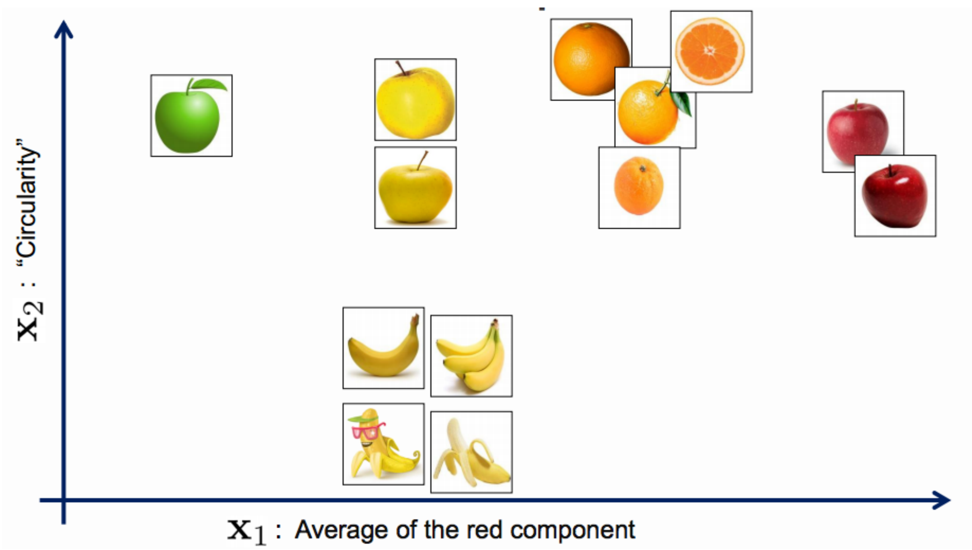
Machine Learning

Feature Extraction

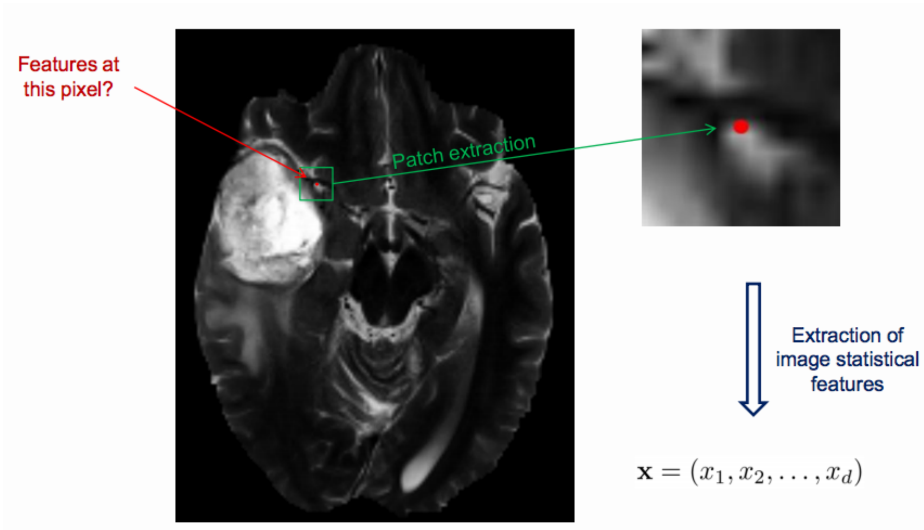
- The process encoding all the datum into a vector
- What are the qualities we expect for features?
    - carrying compact information
    - discriminative 判別
    - easy to compute
    - invariant to Noise, Changes of scale, Transformations



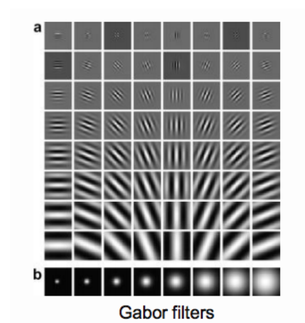
    - Choice of the axes = **feature extraction**
    - Identification of these areas = Goal of learning method
    - Training linear classifiers → SVM, random forest…

- What is an image to a computer?
    - Pixel = **3 integer values (R, G, B)** / **1 integer (gray level)**
    - image = matrix of size **n * P * 3 / n * p * 1** with integer entries (0~255)

