K-NN, decision trees.

V. Kitov

Yandex School of Data Analysis

Imperial College London
Department of Physics

January 2015

Table of Contents

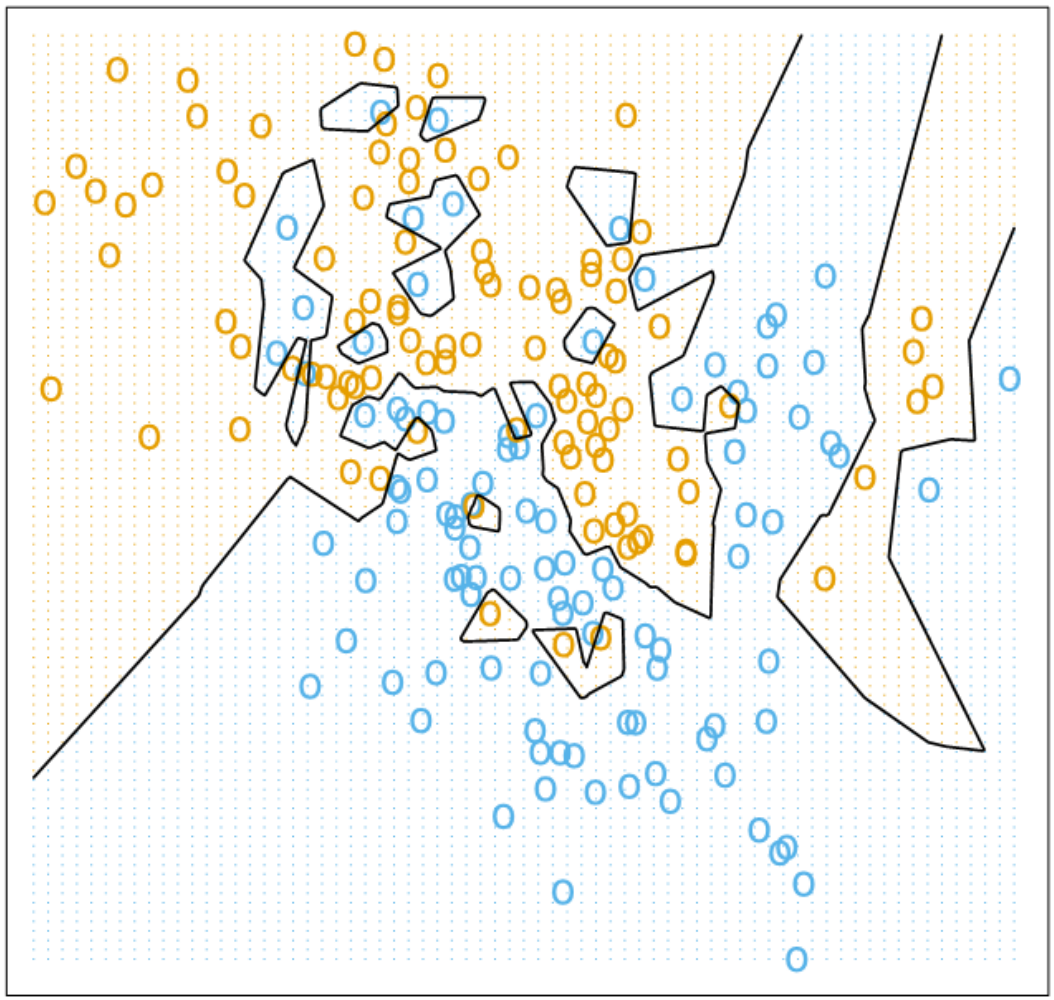
K-Nearest Neigbours classification

k nearest neigbour classification

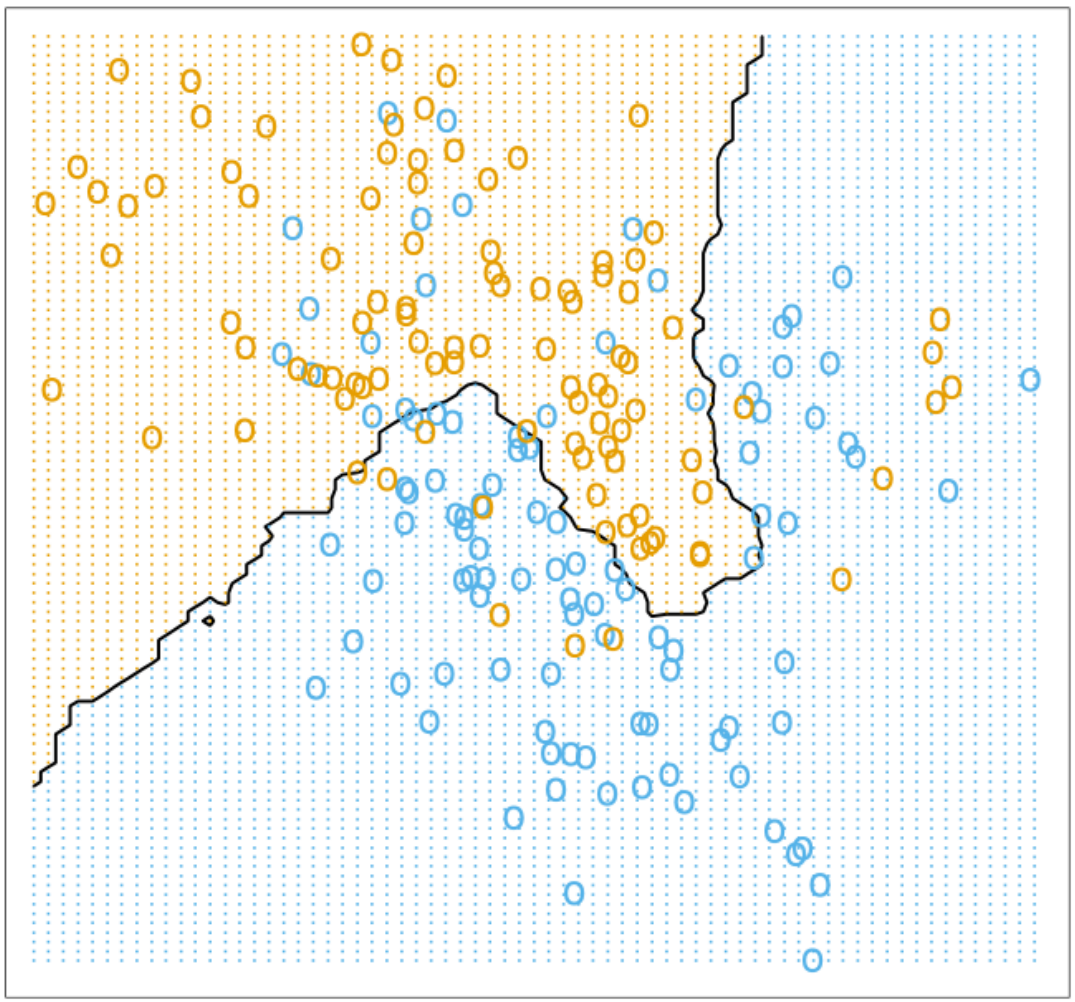
1. Find k neighbours - closest objects to target object x in the training set.
2. Assign x the most frequently represented class among k neigbours.

- Case $k = 1$ is called nearest neighbour algorithm.
- Case $k = N$ is constant classification .with most frequent class
- Regression - averaging of output among k nearest neighbours.

1 nearest neighbour classification



15 nearest neighbours classification



Properties

- Advantages:
  - only similarity between objects is needed, not features themselves

    useful for complicated objects where it's easier to distinguish objects than to define them with features.
  - easy to understand logic (if distance metric is known)
  - simple to interpret results
  - easy to fit (if distance metric is fixed)
- Disadvantages:
  - slow classification with complexity $O(n)$
  - needs to store all training data
  - besides setting k, needs specification of distance metric $d(x, y)$
- Data assumptions:
  - smoothness: close objects belong to close classes

Properties

