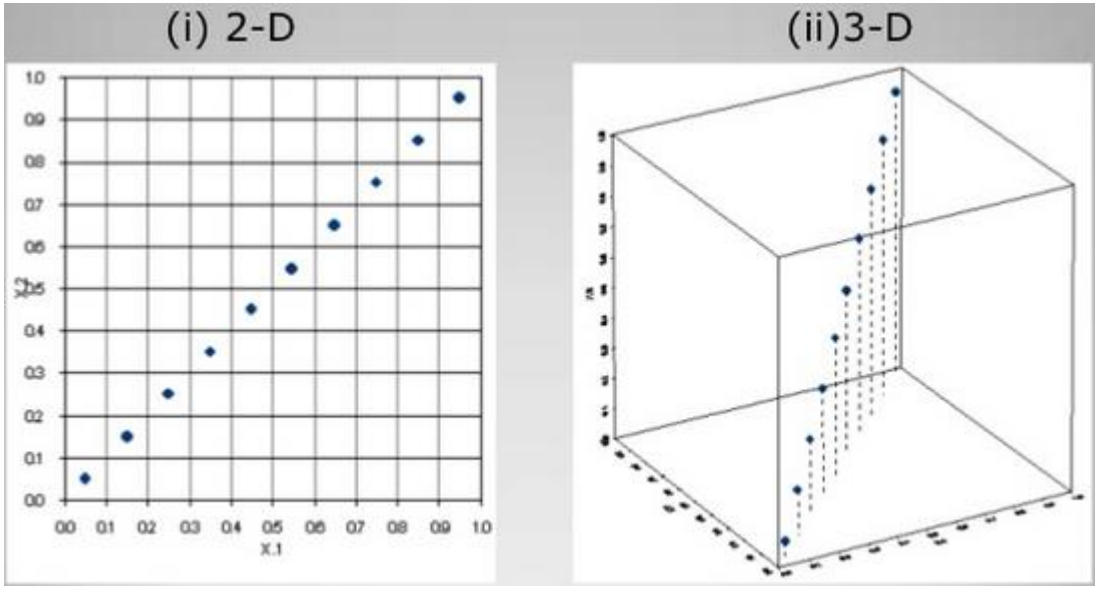
- **Noisy data**: More features can lead to increased noise → it is harder to find the true signal

- **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

Data becomes sparse as dimensions increase

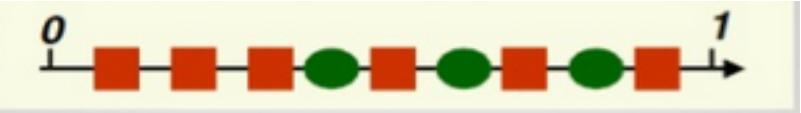
- A sample that maps 10% of the 1x1 squares in 2D represent only 1% of the 1x1x1 cubes in 3D



- There is an exponential increase in the search-space

Sparse example 2

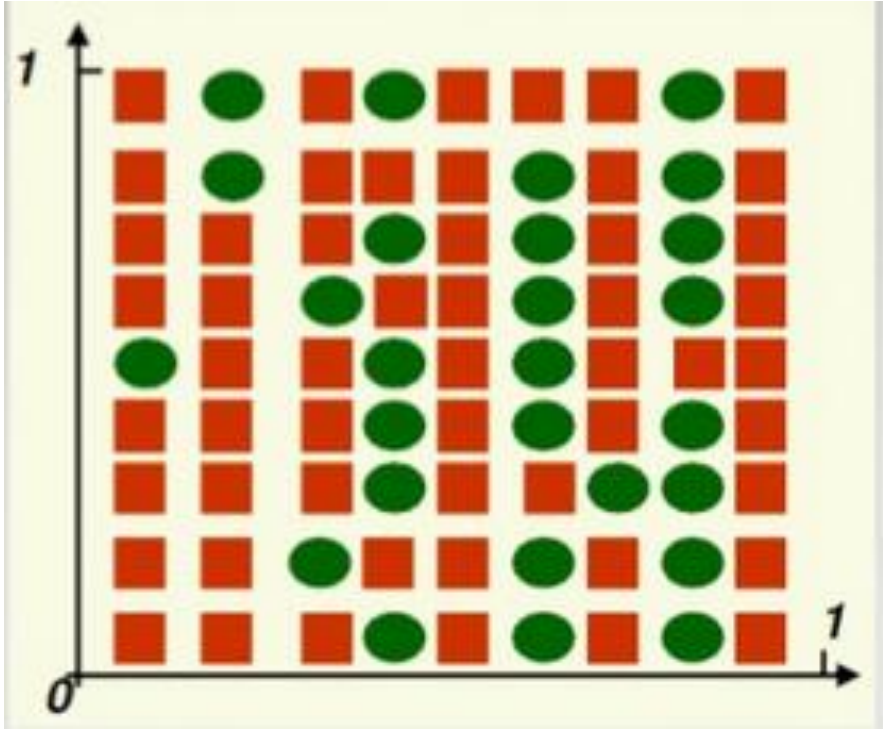
- Suppose we want to discriminate between samples from two categories



