WS06: Applied Statistical Learning in Python

1st Big Data Machine Learning in Healthcare in Japan@TMDU:

TMDU-JSICM-NUS-ANZICS-MIT Critical Data Workshops and Datathon 2018

Calvin J Chiew & Naoaki Ichihara

Sat, 24 Feb 2018 (3.30PM - 5.30PM)

https://github.com/calvinjchiew/tokyo18

Important Concepts for Today

1. **Model fit**
2. **Random forest**
3. **Support vector machine**
4. **Cross-validation**

- Summarized in handout

Popular ML Methods

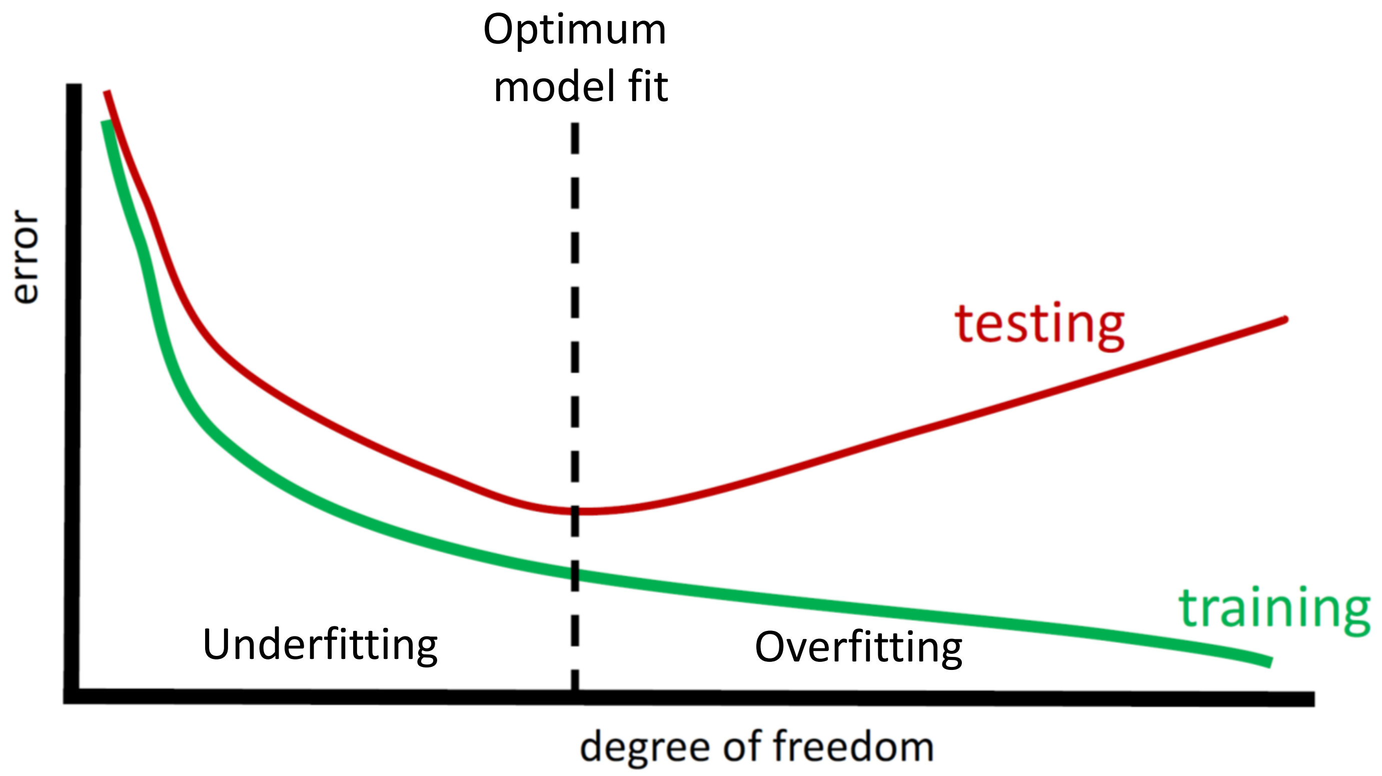
- **Supervised Learning**
  - K-nearest neighbours
  - Regression (linear, logistic, polynomial, spline etc.) ± regularization
  - Linear/quadratic discriminant analysis
  - Tree-based approaches: decision tree, random forest, bagging, boosting
  - Support vector machine
  - Neural network
- **Unsupervised Learning**
  - Principal components analysis
  - Clustering
  - Neural network

