Machine Learning – COMS3007

Applying ML to Real Data

Benjamin Rosman

Based on course notes by Chris Williams,
Victor Lavrenko, and Amos Storkey

Previously on ML…

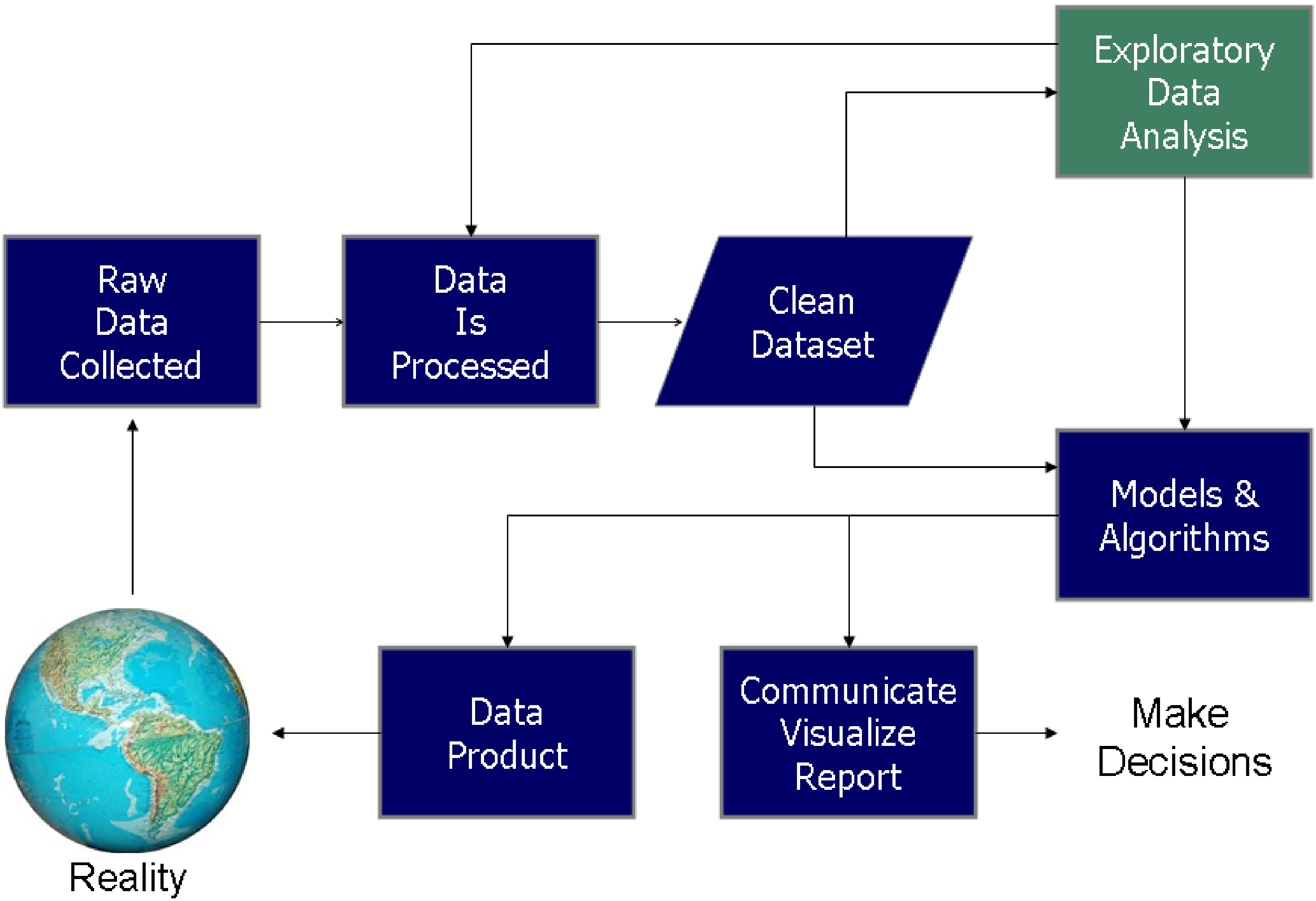
- We've now looked mainly at supervised learning, and touched on unsupervised learning
- Two main types of supervised learning:
  - Classification:
    - $y \in \{0,1\}$ (or more)
  - Regression:
    - $y \in \mathbb{R}$

- How do we actually use this set of techniques to solve real problems?

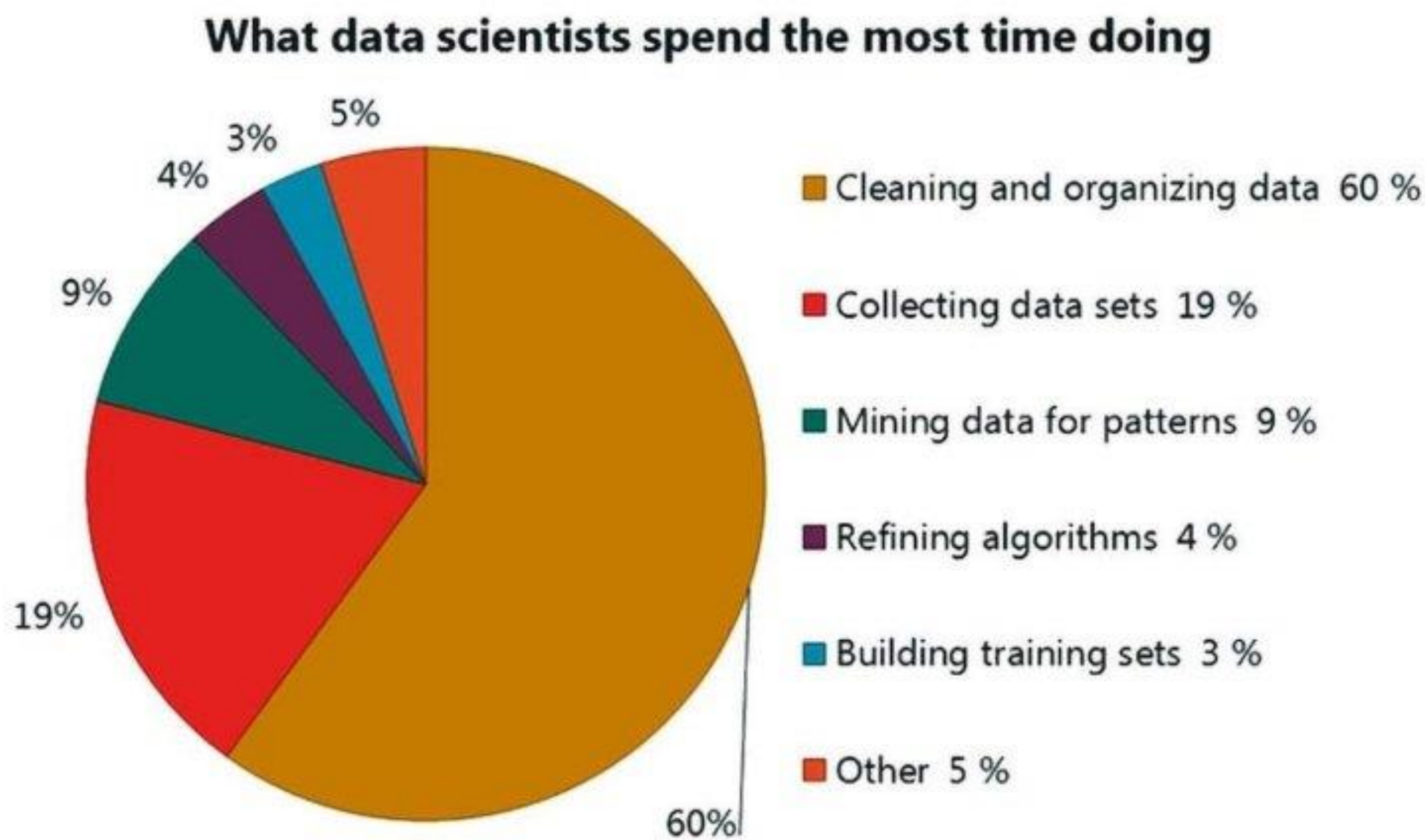
- We've now looked mainly at supervised learning, and touched on unsupervised learning
- Two main types of supervised learning:
  - Classification:
    - $y \in \{0,1\}$ (or more)
  - Regression:
    - $y \in \mathbb{R}$