- This one feature is not discriminative. We cannot classify well, thus we decide to add a second feature.
- To maintain the sample density of 9 samples per unit length (as above) how many samples do we need?

Number of samples

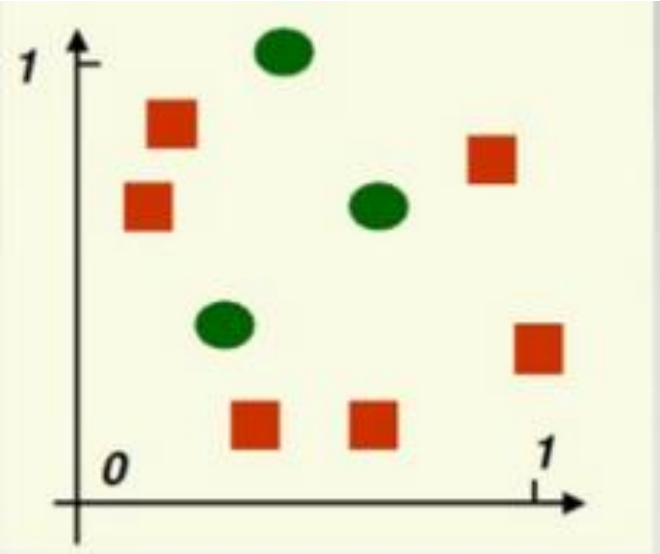
- We need 81 samples to maintain the same density as in 1D



- In d dimension we need 9^d samples!
  - Otherwise, the data becomes more sparse

Number of samples

- We can't always gather new data points, we still have only 9 points.



- The data becomes sparse. Imagine how it would look if we could plot 3D or 4D.

Curse of dim - Running 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.
  - Compute = $

Curse of dim - Some mathematical (weird) effects

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

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

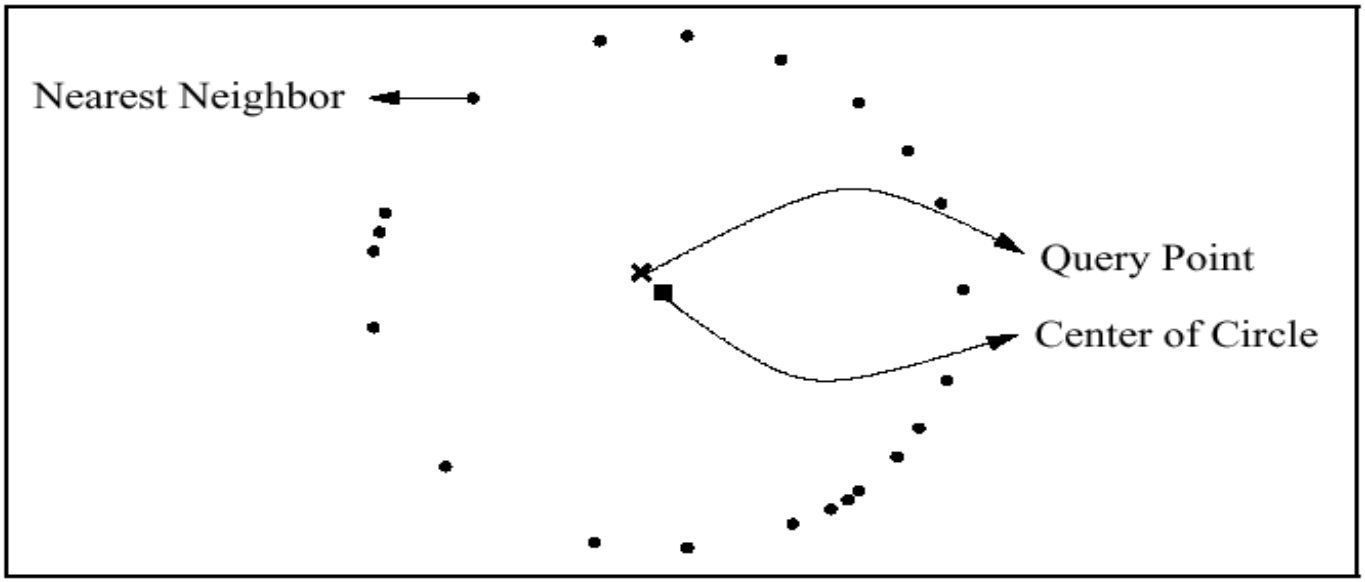
- When **d** tends to infinity the volume tends to zero