Model Fitting

- We want to estimate f where

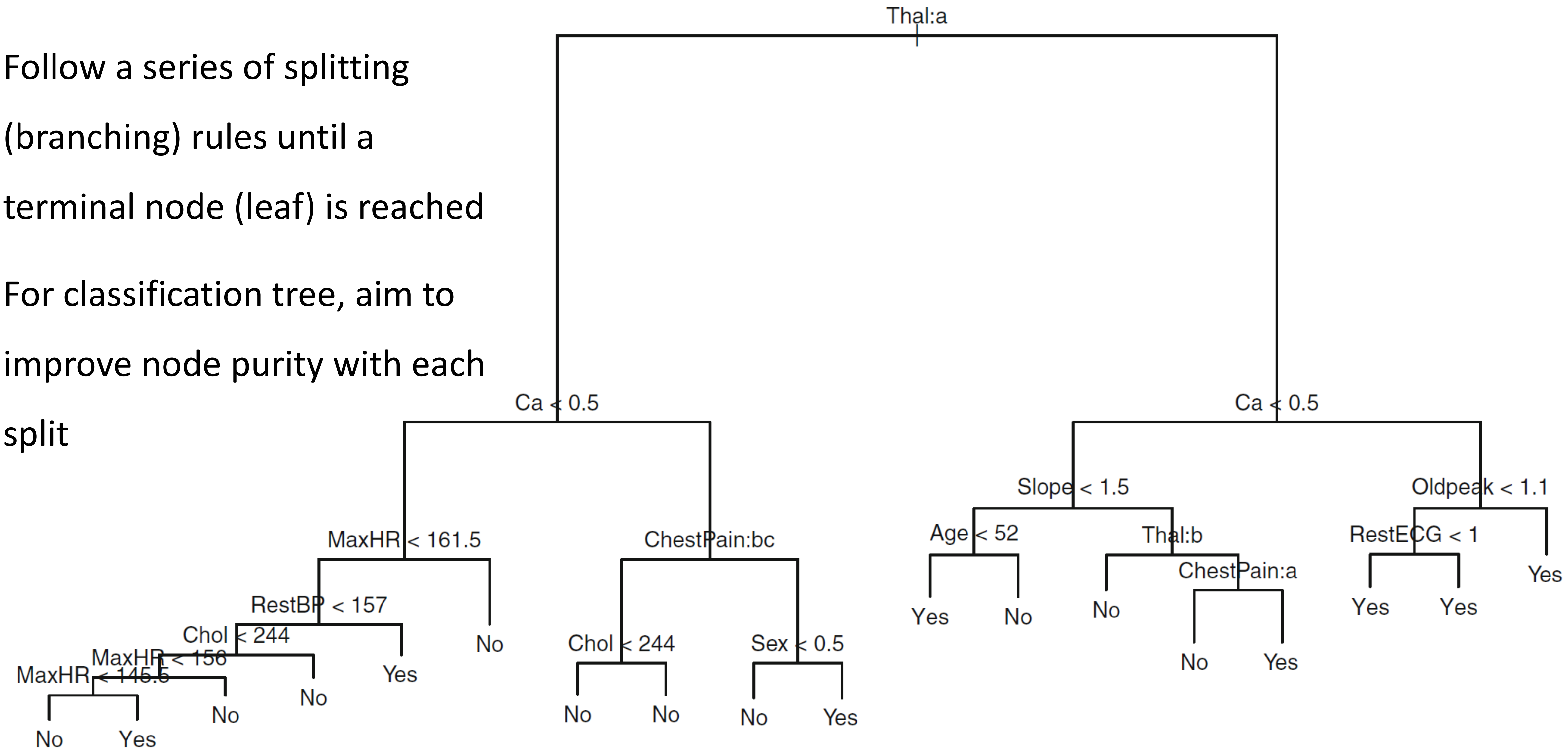
$$Y = f(X_1, X_2, X_3 \ldots) + \varepsilon$$

  - X: feature, predictor, independent var
  - Y: outcome, response, dependent var
  - ε: error
- Data is split into distinct **training** and **testing** sets to prevent **overfitting**
- Loss (error) function depends on prediction task