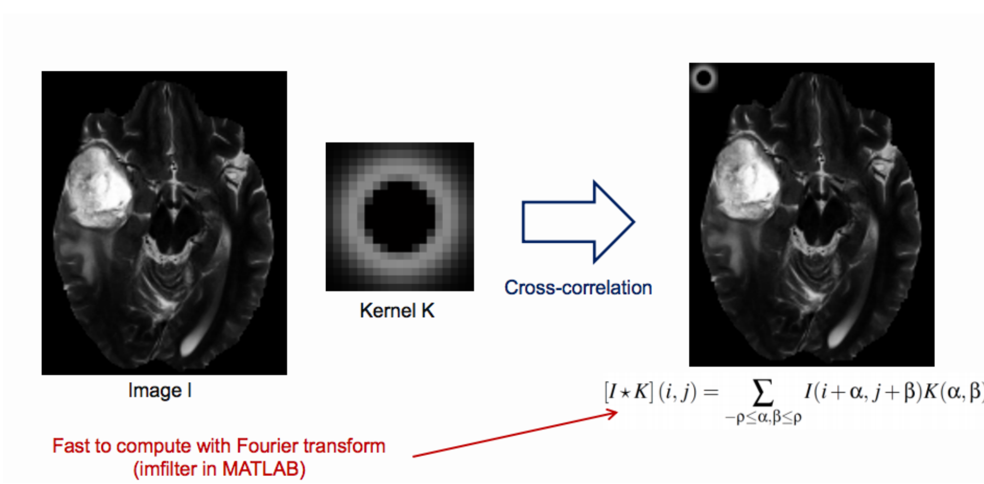
- Local the **contextual features**



   - **Local:**
      - intensity
      - Derivatives : Horizontal, Vertical, Diagonal
      - Edge orientation
   - **Contextual (over a patch):**
      - Mean intensity over the patch
      - Variance / standard deviation
      - Intensity distribution : MAx, Min, Median, Range
      - Statistics : Energy, Skewness, Kurtosis, Entropy
   - **Bank of filters**
      - Classical kernels : Gaussian, GoL…
      - Gabor filters
         - Gaussian kernel manipulated by a sinusoidal正弦 plane wave
         - different scale and orientation
         - each response is a feature : the feature space → **Gabor space**

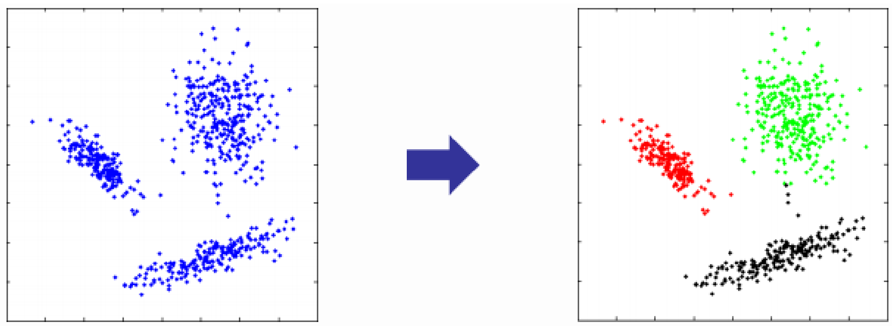


   - **Computation at every pixel!**



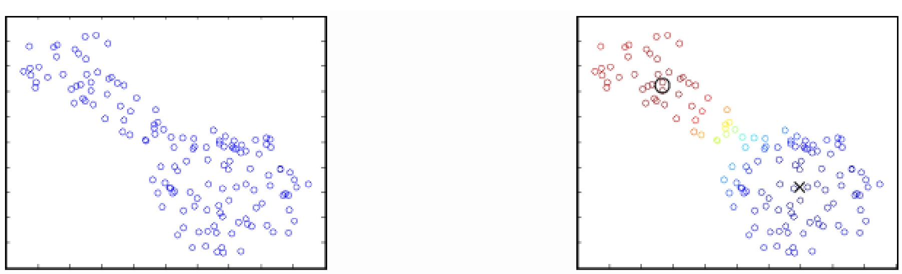
Clustering - K-mean, K-medoids, EM alg.

- **K-mean clustering**
  - Given n unlabelled **data points {x1,.....,xn}** & the number of **group K**
  - Randomly choose K data points → cluster centroids
  - Repeat following steps:
    - Assign every data point to the closest centroid (Euclidean distance)
    - Update the centroid position of every cluster based the points assigned to it
    - While the alg. has not converge (eg. the cluster centroid barely change!)
  - Use different Initialization to run it Again!



Original data　　　　Clustering using K-means

- **Fuzzy C-Means**
  - Soft assignment is needed
  - Randomly initialise the cluster centroids & softness parameter m (m>1)
  - Repeat:
    - compute the membership to every cluster → cluster k $$w_k(x) = \frac{1}{\sum_i^K \left(\frac{d(c_k,x)}{d(c_i,x)}\right)^{2/(m-1)}}$$
    - update the centroid of every cluster $$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}$$
    - while it has not converged
  - use different initialization to run it again!