- How do we actually use this set of techniques to solve real problems?

The data science process



Time breakdown



Why are you doing this?

- **Building new and better algorithms**
  - Go find datasets with the right properties

- **Answering scientific (or other) questions**
  - Go collect the data you need

- **Addressing business questions (for clients/employers)**
  - Given data sets
  - May need to collect more
  - May need to augment with public data

Collecting data

- Where does the data come from?
  - Sensors, online data sets, social media, surveys, company records
  - These all have different problems and types of noise
- Where do labels come from?
  - Label by hand, paying people, Mechanical Turk
  - Typically very time-intensive
- Often **augment** data sets: creating more data
  - Particularly for neural nets
  - E.g. Flip images on some axes
    - Other synthetic data
  - Pretrain with other datasets

Preprocessing and cleaning data

- This is the hard work!
- Real data is very messy!

- Are variables encoded correctly? E.g. categorical data
- Are variables normalized?
- Are there NULLs? NAs? Missing values?
- Do zeros have the correct interpretation? May indicate missing values
- Are any values artificially limited?

- Often write many small programs to clean data
- Work with visualisation techniques to find outliers, etc

Look at your data

- Most important thing you can do!

- **Inspect rows of your data**
  - Often shows up many issues immediately!
  - People often haven't structured databases correctly

- Does the data make sense?
  - What do the attributes mean?
  - Do these give you clues on what models to use? Or how certain features should be normalised?
  - Is there missing data? Why would this have happened?

Exploratory data analysis (EDA)

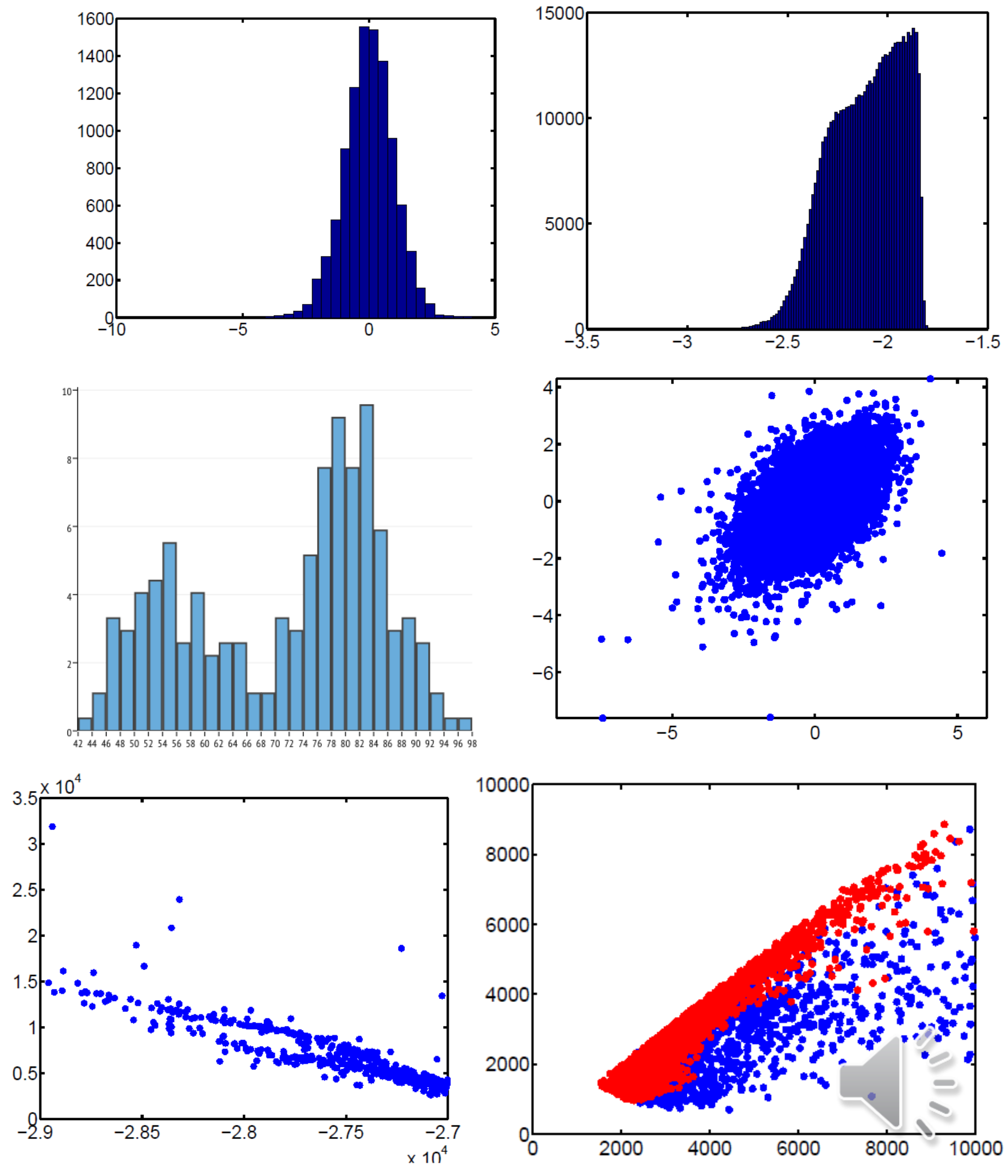
- Build a picture of what is going on in the data set

- Compute summary statistics (means, variances) of features
- Visualise features and their relationships
- Are there clear trends and dependencies in the data?