- 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

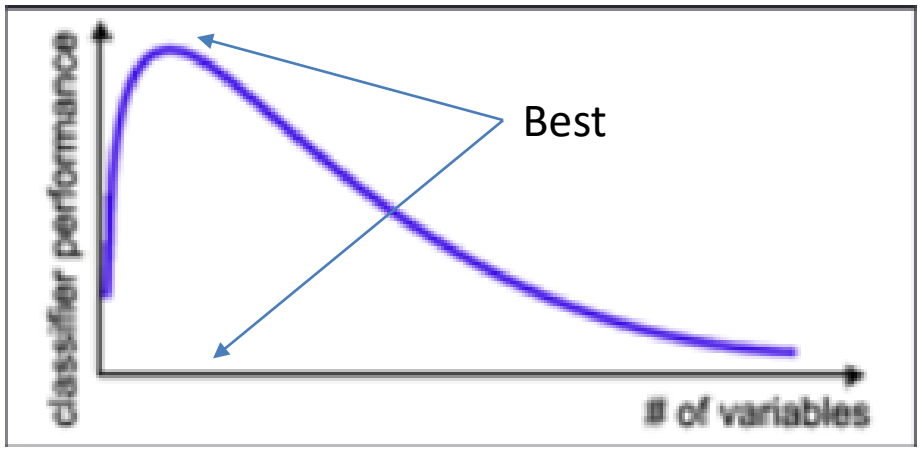
- How many dimensions does the data intrinsically has here?
  - Two!
    - x1 = ½ * x2 (no additional information)
    - x4 is constant

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

- Less is More
  - In many cases the information that is lost by discarding variables is made up for by a more accurate mapping/sampling in the lower-dimensional space



Agenda – Filter methods

- ~~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)

Feature selection goals

- **Features selection is** the choice of a subset of features (variables/attributes) from all available features to be used in the learning model

- Feature selection is **not always necessary**

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

Feature selection challenges

- It is a search/optimization problem
- Exhaustive testing of features is NP-hard and 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

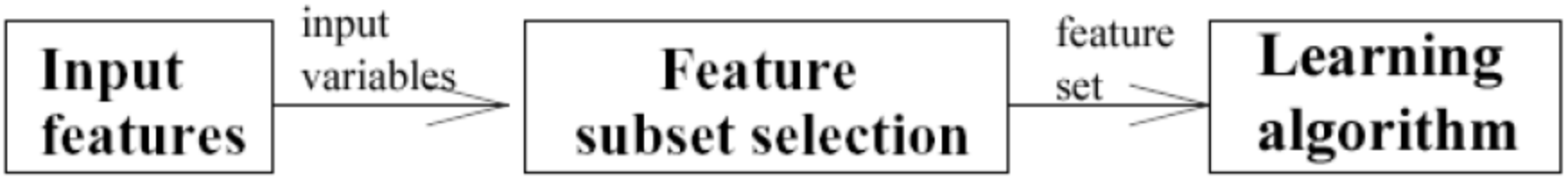
- **Filter method:** Ranks features or feature subsets independently of the classifier
  - Low computational power
  - Independent of model type

- **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, **independently of the learning model**



- Relatively fast
- Not tuned by a given learner (it's good and bad ☺)
- Often used as a preprocessing step for other methods