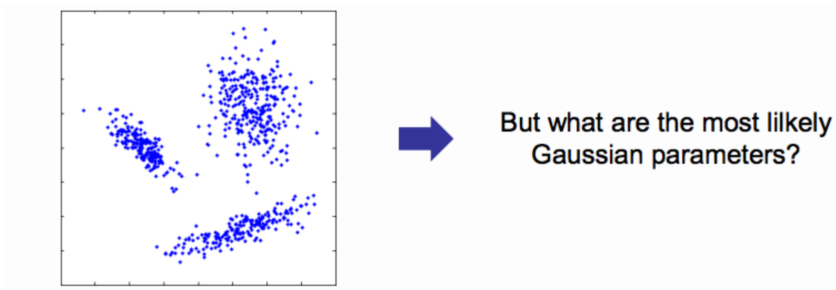


Data　　　　Fuzzy C-Means Result

- **Expectation-Maximization (EM) Algorithm**
    - in all previous method, 2 steps implicitly/explicitly alternate

  → **Minimise** over the membership assignment, by finding the closest cluster centroid
        - Expectation of the memberships given the centroids (**model parameters**)

  → **Minimise** over the choice of the centroids, by updating the centroid positions
        - Maximisation of the likelihood by adjusting the centroids (**model parameters**)
    - MAximum likelihood for model-parameter estimation

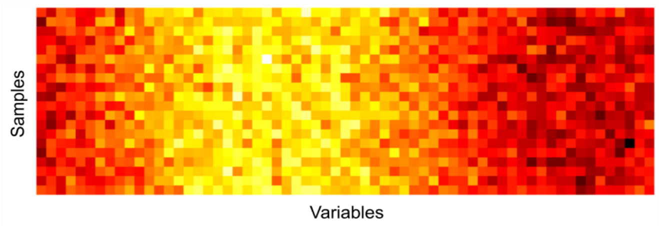


Data showing a Gaussian mixture structure

- **The EM Alg. for Gaussian Mixture Estimation**
    - Given **n data points {x1,...,xn}**, define the number of **Gaussians K**
    - Initialise the Gaussian parameter estimates $\{(w_1, \mu_1, \Sigma_1), \ldots, (w_K, \mu_K, \Sigma_K)\}$
    - Repeat :
        - Expectation:
            - compute the membership of every pixel point i → every Gaussian j $\gamma_{ij} = \dfrac{w_j N(x_i | \mu_j, \Sigma_j)}{\sum_{l=1}^{K} w_l N(x_i | \mu_l, \Sigma_l)}$
        - Maximisation:
            - compute the Gaussian parameters with the membership

$$w_j = \frac{1}{n}\sum_i^n \gamma_{ij} \qquad \mu_j = \frac{\sum_{i=1}^n \gamma_{ij} \cdot x_i}{n \cdot w_j} \qquad \Sigma_j = \frac{\sum_{i=1}^n \gamma_{ij} \cdot (x_i - \mu_j)(x_i - \mu_j)^T}{n \cdot w_j}$$

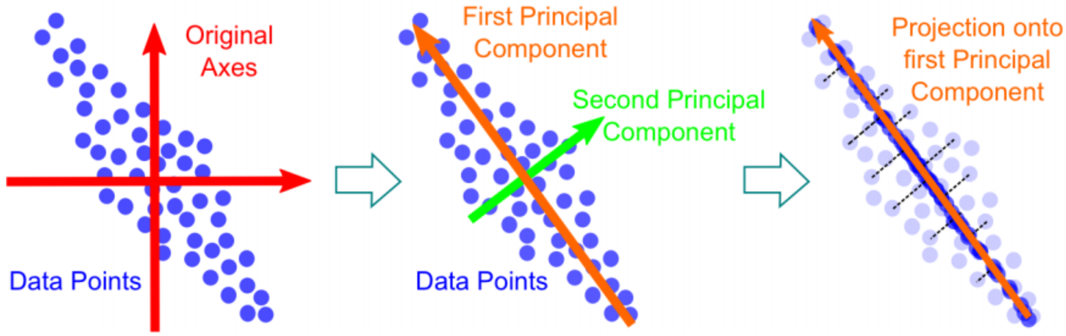
Dimension reduction - PCA



An Example of the HDLSS Problem

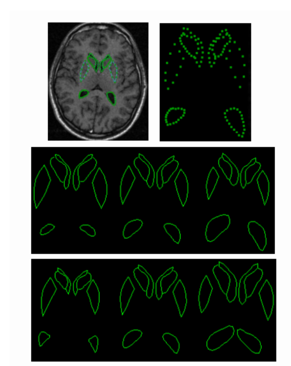
Curse of Dimensionality → the **H**igh **Di**mension **L**ow **S**ample **S**ize problem