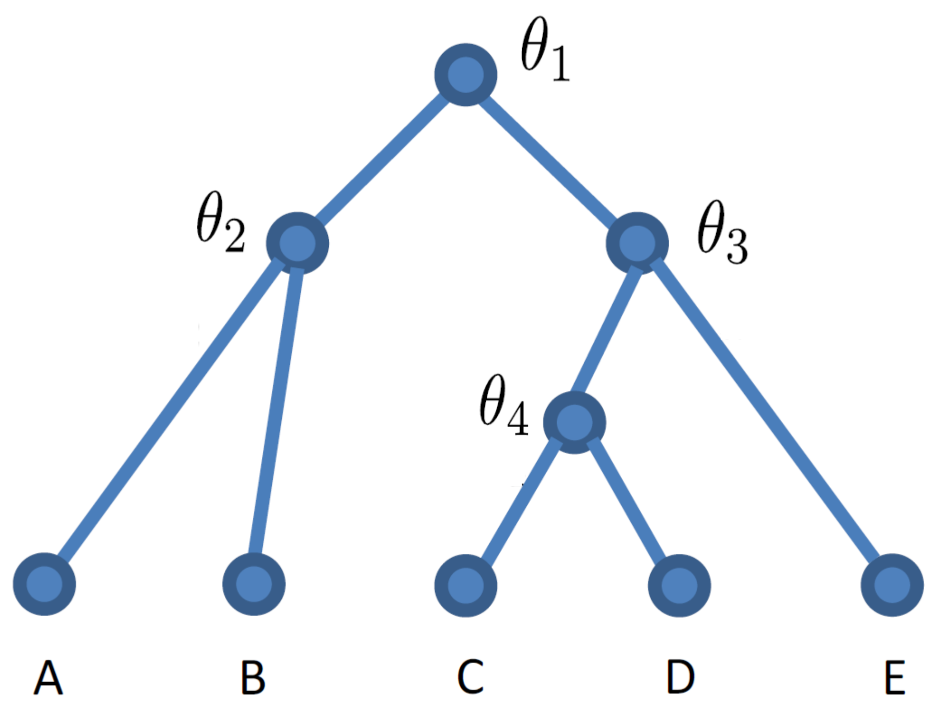
Decision Tree

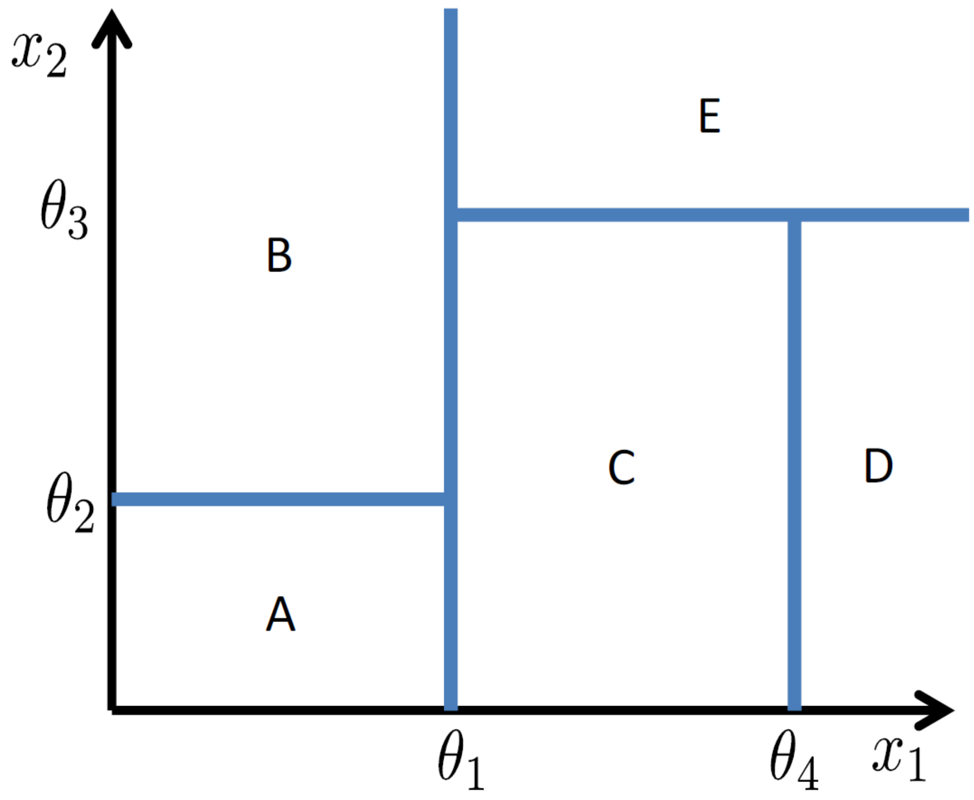
- Follow a series of splitting (branching) rules until a terminal node (leaf) is reached
- For classification tree, aim to improve node purity with each split



Decision Tree

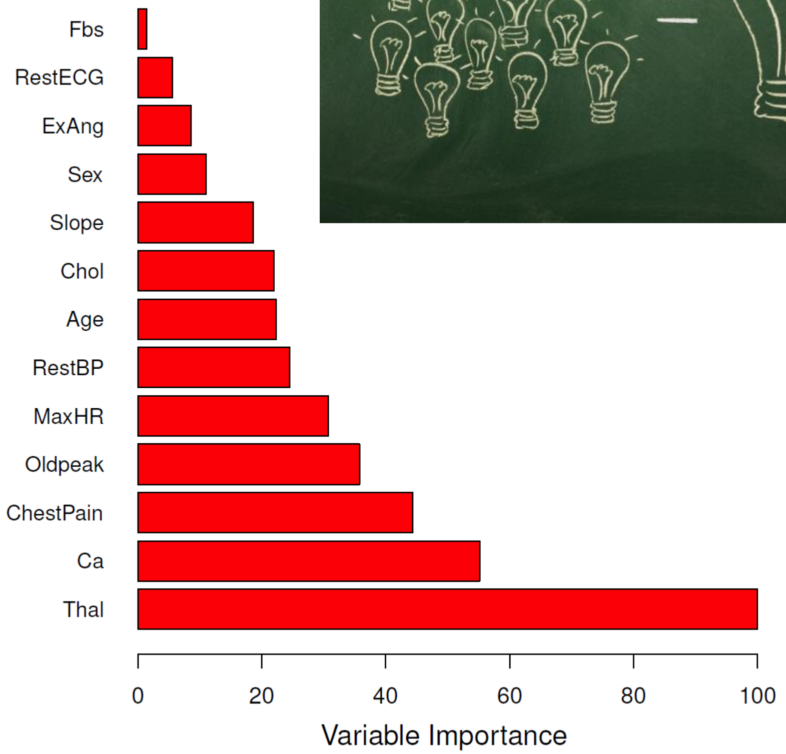
- The feature space is split into rectangular regions (boxes)
- We use the mean or majority class of observations in each region for prediction