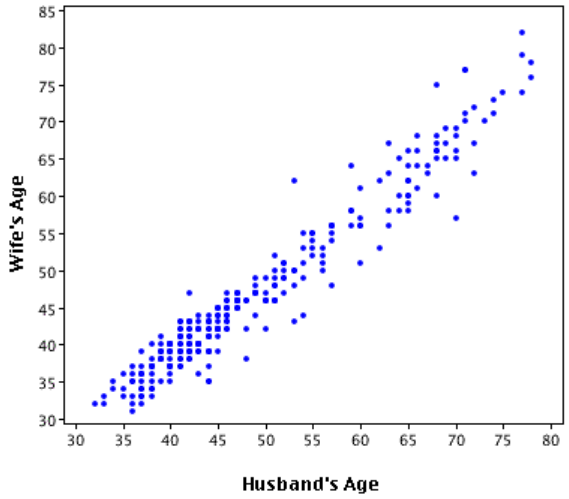
Filter methods in practice

- What is it?
  - Selection of highest ranked features according to a some scoring function
  - Very efficient, reasonable results, often preferable to other
- What is the scoring function?
  - It is a measure (such as correlation) with respect to
    - Label
    - Other features (redundancy, diversity, etc.)
    - The feature itself

Common scoring functions

Pearson Correlation

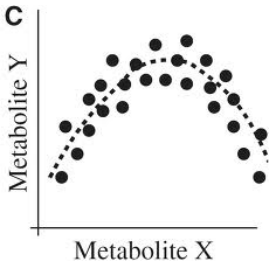
- Can only detect linear dependencies

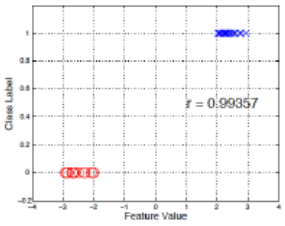


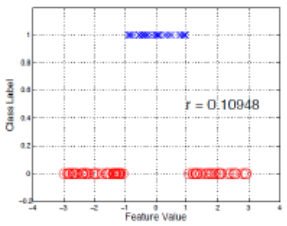
Mutual information (MI)

"How much information two variables share"

- Can detect any form of statistical dependency







Feature ranking examples

- Label association:
  - Choose top 10 features associated (Pearson correlation) to the label
- Low variation features:
  - Remove features with little variation
- Correlated features:
  - Keep only one of two highly correlated features

Multivariate filter methods challenges

- Search strategy to select candidate subset
  - Guide and direct the exploration of the space

- Evaluation strategy - An objective strategy to evaluate these candidates
  - Provide a feedback to the search strategy for the relevancy ("goodness") of the candidate subsets

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

Wrapper methods

- Definition: Wrapper methods refer to the use of a learning model to choose features. The model is NOT 'the model' that is eventually used.

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

- Benefit:
  - Often good selection of feature subsets

- Drawbacks
  - Requires training a model on each feature subset iteration → very costly
  - Result vary for different learning models

Wrapper methods in practice

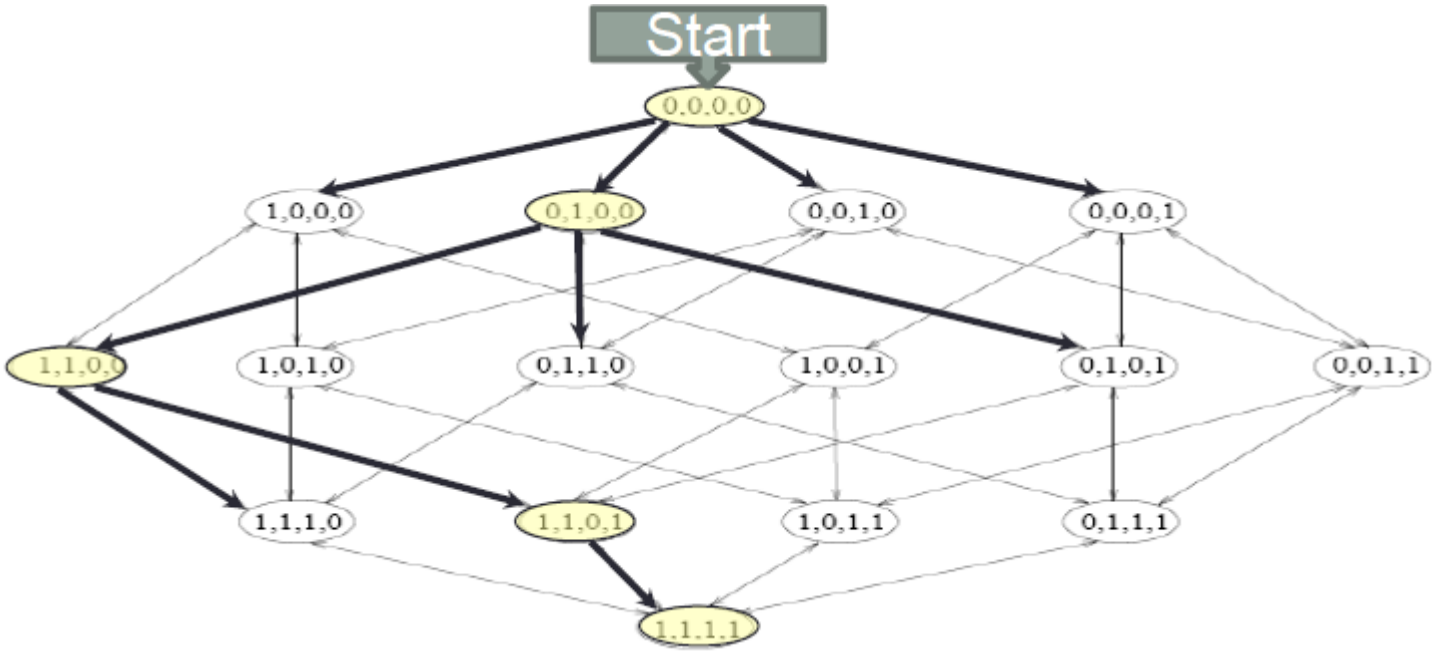
- Often preceded by filter methods to reduce computational cost