- **Principal Component Analysis (PCA)**



  - Center the data (subtract the mean from x)
  - Diagonalize **X^T**
    - Using **SVD** → **X^T = WV^T**
    - Computing the eigenvectors of the covariance matrix **C=X^TX**
  - Form the transformation matrix **T** with the d singular vectors with highest singular values

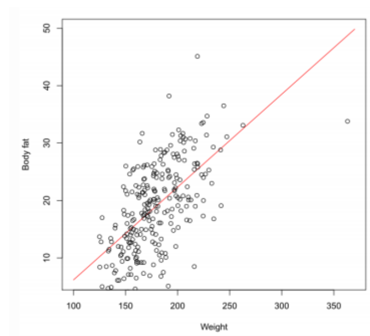
- **Medical image Segmentation**
  - Shape prior deformable model : **Active Shape Models (ASM)**
  - **Learn shape statistic** from sampled point coordinates over a training set of example shape!
    - sample the “shape” with landmark
    - establish correspondences across the training set
    - perform PCA →
      - The mean shape (N-vector)
      - the K modes of shape variation (PCA eigenvectors)
    - New shapes = only linear combination of shape variation modes



Linear regression & classification - Ordinary linear regression, Logistic regression, SVM

- Definitions & Problem statement
  - A training set {x_i, y_j} is comprise of n samples
  - Every training sample x_i consists of m features & is associated with output y_i
  - Features & output can be either cont./ discrete.
    → a **regression** problem if the output is **continuous**
    → a **classification** problem if the output **discrete**
  - Let x indicate a matrix → every row is a sample & every column is a variable/feature
  - Assumption : there is a function f(x) → relating to features→output
  - Goal : find a good f(x) to the function

Linear Model



- Define the linear model

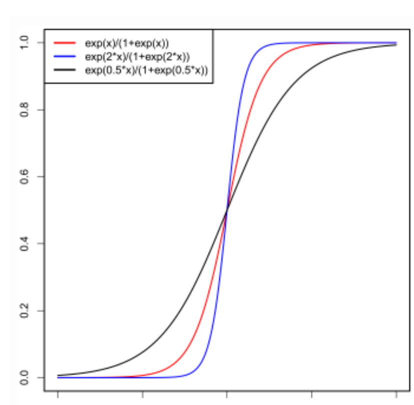
$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_m x_{im} = x_i^T \beta + \varepsilon$$

- The parameter β are **coefficients/weights** of the features and are to be estimated from the training data
- The error term εi
  → Gaussian independently identically distributed

Ordinary Linear Regression線性回歸 - Least Squares Estimation

- Choose the square error loss function $L = \left(y_i - \hat{f}(x_i)\right)^2$
- The estimation (ordinary least squares estimation) $\hat{\beta} = \left(X^T X\right)^{-1} X^T y$
- The minimum of the loss function can be computed analytically!
- x must have full column rank
- Regression is performed using : $\hat{f}(x_1, x_2, \ldots, x_m) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_m x_m$

Logistic Regression - Problem



- Consider a binary classification problem where y_i = {0,1}
- if y_i = 1 → the i-th sample belongs to the positive class, otherwise the negative class
- create a model of the probability of sample x_i belonging to the positive class πi

$$\pi_i = \frac{1}{1 + \exp^{-1}(\eta_i)}$$

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_m x_{im}$$

Log-Odds Ratio

- The model is linear with respect to the log-odds

$$\pi_i = \frac{1}{1+\exp^{-1}(\eta_i)} \Leftrightarrow \log\left(\frac{\pi_i}{1-\pi_i}\right) = \eta_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_m x_{im}$$

- Coefficient βj represents the log-odds ratio of the j-th feature
- $\beta j > 0 \Rightarrow$ Odds Increase with the j-th feature
- $\beta j < 0 \Rightarrow$ Odds decrease with the j-th feature
- useful for determin feature influences
    - when predicting diseases based on clinical features

Maximum Likelihood Estimation of Logistic Regression

- The likelihood function $L(\beta) = \prod_{i=1}^{n} P(y_i|x_i) = \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i}$
- The log-likelihood function $l(\beta) = \sum_{i=1}^{n} y_i \log(\pi_i) + (1-y_i)\log(1-\pi_i)$
- Maximum likelihood estimation (MLE) $\hat{\beta} = \arg\max_{\beta} l(\beta)$