- “Play” with the data
    - Run small models on it to try understand things and test hypotheses
    - E.g. clustering: do clusters align with classes? Do they mean something else?

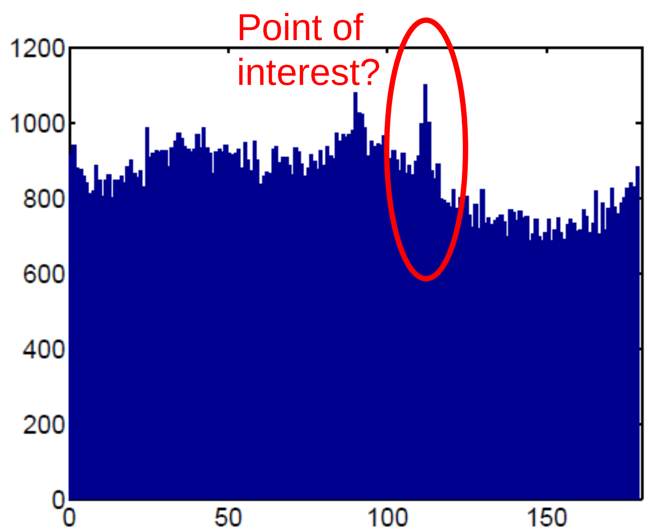
Visualising data

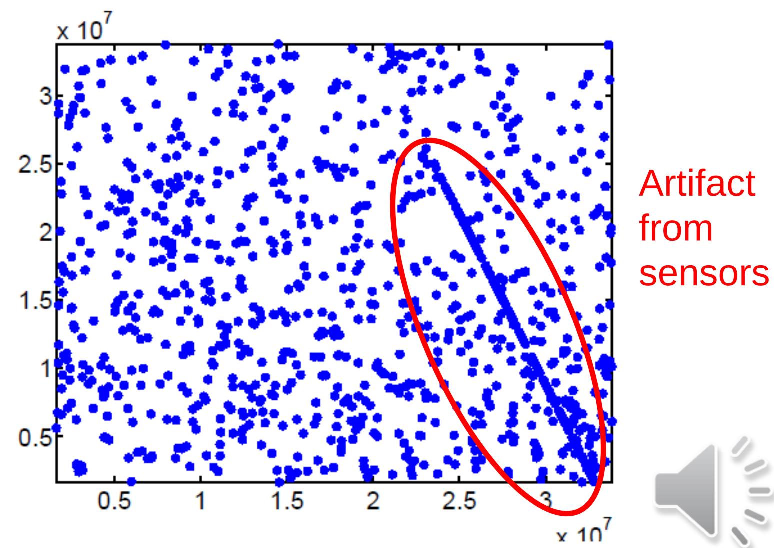
- What values do attributes take on?
- Histograms of single features:
    - What kinds of distributions?
- Plot attributes against each other
    - Dependent or independent?
    - Conditional distribution
        - Colour by class



Visualising data

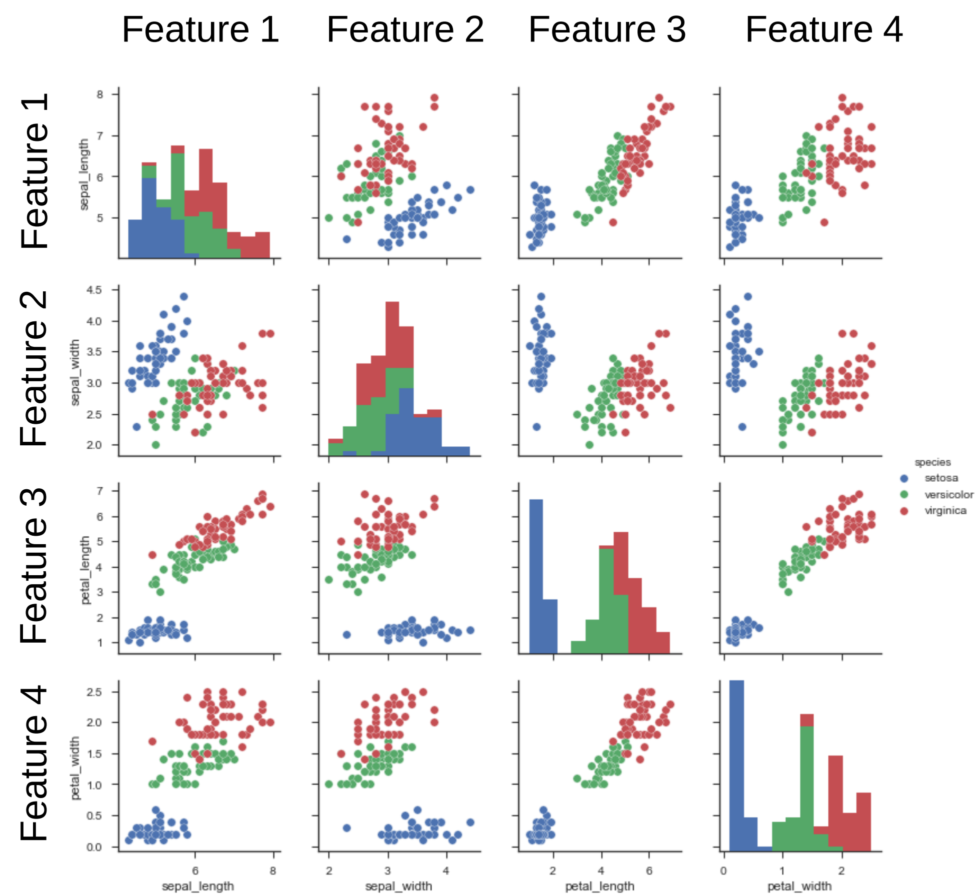
- How easily do we expect regression/classification to work?
- What features look to be useful?
- What about outliers?
    - In some tasks (anomaly detection): really useful, what we care about
    - In others: noise to be cleaned away
- How to visualise 100D data?