- 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

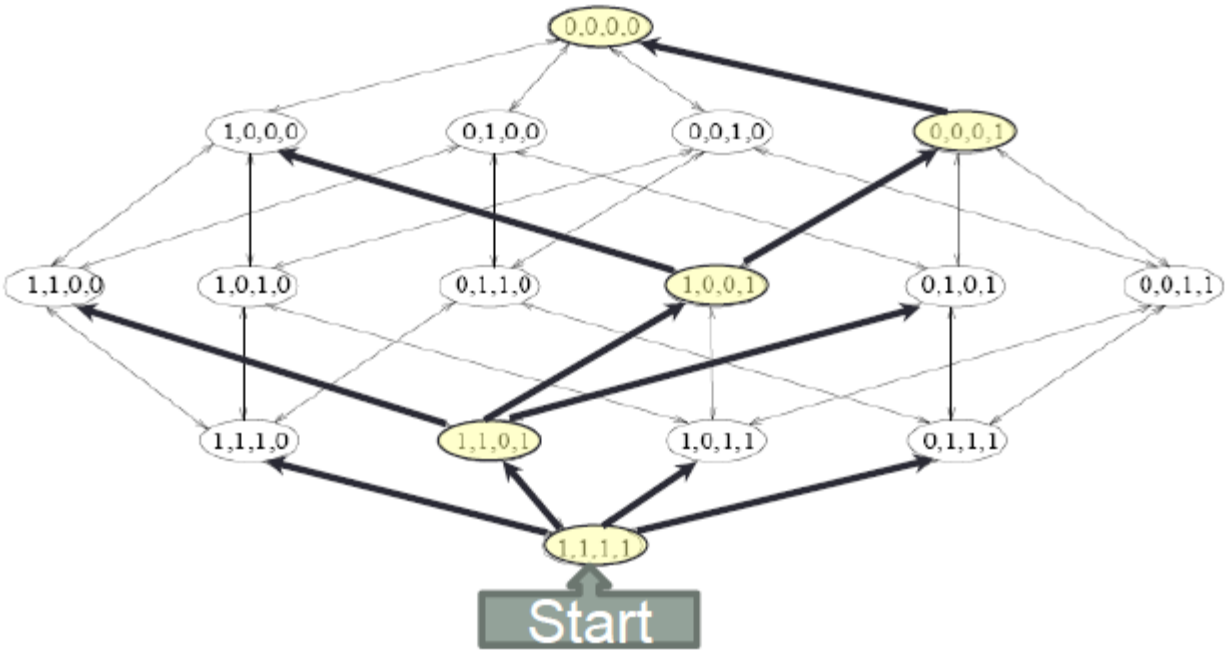
- Evaluation is usually done on validation set (like hold-out) or using cross-validation

Sequential Forward Selection (SFS)



SFS perform better when the “optimal” subset has a small number of features

Sequential Backward Selection (SBS)