Support Vector Machines (SVM)

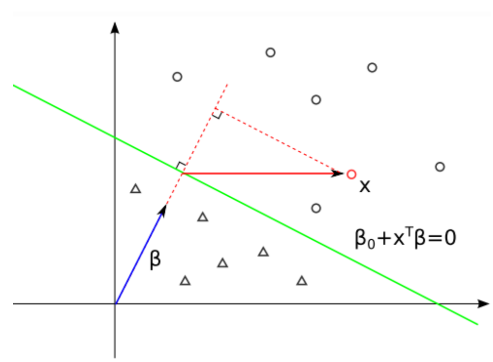
- The optional separating hyperplane problem
- Consider a binary classification problem where 2 classes are optimally separable
- A lot of hyperplane solve this problem, but which Better?
- Intuition : the margin separating both classes has to be maximize

- **Geometric Margin**
    - The linear hyperplane is given by

$$f(x) = \beta_0 + x^T\beta = 0$$

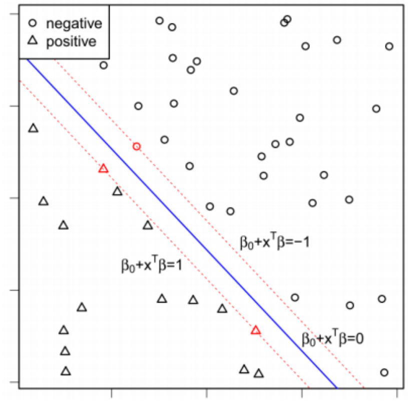
    - The assigned distance of a point x_i to the hyperplane is given by

$$\frac{\beta_0 + x_i^T\beta}{\sqrt{\beta^T\beta}} = \frac{\beta_0 + x_i^T\beta}{\|\beta\|}$$



- **Optimal Separating Hyperplane & Support Points**

  The margin is given by $\frac{1}{\|\beta\|}$

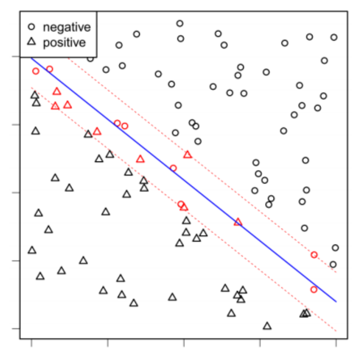
  - Maximizing the margin is equivalent to

    $\min \frac{1}{2}\|\beta\|^2$ subject to $y_i\left(\beta_0 + x_i^T\beta\right) \geq 1$,
  - The solution $\hat{\beta} = X^T\alpha$, where $\alpha \in R^n$ is estimated by the classifier and

    → αi > 0 for support vectors

    → αi = 0 for other data points
  - A new sample is classified by $y' = \mathrm{sign}\left(\beta_0 + x'^T\beta\right)$

- **Soft Margin SVM**
  - In real application data from different classes often overlap
  - Ideal : Still maximise the margin but allow for some points to reside the wrong side of the hyperplane (soft margin)

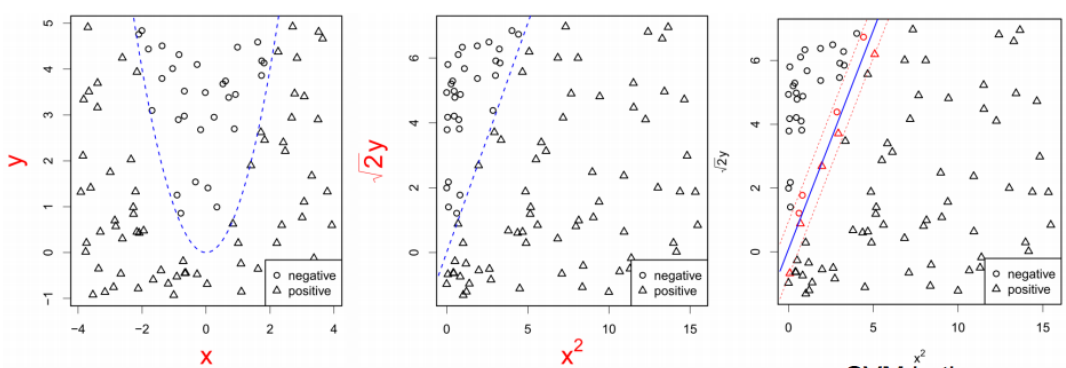
  

  - The cost function become $\min_{\beta_0,\beta}\left(\frac{1}{2}\|\beta\|^2 + C\sum_i \xi_i\right)$