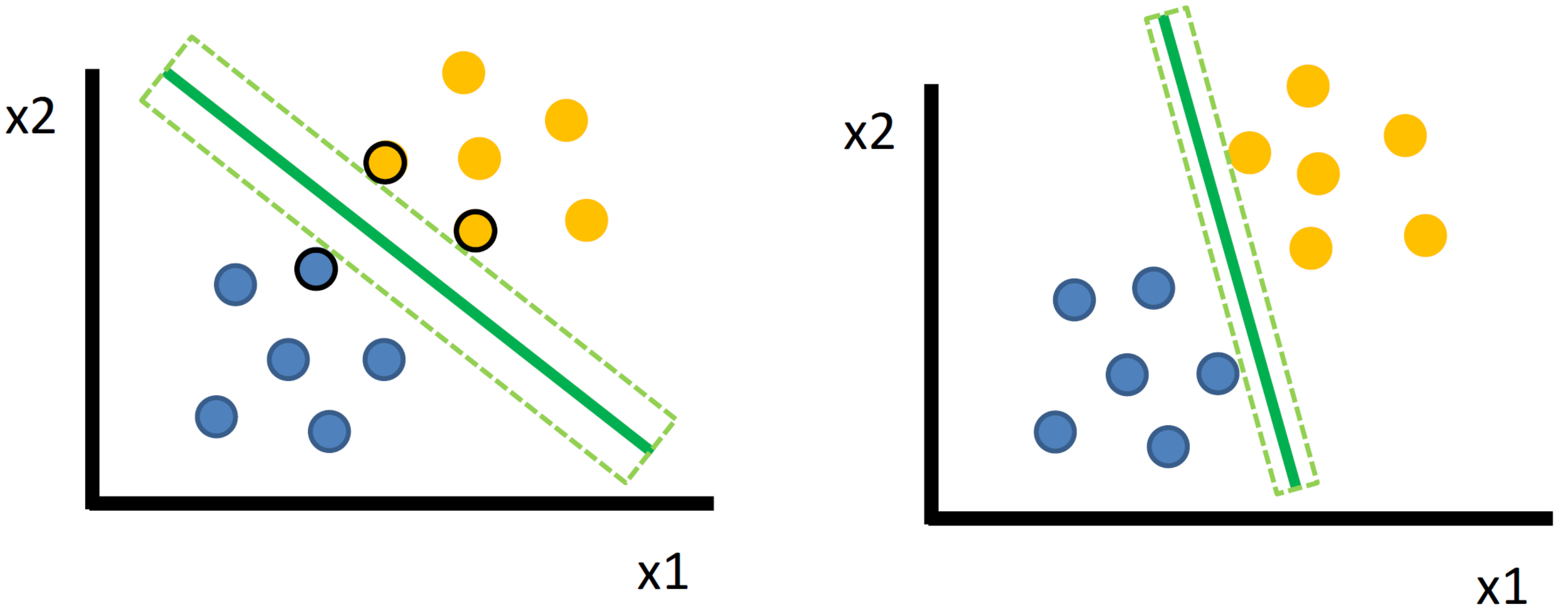
Random Forest

- Multiple trees which are combined to yield a single consensus prediction
  - Averaging multiple onerous predictions produce less uncertain results
- At each branch, only a **random subset** of all the predictors are considered as potential split candidates
  - To obtain trees that are less similar to each other
- Feature importance can be visualized by total decrease in Gini index due to splits over the feature