SBS perform better when the “optimal” subset has a large number of features

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

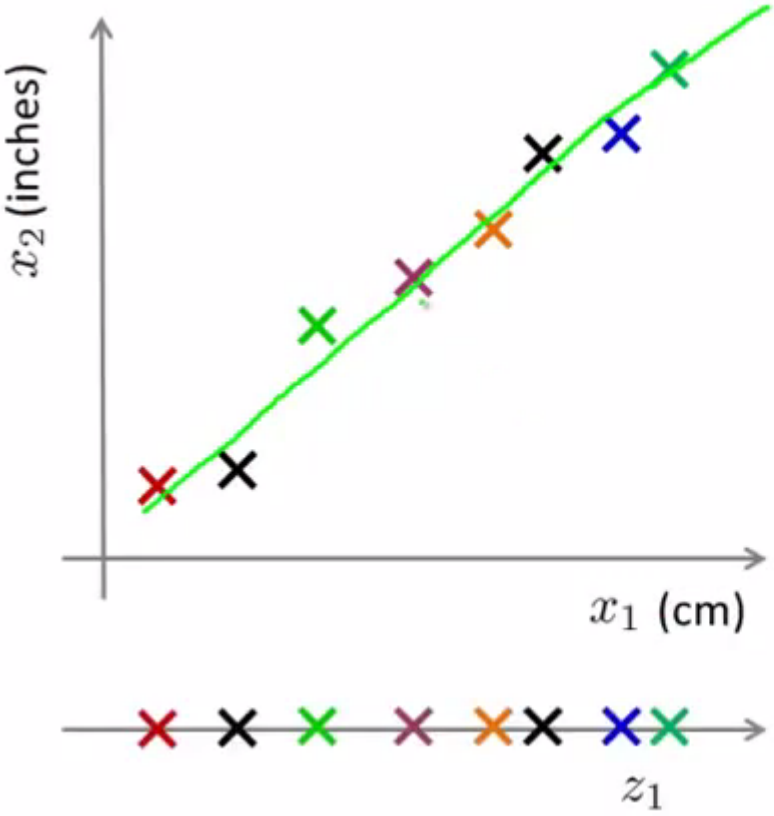
Dimensionality reduction goals

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

Example – reducing data from 2d to 1d

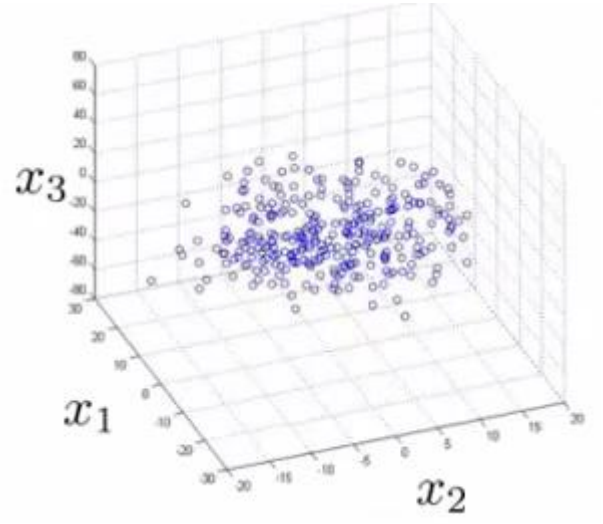
- X1 and x2 are pretty redundant. We can reduce them to 1d along the green line

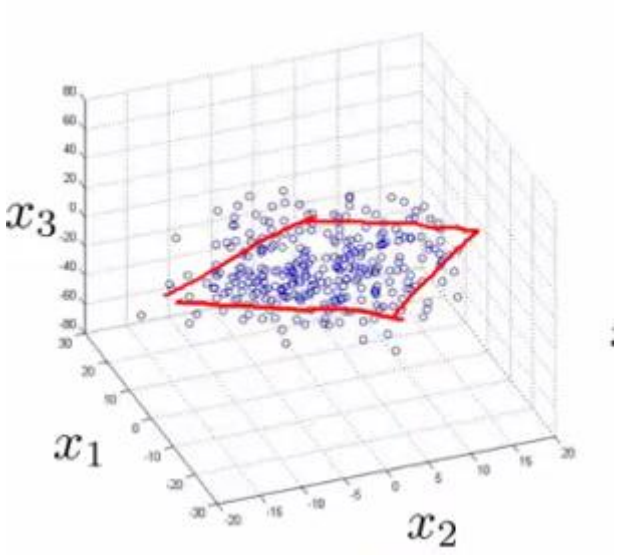
- This is done by projecting the points to the line (some information is lost, but not much)



Example – 3D to 2D (1)