    → subject to $\xi_i \geq 0, y_i\left(\beta_0 + x_i^T\beta\right) \geq 1 - \xi_i, \forall i$
  - C controls the tradeoff between the margin and the amount of misclassification in training

- **Non-Linear SVM**
  - in many application data are not linearly separate
  - Idea : Find a non-linear mapping from the **input space**

    → (higher dimensional) **feature space** where the data is linearly separable



Original space

Transformed space

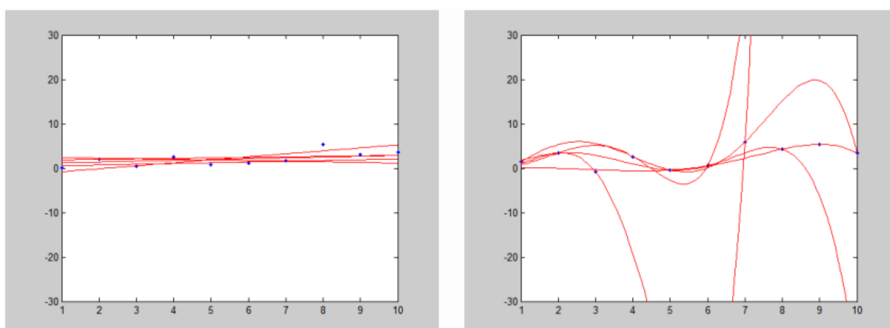
SVM in the transformed space

- **Kerbel Trick SVM**
  - instead of computing the transformation explicitly, use a **kernel to compute the inner product** in the **transformed space** & perform classification directly in that space
  - Kernel function $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
  - Kernel SVM classification $f(x') = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x_i, x')$
  - Useful kernels
    - Polynomial kernel $K(x_i, x_j) = (x_i \cdot x_j + \theta)^q$
    - Gaussian kernel $K(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$

Ensemble Learning - Bagging, Boosting, Adaboost, Random forest

The Bias-Variance Dilemma

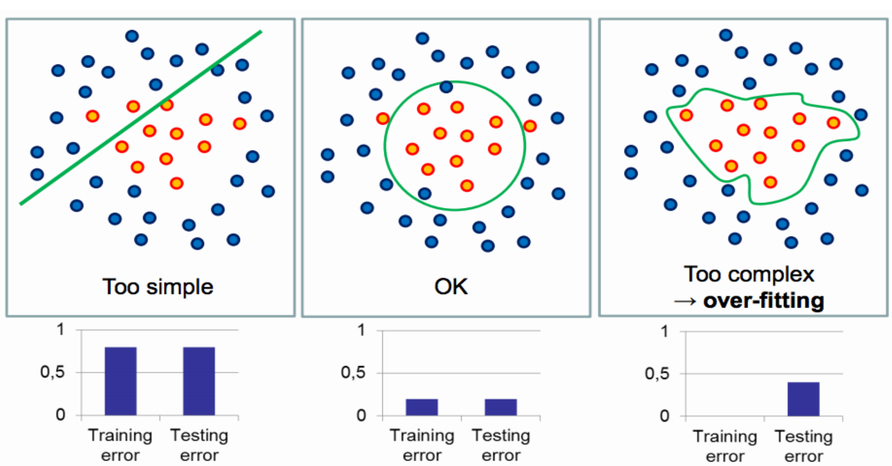
- Consider a regression problem



Linear model   Order 5 polynomial model

  - Underfit → simple models : Higher bias, Lower variance
  - Overfit → complex models : Lower bias, Higher variance

- How to choose Right Model?

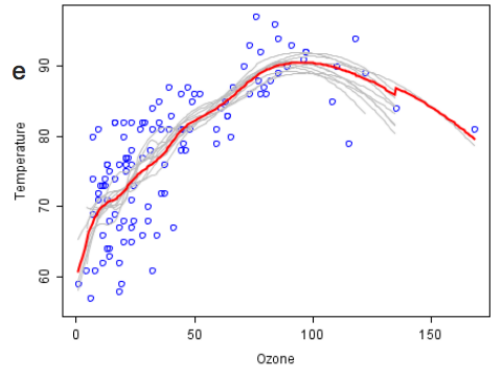


Bootstrap Aggregating (Bagging)

- Reduce the variance by averaging
- From a set of weak predictors (f1, f2,... fm)

  $$\hat{f} = \frac{1}{M}\sum_{i=1}^{M} f_i$$

  → form a committee predictor
- Example : relationship between ozone臭氧 & temperature
- Bagging with 100 bootstrap samples
- Individual predictors (gray lines) overfit!
- Committee decision (red line) → good!



