# ML Overview for Statisticians

Jack Baker, STOR-i, Lancaster University

# Introduction

- Pretty much exactly the same as stats
- But more interested in automation and minimising prediction error
- A lot of methods simple enough to 'high-level learn' over a weekend for an interview
- Will try to give brief overview of common methods that might come up in interview. Then maybe more interesting stuff.

# General Stuff

- Scikit-learn: massive toolkit of methods in Python.
- Supervised learning: has response e.g. regression, classification.
- Unsupervised learning: unstructured learning e.g. clustering, dimension reduction.

# Tuning & Model Selection

- Lots of it – price to pay for nonparametric?
- Almost always uses cross-validation on held out test set.
- Scoring supervised learning: use score on response on test e.g. residuals, accuracy.
- Scoring unsupervised learning: use log predictive or log likelihood on test. Also often used for supervised.
- Tuning often just uses 'grid-search', library for this in sklearn.

# Regularisation

- Important when high-dims or have inconsistent system.
- Ridge: $\min l(\theta) + \lambda|\theta|^2$ (Normal prior). Just useful to prevent overfitting.
- Lasso: $\min l(\theta) + \lambda|\theta|$ (Laplace prior). Shrinks some params to 0. Automatic 'feature selection'. Less stable than Ridge.

# Supervised Learning (regression or classification)

- Random forests: decision trees with less bias.
- Neural Networks: stacked GLMs. 'Universal function approximator'
- Support Vector Machines: maximize distance between 'support points'
- Gaussian Processes

# Logistic Regression (just adv and disadv)

- Logistic space has to be linear.
- Interpretable, good uncertainty quantification.
- If doing big data style then tuning becomes harder than random forests.

# Random Forests

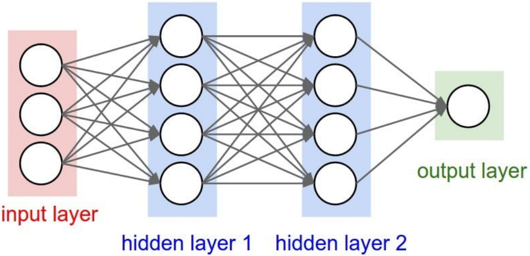
- Decision trees known to be biased.
- So random forests fits loads of decision trees.
- Reduces bias by subsampling data as well as EXPLANATORY VARIABLES. Then take average/'voting' of these.
- Very little tuning, very scalable, nonparametric so can fit to complex spaces.
- But better with lots of data, bad at uncertainty and interpretation.

# Neural Networks

- Similar to stacked logistic regression.
- But can use generic 'squashing function' not just logit.
- Basically just tries to nonparametrically approximate any function.
- $h(.)$ squashing fn then e.g.

$$\mathbf{y} = h(W_2 h(W_1 \mathbf{X} + \beta_1) + \beta_2)$$

- But more complex architectures exist.

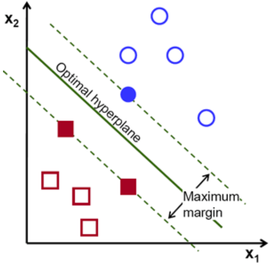


# Neural Networks

- Can give excellent prediction accuracy.
- Automates structure in lots of different areas: vision, text, etc. Nonparametric: Highly nonlinear spaces.
- Needs lots of data, tuning a pain (also choosing architecture), more for BIG PROJECTS.
- Lots of automated packages: Tensorflow, Pytorch, etc.

# SVMs

- Maximize distance between closest points to hyperplane
- To get around linear separable problem: map points to higher dimensional space.
- Uses theory of RKHS so actual higher dim points don't need to be calculated.



# SVMs

- Not as scalable as RFs and NNs.

- Can deal with high dims, complex spaces.

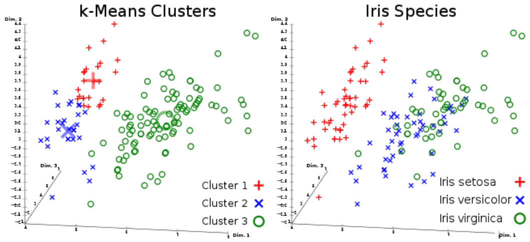
- Annoying to tune, non-probabilistic, binary.

# Unsupervised: Clustering

- K-means: Easy and scalable not very flexible.
- Spectral: Not as scalable, Excellent with complex shapes
- DB Scan: Scalable, good with complex shapes
- Mixtures: Probabilistic, no complex shapes, not scalable(??)

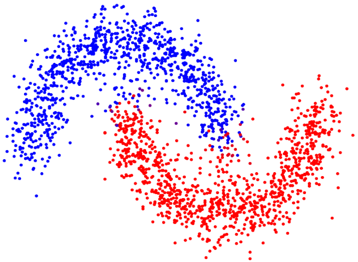
# K-Means

- Minimize distance between within cluster sum of squared.
- Produces round clusters of similar size.
- Scalable
- Often performance is poor



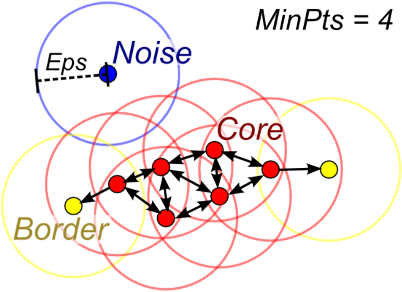
# Spectral

- Construct similarity matrix between points (e.g. euclid distance).
- Applies spectral decomposition – repeatedly finds evalues & evectors to split.
- Good with nonstandard data.
- Great with complex shapes. Okay scalable.



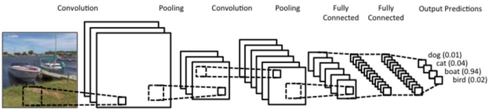
# DBScan

- If $m$ points in $\epsilon$-ball path, these are a core
- All points within $\epsilon$-ball 'path' to core in cluster.
- Robust, dont need K.
- Tuning, Good with nonstd shapes, scalable.



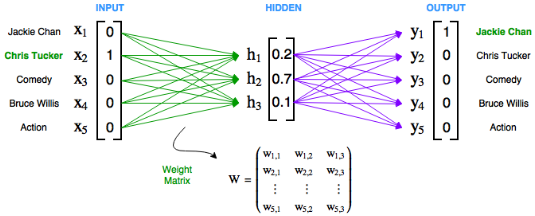
# Image Data

- Convolutional Neural Nets become the standard.
- Automatically learns features e.g. edges, blurs etc.
- See e.g. MNIST tutorial on TensorFlow.



# Text Data – Formats

- Bag of Words format [on board].
- Word Embeddings: Words mapped to high-dim vector space using Neural Networks based on words around it (idea is to capture semantic meaning).
- Idea is that semantically similar words will be close in the space.
- Loads of previously learnt embeddings you can download so can just use off the shelf.



# Text Data – LDA

- Clusters a set of documents.
- Input: Bag of Words.
- Learns different clusters of documents, meant to be related to topics.
- Associates each word with probability of being in topic k.
- Associates each document with probability of being in topic k.
- Combines to cluster.

# Recommender Systems: Probabilistic Matrix Factorisation

- Dimension reduction: factorise matrix of customers–products to matrices of customers–D and products–D.

- Lots of missing info – probabilistic.

- Netflix prize, amazon recommendations.