- Let's have a look at a 3D dataset

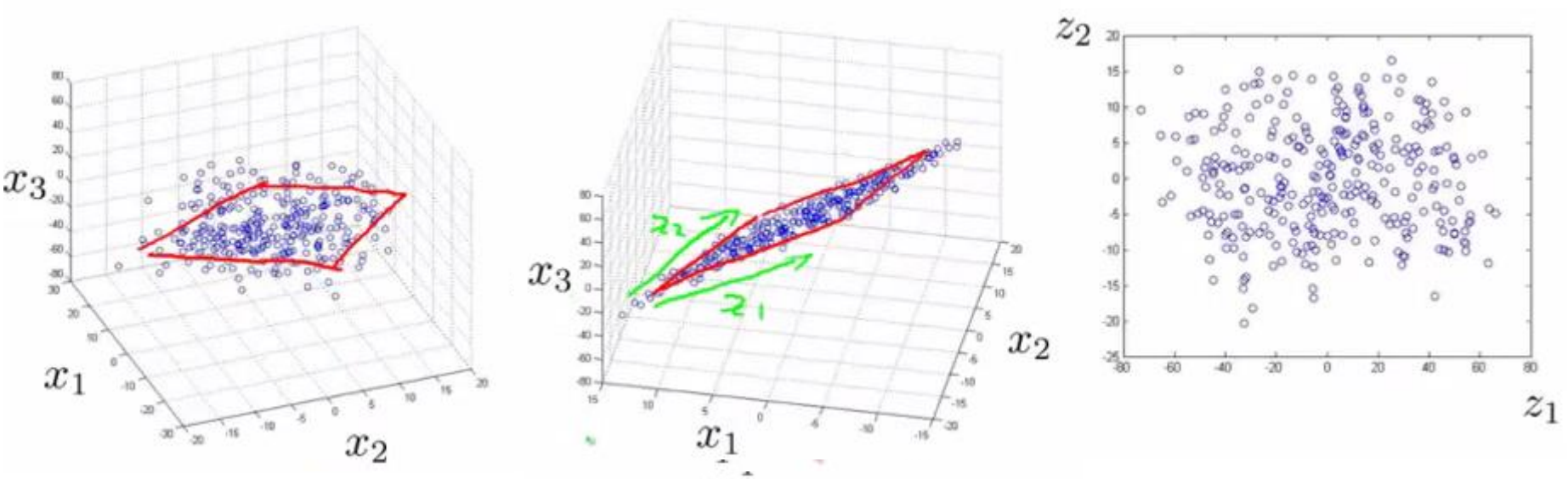




- Despite having 3D data most of it lies close to a plane

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

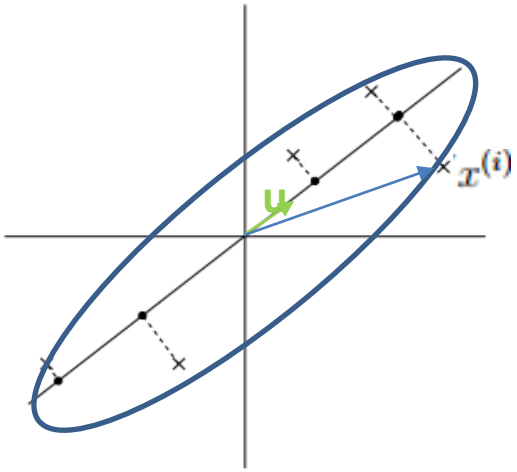
Principal Component Analysis (PCA)

- 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 - ….