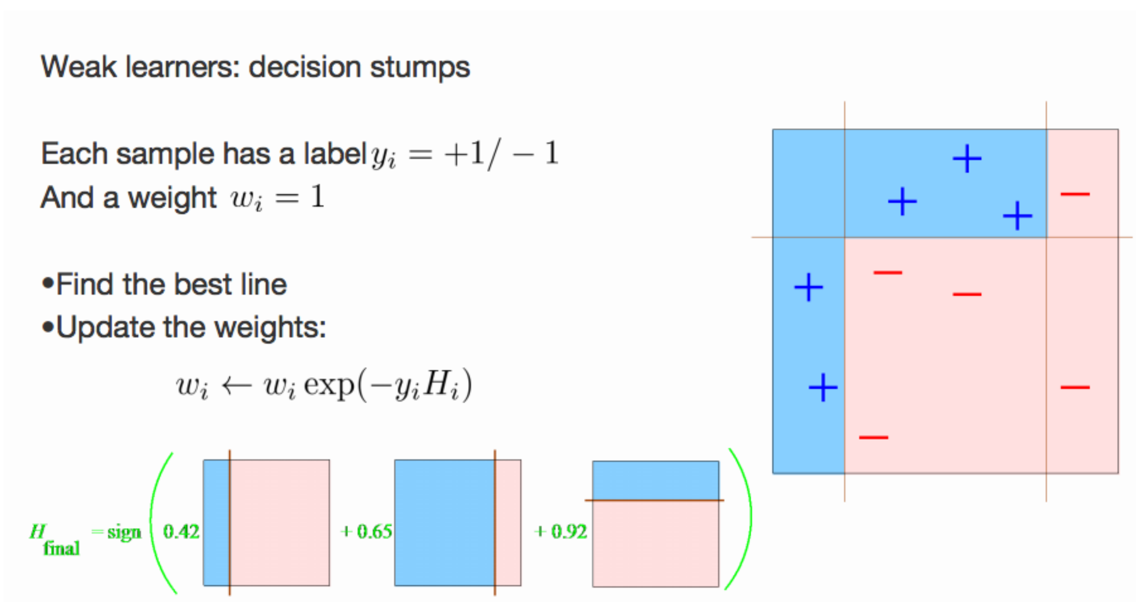
Boosting

- Reduce both bias and variance by averaging with weightings

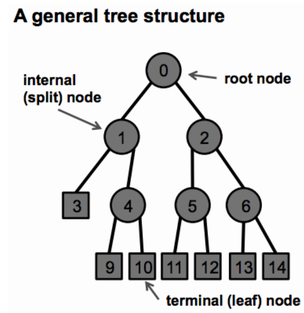
  $$\hat{f} = \frac{\sum_{i=1}^{M} w_i f_i}{\sum_{i=1}^{M} w_i}$$

- From a set of weak predictors (f1, f2,... fm) → form a committee predictor
- Wi depends on the performance of fi
- New weak learners are trained with more focus on previously misclassified samples

Adaboost

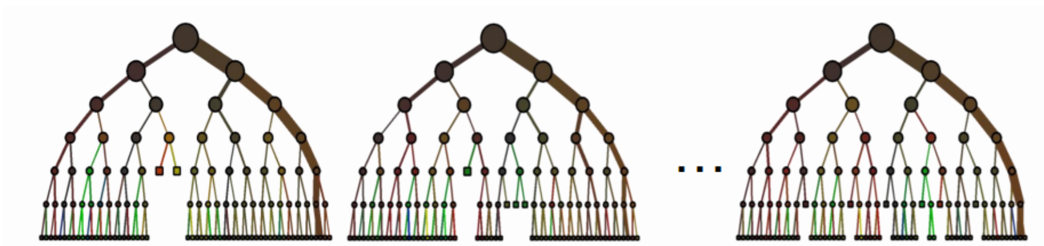


Decision Tree



- A tree is a directed acyclic graph
- A tree is a hierarchical learner – divide and conquer
- Two types of nodes: split nodes and leaf nodes
- A single decision tree is prone to overfitting