Scatterplot matrix

- Create a matrix of scatter plots
  - Plot the data using only two features at a time
  - With histograms of individual features
  - Label data by class (if classification)
- Shows if any features have particularly good discriminating power



Note the symmetry: only need to generate half of it

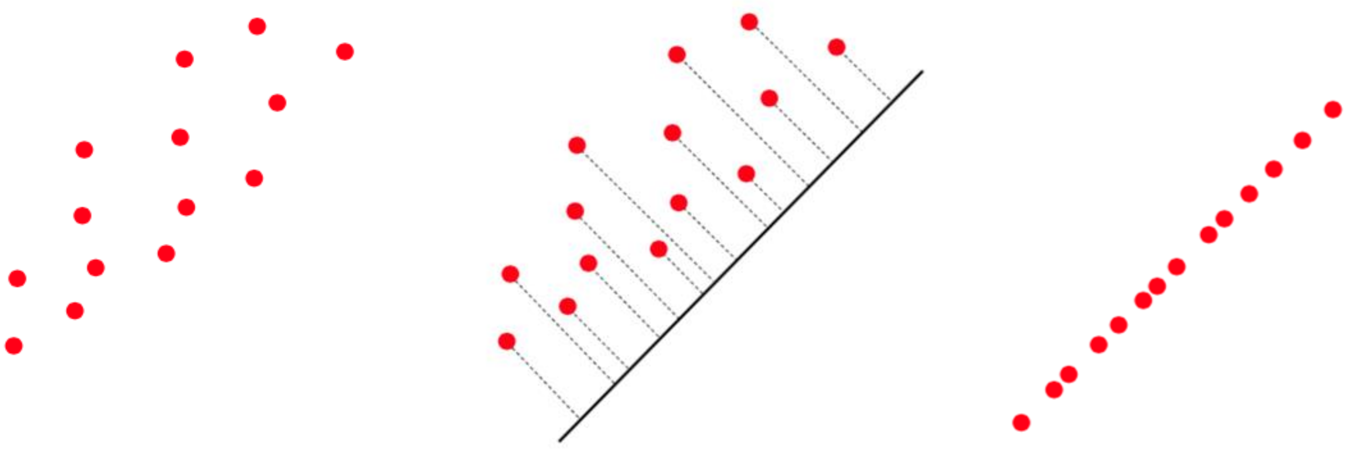
Dimensionality reduction

- Visualising is hard in high dimensional datasets
    - Scatterplot matrices are useful
    - But only pairs of features

- Instead of only looking at some dimensions/features
- Use other techniques to find lower dimensional projections of your data
    - These are actually unsupervised learning
    - We care about the underlying structure of the data

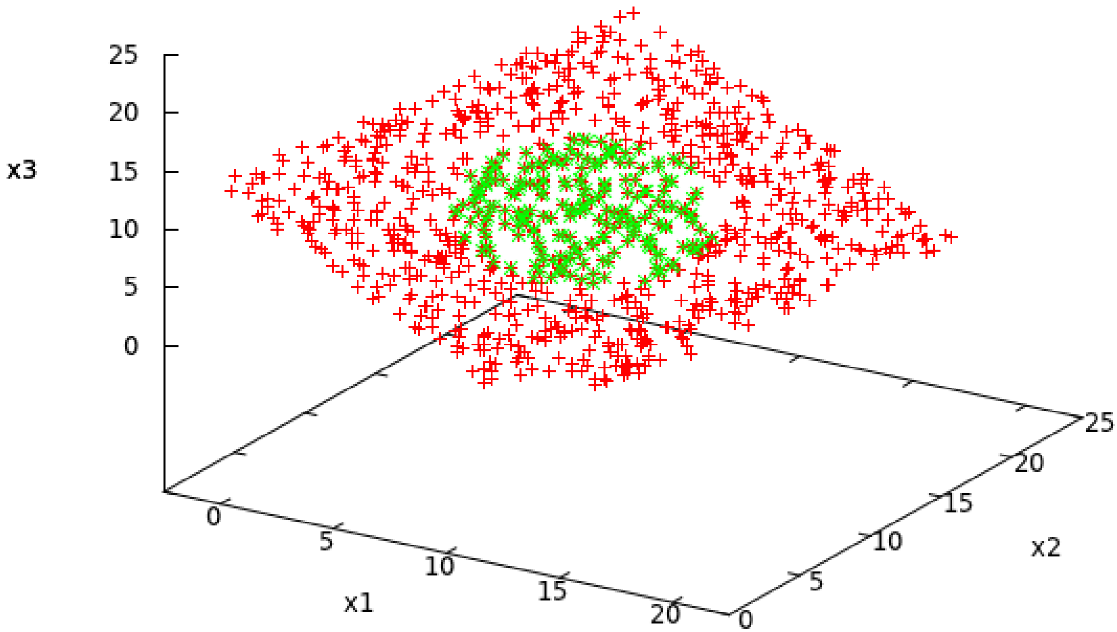
Principal components analysis (PCA)

- Linear dimensionality reduction (but many others)
- Idea: try find a (linear) lower dimensional projection of the **data**
  - Note this is done on the feature space: ignore targets

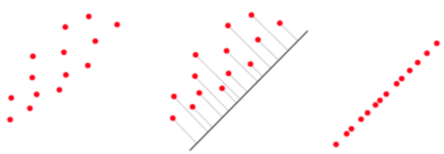


Higher dimensions

- Map high dimensional problem into lower dimensional space
- Useful for discarding features and for visualisation



Finding the best projections

- Based on covariance between features: how much are two features dependent on each other?
- Why?
  - In  : the point is that x and y are related
- Covariance between features X and Y on n data points:
  - $cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$
  - If cov > 0: both dimension increase together
  - If cov < 0: as one increases, other decreases
  - If cov = 0: independent

Compute a covariance matrix

- For each feature X, **normalize by subtracting the mean**
- For each data point i, for feature x: $x_i \leftarrow x_i - \bar{x}$
- Where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