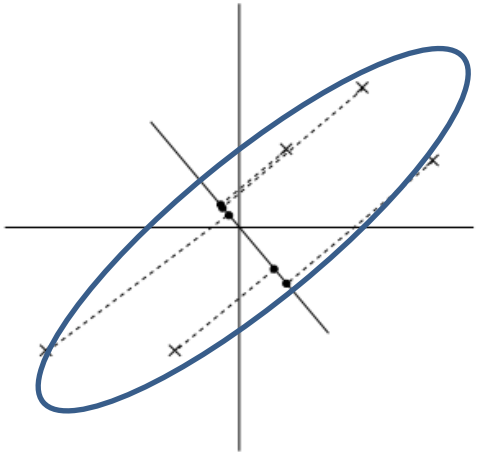
Geometrical intuition

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

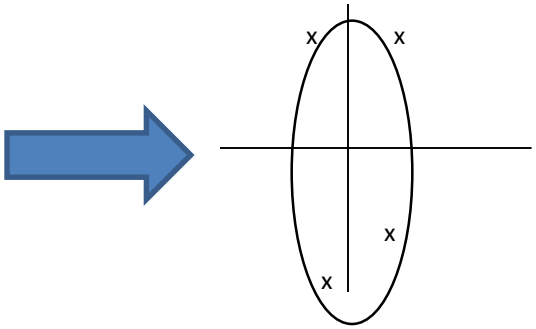
Large variance axis

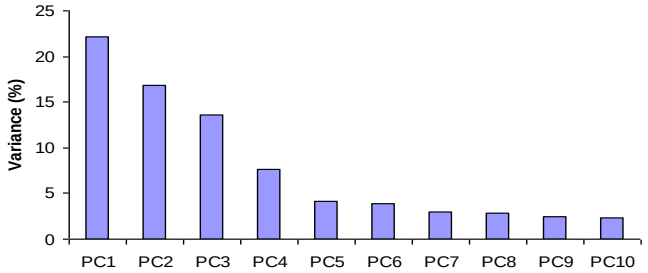


Small variance axis



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





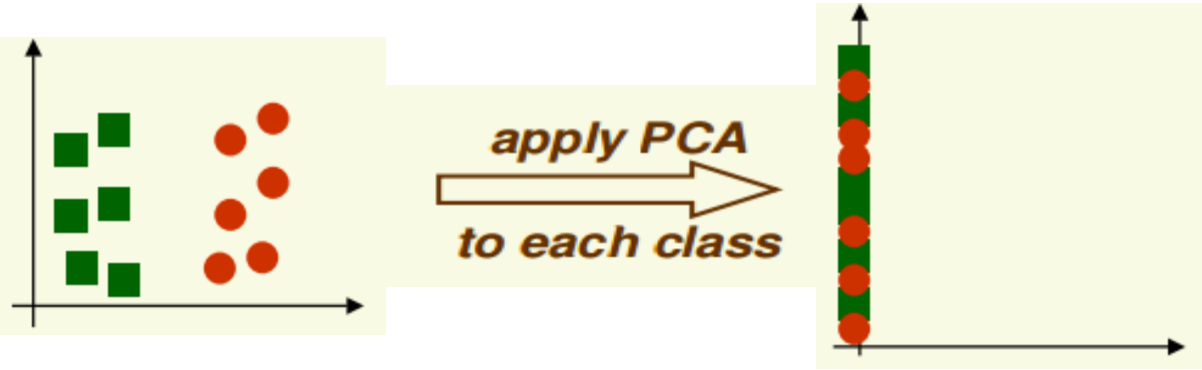
PCA algorithm

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:
   - 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's 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)

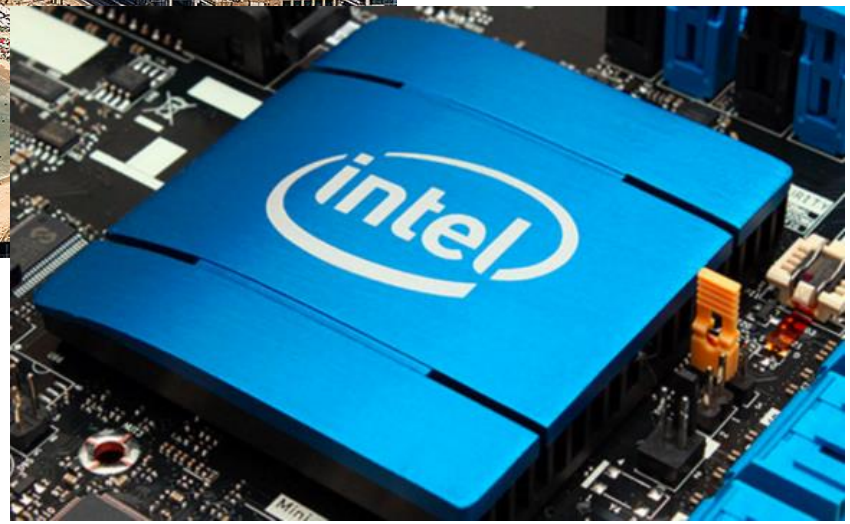
Agenda – Done

- ~~Quick ML review~~

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

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

Thank you and enjoy Israel



Zeev Waks, zeev.waks@intel.com
Senior data scientist
Intel – Advanced Analytics