From Tree to Forest with Bagging



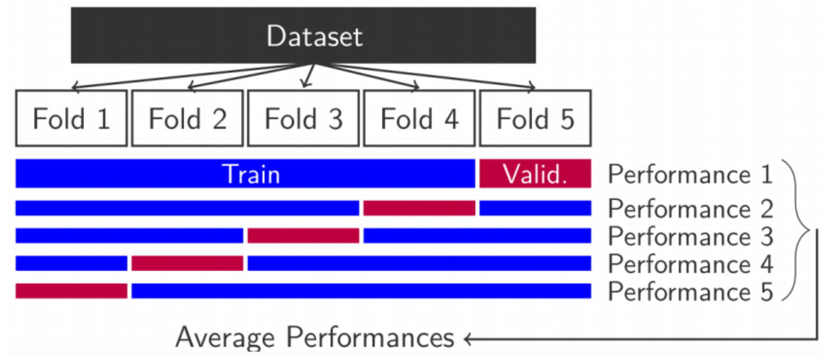
- Trees need to be highly uncorrelated
- Train each tree with a random subset of the samples
- And a random subset of variable
- Create a random subset of possible decision for each node
- Pick the best from the subset decision
- Average the output as Bagging

Evaluation - Cross validation, Evaluation Metrics

Cross Validation

- Optimise the bias-variance tradeoff
- Assume we have a large amount of data
- Construct 3 different sets
  - **Training** set → Fit the model
  - **Validation** set → Estimate prediction error to choose the best model
  - **Test** set → used to assess how well the final model generate
- **Leave-one-out Validation**
  - Use all but one sample for training & assess performance on the excluded sample
  - Not Suitable if data set is very large and/or training the classifier takes a long time

- **K-fold cross-validation**
  - Divide the data set into K folds at random
  - For each fold
    - Find a subset of “good” feature
    - Build a classifier using all samples except those in this fold and the selected subset of features
    - Use the classifier to predict the class label of samples in this fold



An illustration of K-fold cross validation

Classification Evaluation

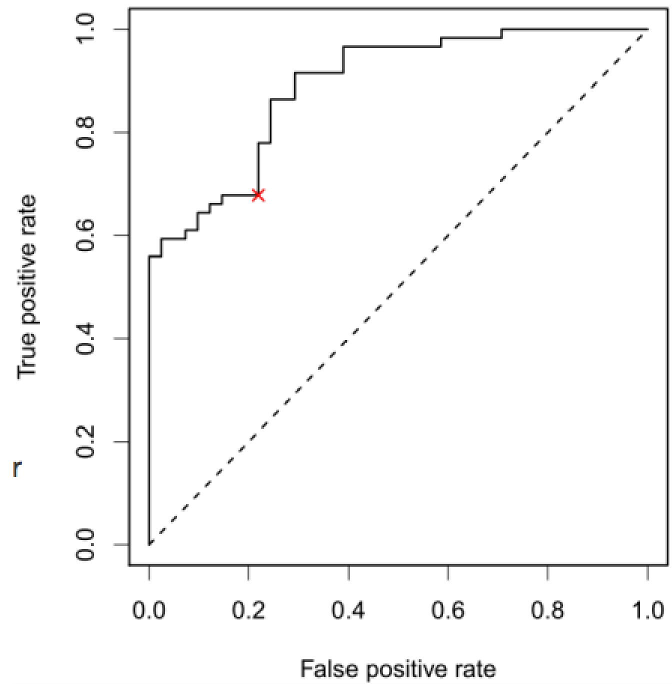
- True Positive TP → Positive Samples → correctly classified → Positive Class
- False Positive FP → Negative Samples → misclassified → Positive Class
- True Negative TN → Negative Samples → correctly classified → Negative Class
- False Negative FN → Positive Samples → misclassified → Negative Class

		Ground Truth	
		Class A	**Class B**
Prediction	**Class A**	True positive	False positive Type I Error (α)
	Class B	False negative Type II Error (β)	True negative

- **Accuracy** = TP+TN / TP=FP+TN+FN
- **Error rate =** 1 - Accuracy
- **Sensitivity (TP rate or recall) =** TP/TP+FN
- **Specificity (TN rate) =** TN/TN+FP
- **FN rate =** 1 - Sensitivity
- **FP rate =** 1 - Specificity

Receiver Operating Characteristic (ROC) Curve

- Binary classifier returns probability or score that represents the degree to which class an instance belongs to
- The ROC plot compares **sensitivity** (y-axis) with **false positive rate** (x-axis) for all possible thresholds of the classifier's score
- Visualises the tradeoff between the benefits(**sensitivity**) & costs (**FPR**)
- Line from the <u>low left to upper right corner</u> indicates the **performance of the random classifier**
- Curve of a **perfect classifier** goes through the **upper left corner at (0,1)**
- The area under curve (**AUC**) is the **probability** → classifier will rank a <u>randomly chosen positive instance</u> > <u>a negative instance</u> → good measure of the classifier performance.



Regression Evaluation

- Sum of absolute errors (SAE)

$$\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Sum of squared errors (SSE)

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Mean squared error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Root mean squared error

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$