- Then compute the covariance matrix $C = (c_{i,j})$, where $c_{i,j} = cov(X_i, X_j)$

- So, if data D = n rows (data points) of m columns (features)

- Then $C = \frac{1}{n-1} D^T D$

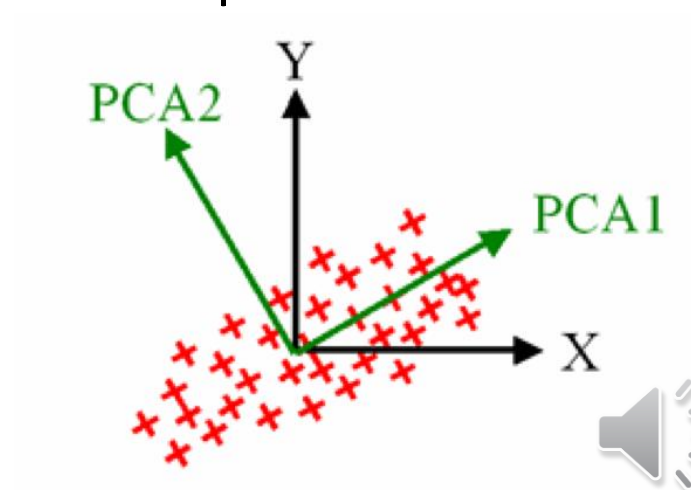
Eigenvalues and eigenvectors

- Eigenvalues and eigenvectors:
  - $(A - I\lambda)v = 0$: solve for λ and v

- Compute eigenvectors for covariance matrix $C = \frac{1}{n-1} D^T D$

- Eigenvectors are orthogonal to each other
  - So: use them as a new basis for a vector space
  - Called: principal components

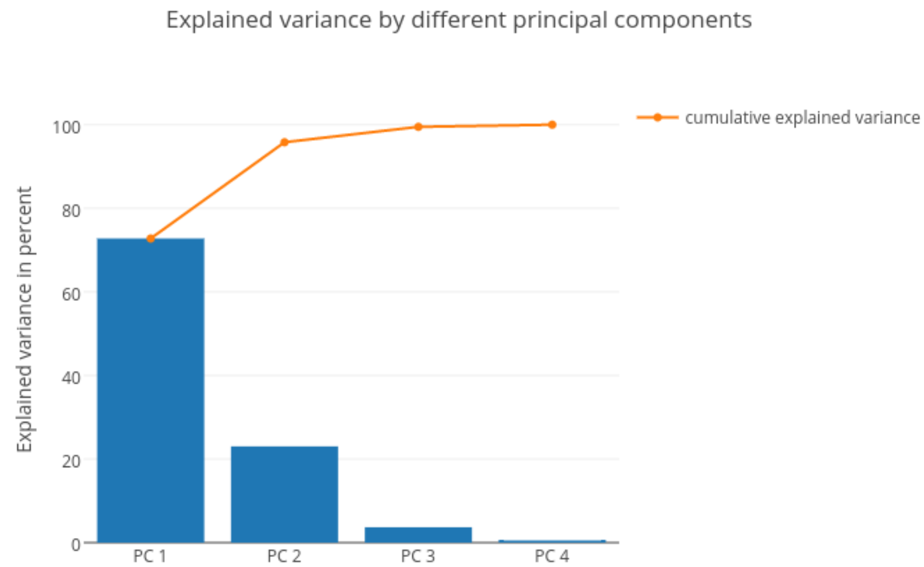
- In practice, often use SVD
  - Singular Value Decomposition
  - For stability reasons



How many principal components?

- Each eigenvalue λ tells you how much variation in the data its eigenvector v is responsible for
- $Variation_i = \frac{\lambda_i}{\sum_{j=1}^{m} \lambda_j}$
- Project data into a lower dim subspace:
  - K-dim: k < m
  - Can compute how much variation we keep
- Projection = features x data point

eigenvectors



Example

Data =

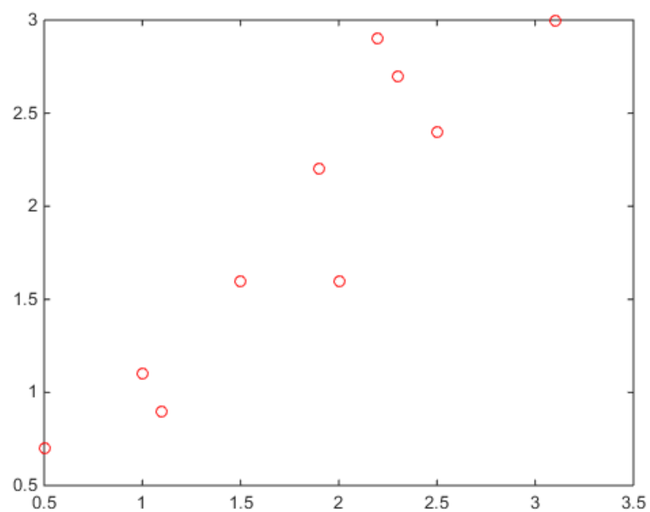
x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

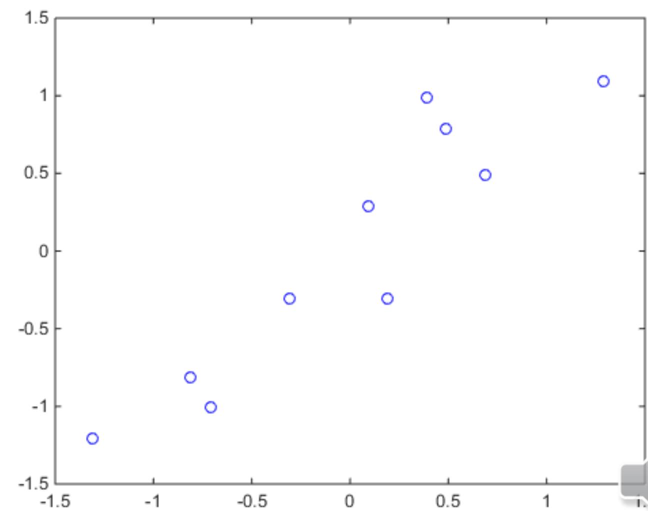
Mean: 1.81 1.91

Subtract means

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01





Example

	x	y
DataAdjust =	.69	.49
	-1.31	-1.21
	.39	.99
	.09	.29
	1.29	1.09
	.49	.79
	.19	-.31
	-.81	-.81
	-.31	-.31
	-.71	-1.01