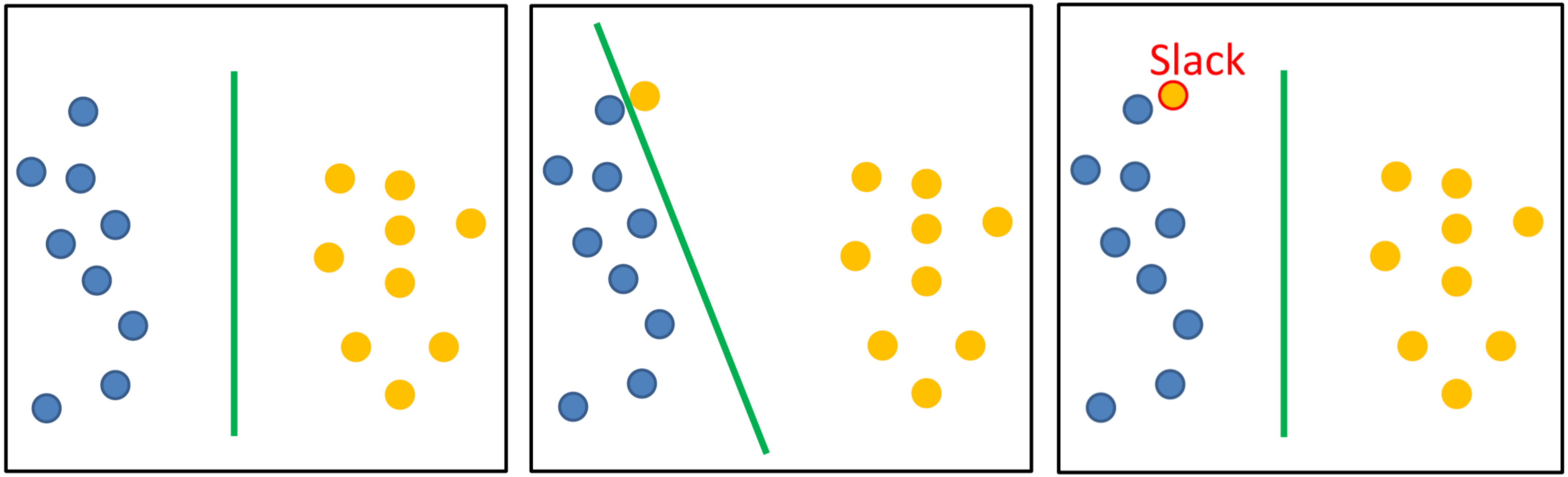
Support Vector Machine

- The wider the **margin**, the more confident we are in the separating hyperplane
- Separating hyperplane depends only on the **support vectors**



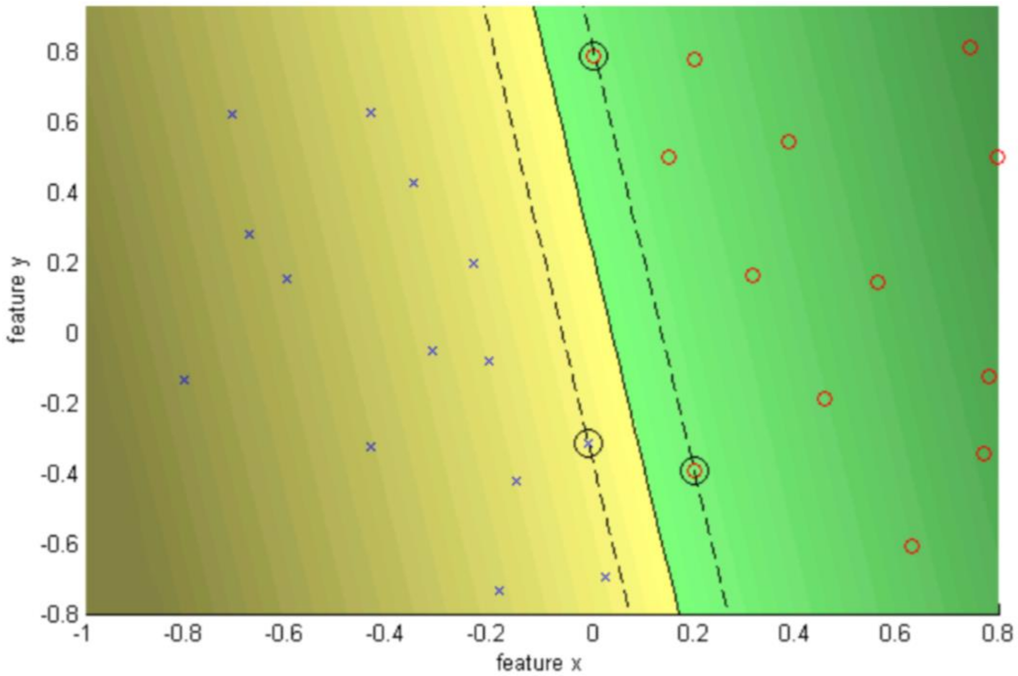
Support Vector Machine

- We allow some **slack** for data points to be on the “wrong” side of the hyperplane in exchange for a more robust hyperplane (against outliers)