Calculate covariance matrix:

$$C = \frac{1}{n-1} D^T D$$

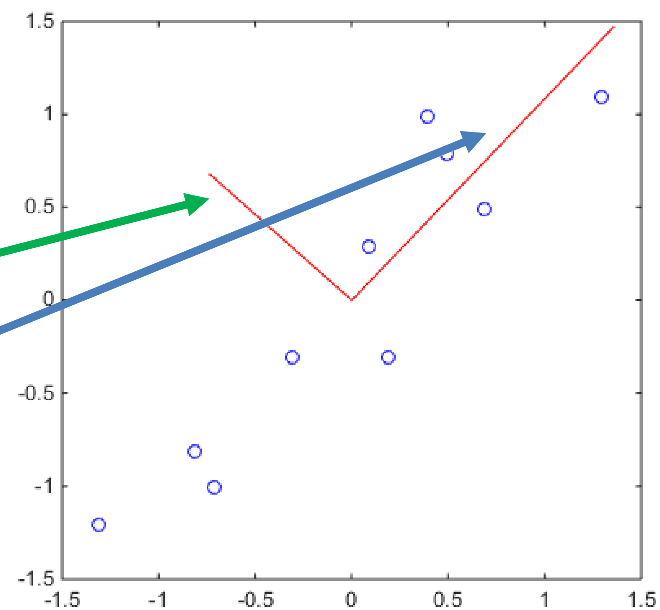
$$C = \begin{bmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{bmatrix}$$

Note: cov matrix is square, symmetric, with dimensions = number of features

Compute eigenvalues/vectors:

$$\lambda = \begin{pmatrix} 0.0491 \\ 1.2840 \end{pmatrix}$$

$$v = \boxed{\begin{bmatrix} -0.7352 \\ 0.6779 \end{bmatrix}} \boxed{\begin{bmatrix} 0.6779 \\ 0.7352 \end{bmatrix}}$$



Example

Eigenvalues/vectors:

$$\lambda = \begin{pmatrix} 0.0491 \\ 1.2840 \end{pmatrix}$$

$$v = \begin{bmatrix} -0.7352 & 0.6779 \\ 0.6779 & 0.7352 \end{bmatrix}$$

Order by decreasing eigenvalues:

$$\lambda_1 = 1.2840$$
$$v_1 = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$$

$$\lambda_2 = 0.0491$$
$$v_2 = \begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix}$$

Proportion of variation in data:

$$\frac{1.2840}{1.2840 + 0.0491} = 0.9632$$

$$\frac{0.0491}{1.2840 + 0.0491} = 0.0368$$

Example

$$\lambda_1 = 1.2840$$
$$v_1 = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$$

$$\lambda_2 = 0.0491$$
$$v_2 = \begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix}$$

Project data:

Space spanned by v_1 and v_2:

$$Features = \begin{bmatrix} 0.6779 & -0.7352 \\ 0.7352 & 0.6779 \end{bmatrix}$$

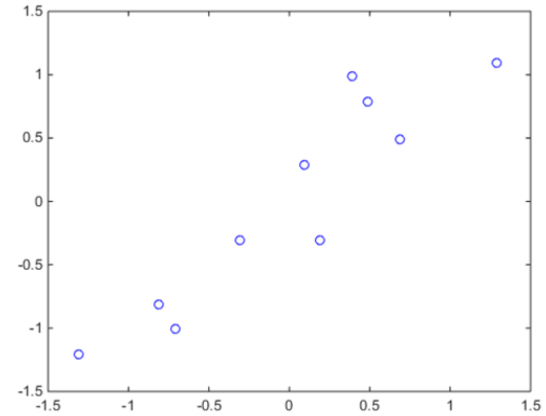
Or space spanned by v_1: (only lose 3.68% of variation)

$$Features = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$$

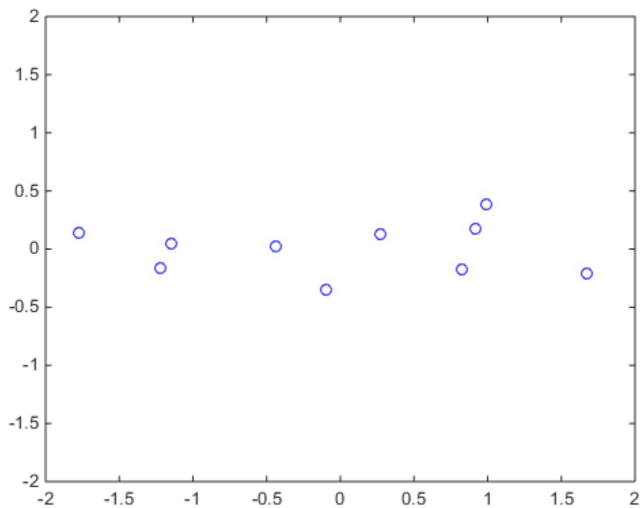
$$ProjectedData = Features \times Data$$

Example

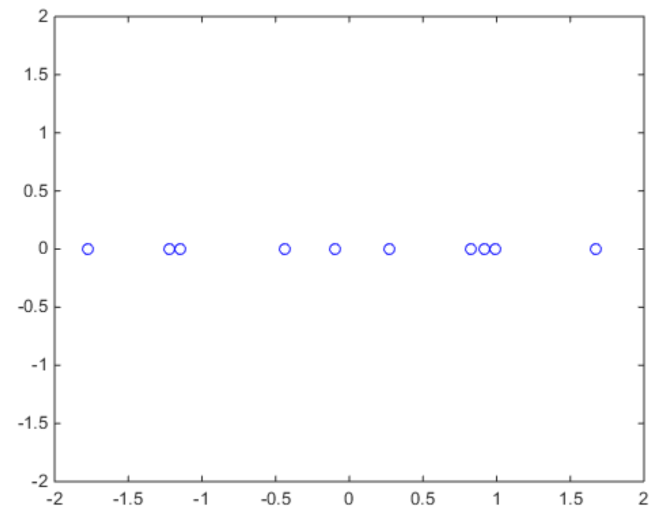
Original normalised data:



Projected to 2 PCs:



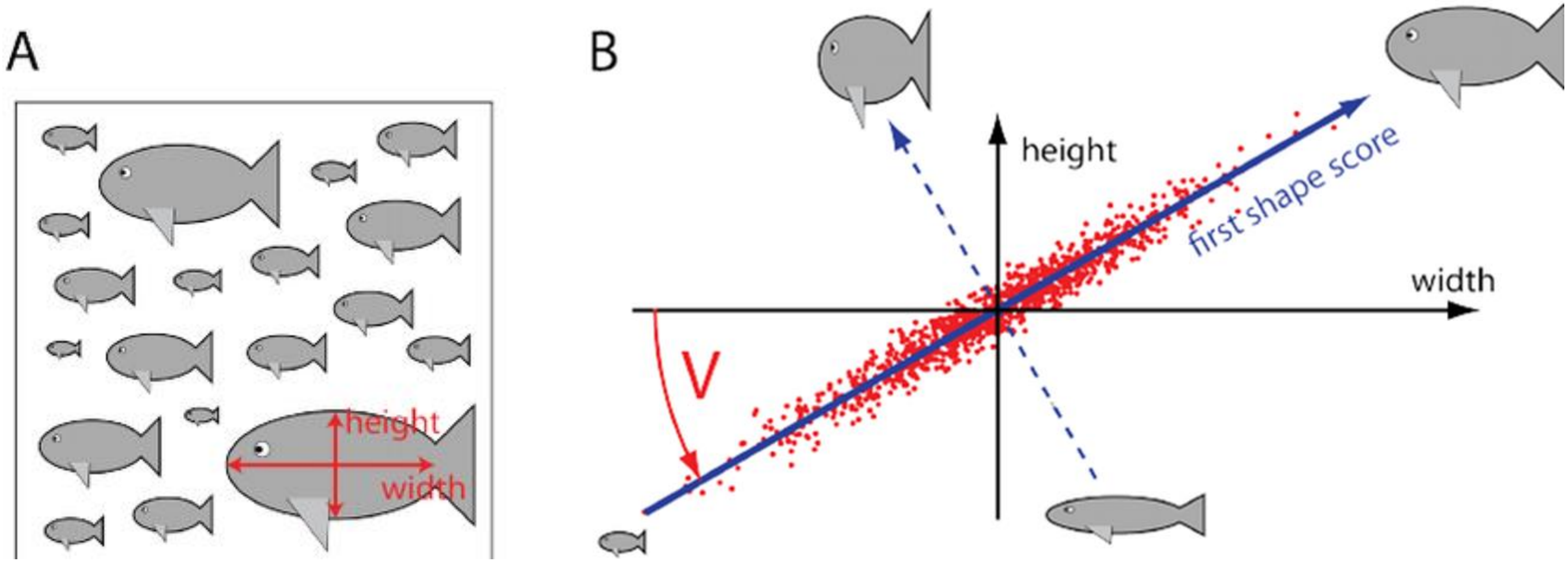
Projected to 1 PC:



Use PCA to remove additional dimensions, or to visualise high dim data

Interpreting principal components

- Additionally, each PC typically has some “meaning”
  - It encodes a direction of variation in the data



Understanding our problem

- Once we've understood the data, focus on the task
- What task are we trying to solve?
  - What do we want to predict or understand from the data?
  - How easily could a human make this prediction?
  - Do we have enough information to reasonably be able to solve the task?
    - Features?
    - Data points?

Selecting a model

- What model and algorithm to use?
  - Do we have a lot of knowledge about the problem?
  - Do we need probabilistic outputs?
  - Do we need the results to be interpretable?
  - How much data do we have? How many features?
  - Are there missing features?
  - Are features categorical?
  - Do you need a generative model?
  - How much compute time and storage space?
- Usually start with simpler models
  - Try understand their limitations
- What methods have people used on similar data before?

Unbalanced classes

- Sometimes **most** of your data comes from one class, and only a few points from another
- Examples:
  - Finance: normal vs fraudulent (anomalous) behavior
  - Medicine: good health or minor illness vs rare diseases

- Models may just predict everything is normal
  - Bias towards dominant classes

- What to do?
  - Collect more data on minority class? Generate more data? Remove some data from majority class?
  - Change cost function to make missing minority class more costly

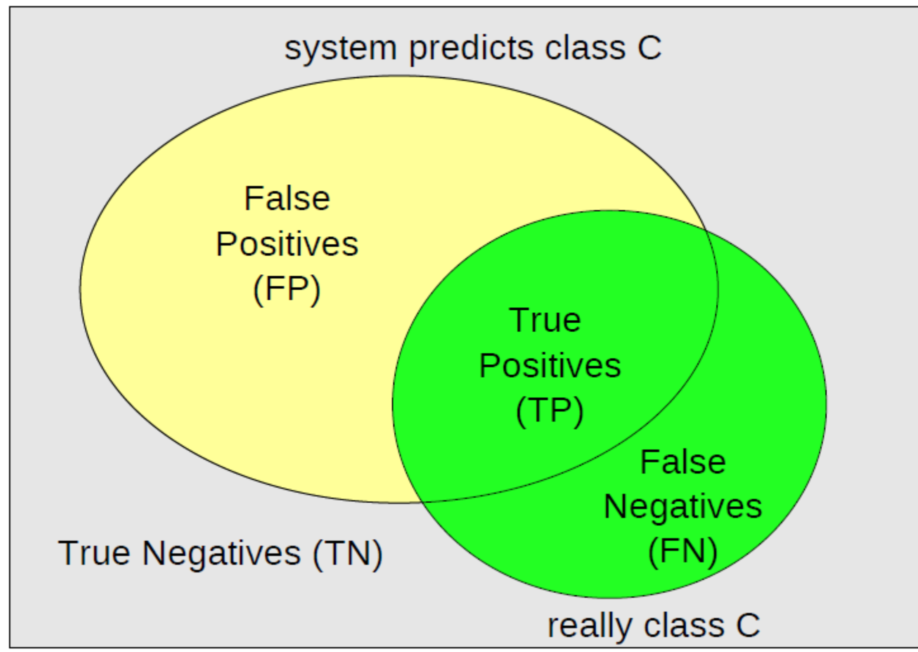
Do your results actually make sense?

- Look at the model you've just fit – is it reasonable?
  - Plot it if you can
  - Probabilities adding to 1?
  - Confirm with predictions against test data (actually look at them)
- Train models repeatedly:
  - Are results consistent?
  - Report averages
- Don't forget to split training, testing, and validation data!
  - Training data: learn model parameters
  - Validation data: learn hyperparameters
  - Testing data: evaluate model
- Unsurprising for models to perform well on training data
- **Think critically about what you're doing!**

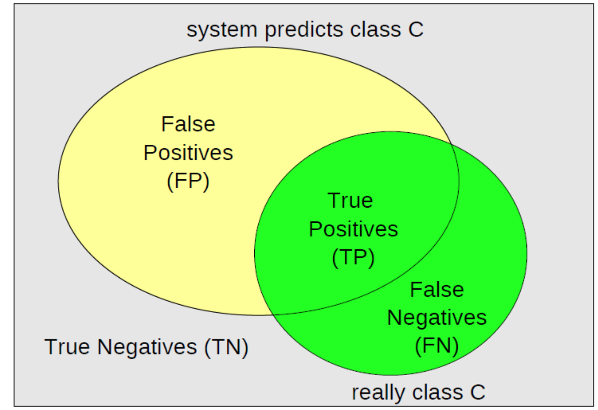
Evaluation metrics

- In classification, common metrics come from the confusion matrix

Really C? \ Predict C?	Yes	No
Yes	TP	FN
No	FP	TN

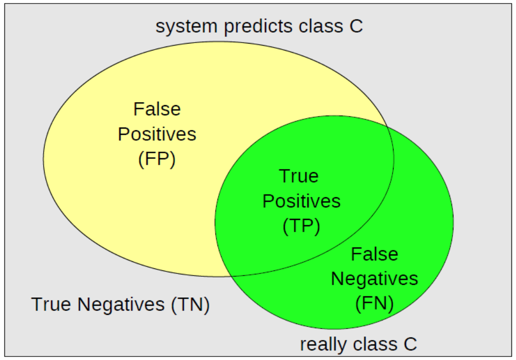


Evaluation metrics



- Derive other metrics from these:
- Classification error: $\frac{errors}{total} = \frac{FP+FN}{TP+FN+FP+TN}$
- Accuracy = (1 – error) = $\frac{correct}{total} = \frac{TP+TN}{TP+FN+FP+TN}$
- Overall measure of system quality
- But: can’t handle unbalanced classes
  - E.g. always predicting “normal” for a rare event has a high accuracy

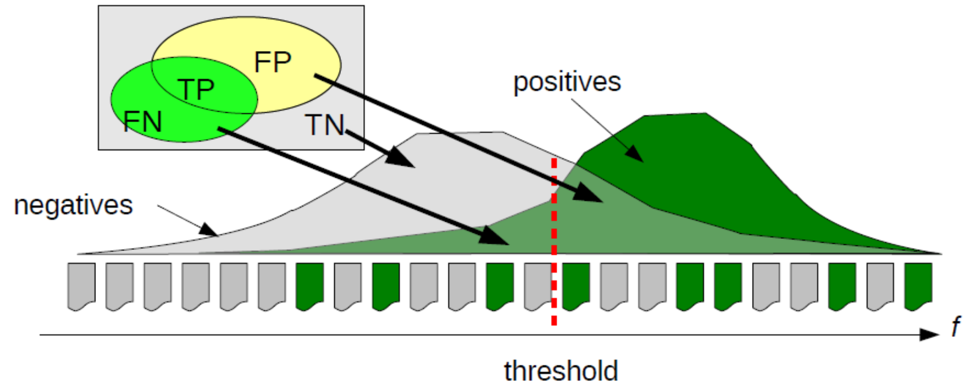
Evaluation metrics



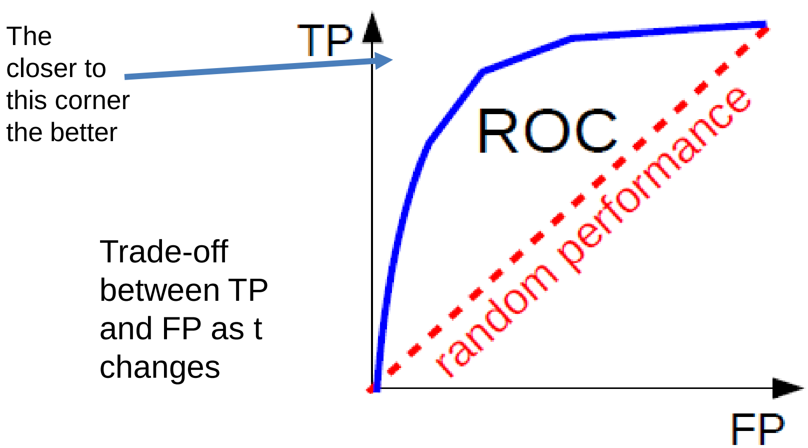
- False alarm = false positive rate = $\frac{FP}{FP+TN}$
  - % of negatives we mis-classified as positive
- Miss = false negative rate = $\frac{FN}{TP+FN}$
  - % of positives we mis-classified as negative
- Recall = true positive rate = $\frac{TP}{TP+FN}$
  - % of positives we classified correctly (1 – miss rate)
- Precision = $\frac{TP}{TP+FP}$
  - % positives out of what we thought was positive
- Meaningless to report just one of these
  - Easy to get 0% false alarm or 100% true positive
  - Typical: precision/recall, miss/false alarm, TP/FP rate

ROC curves

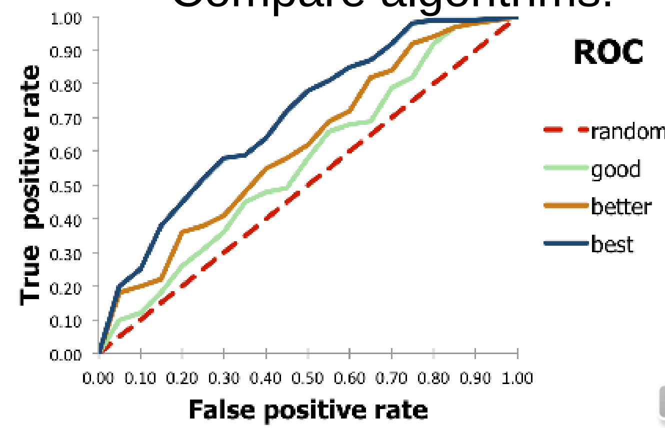
- Receiver Operating Characteristic curve
- Many algorithms compute a “confidence” f
    - Threshold t to classify:
        - Positive class if f>t
        - Negative class if f<t
    - Not always explicit



- ROC curve:
    - Plot TP vs FP as t varies



Compare algorithms:



Recap

- The data science process
- Collecting data
- Pre-processing data
- Exploratory data analysis
- Visualising data
- Principal components analysis
- Model selection
- Unbalanced classes
- Evaluation metrics