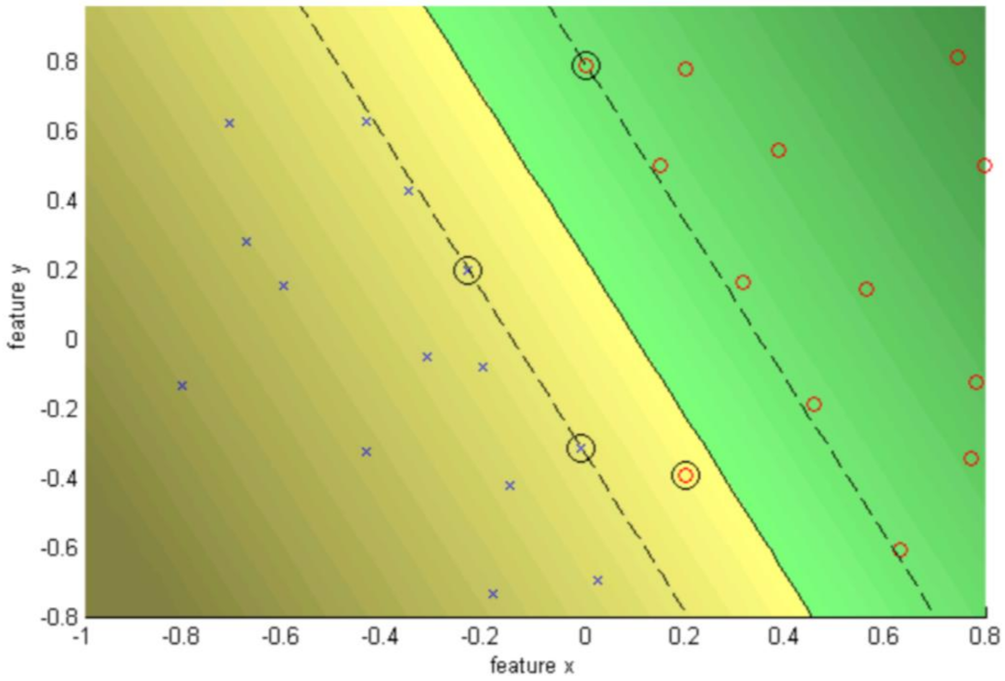


SVM Regularization

- When **C** is small, more slack is allowed, resulting in a softer (but wider) margin



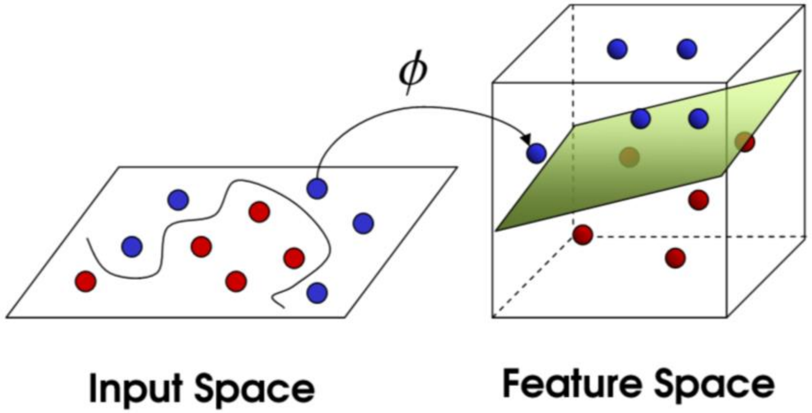
Hard margin
(Large C)

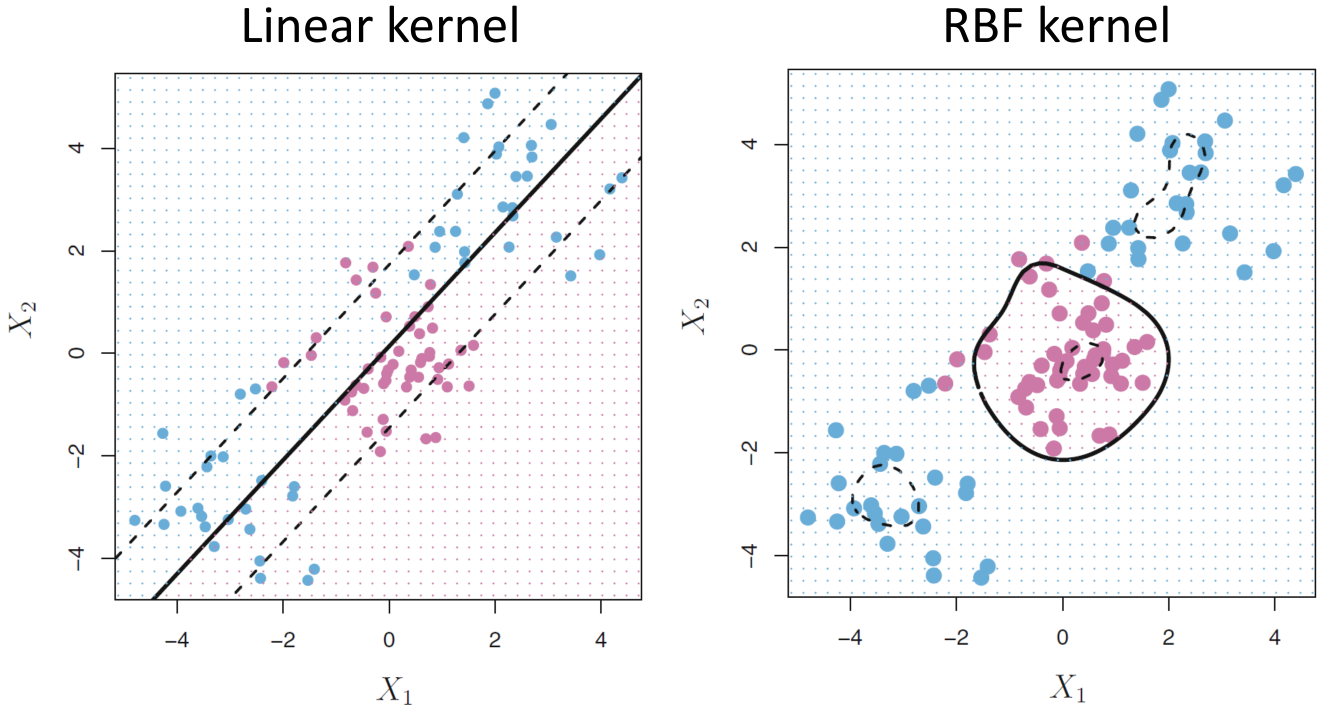


Soft margin
(Small C)

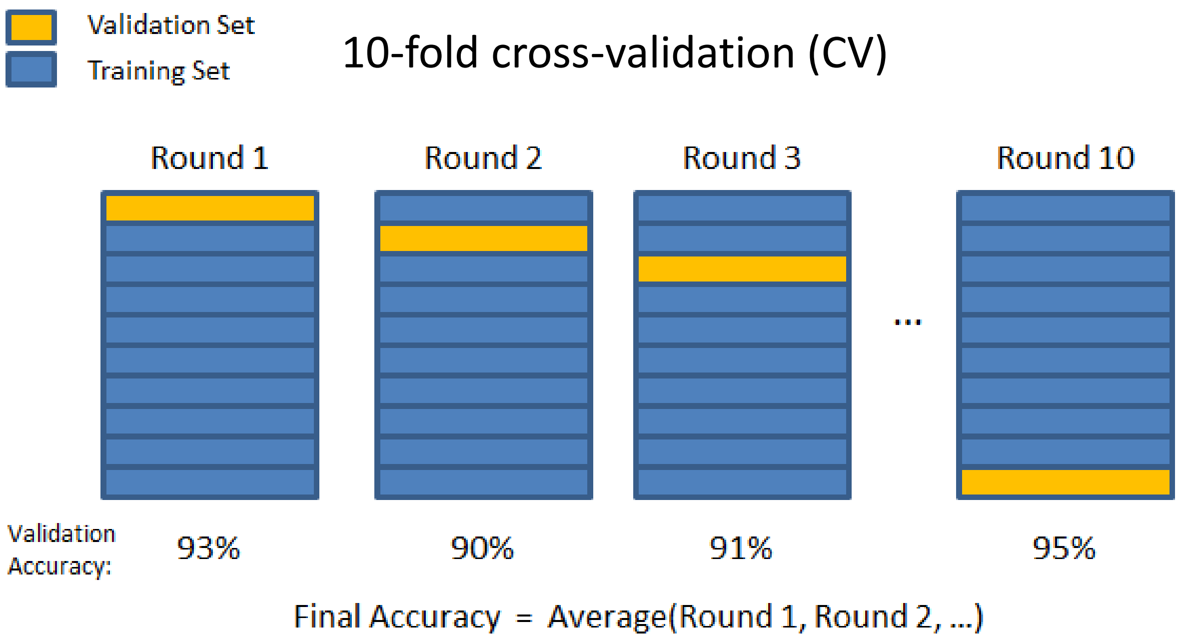
SVM Kernel

- Non-linear kernels allow us to project data points not linearly separable (on the input space) onto a higher-dimensional (feature) space where a linear separating hyperplane can be drawn
- This projection is done by a **kernel function** eg. radial basis function (RBF)
- When projected back onto the input space, the decision boundary is non-linear





k-fold Cross-validation

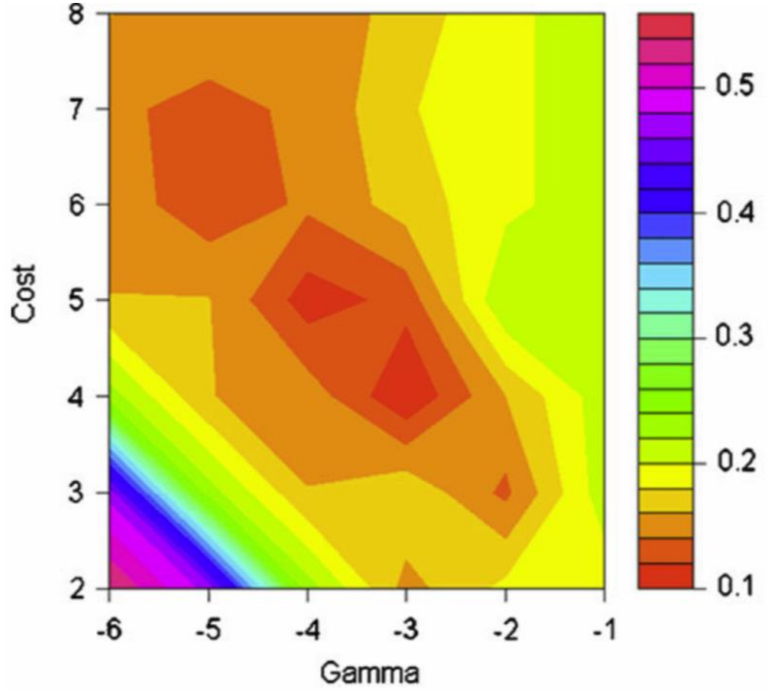


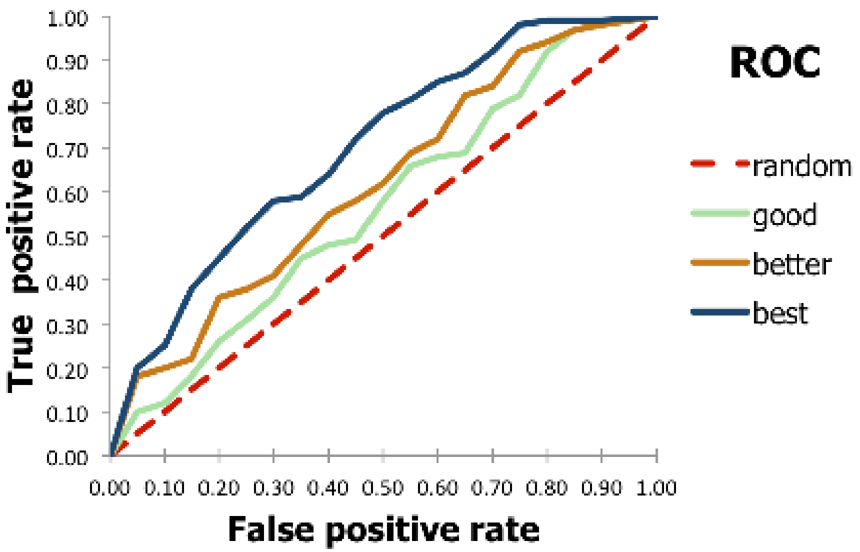
Examples of parameters to optimize:

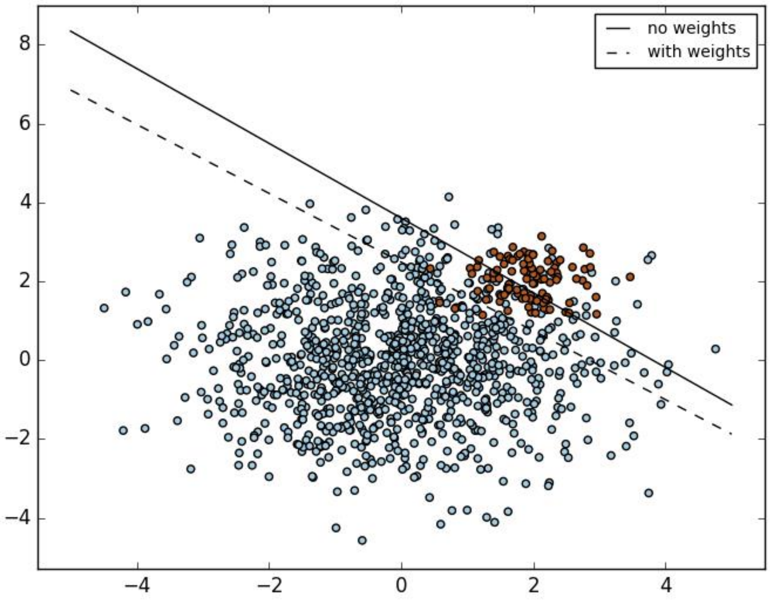
- Random forest
  - Number of trees, B
  - Number of predictors considered at each split, m
  - Max tree depth / min node size
- Support vector machine
  - Penalty / amount of slack tolerated, C
  - Kernel coefficient, γ
- Boosting
  - Learning rate, λ
  - Number of splits in each tree, d

Others

- Grid Search CV, Receiver Operating Characteristic (ROC) curve, class weighting







Thank You!

Example & Hands-on Exercise in Python:

https://github.com/calvinjchiew/tokyo18