- It is Bayesian minimum error algorithm for particular choice of density estimation.
- Performance deteriorates for high dimensional spaces
- Non-trivial to define distance when features are both continuous and categorical.

Dealing with similar rank

When several classes get the same rank, we can:

- Assign to class with higher prior probability
- Assign to class having closest representative
- Assign to class having closest mean of representatives (among nearest neighbours)
- Assign to more compact class, having nearest most distant representative

K-NN is a Bayesian classifier

- K-NN: For given x, a minimal hypersphere H, centered at x and holding K neighbours is determined.
- Suppose, that classes $\omega_1, \omega_2, ...\omega_C$:
  - have $n_1, n_2, ...n_C$ patterns in the training set, containing n elements
  - $k_1, k_2, ...k_C$ patterns in the hypersphere
- $p(\omega_i) = \frac{n_i}{n}$
- $\int_V p(x|\omega_i)dx \overset{n\to\infty}{\longrightarrow} \frac{k_i}{n_i}$, $\int_V p(x|\omega_i)dx \overset{V\to 0}{\longrightarrow} p(x|\omega_i)V$, from which it follows that $p(x|\omega_i) \to \frac{k_i}{n_i V}$

K-NN is a Bayesian classifier

- After plugging in estimate for $p(\omega_i|x) \approx \frac{k_i}{n_i V}$, we obtain, that discrimination rules for Bayes minimum error and K-NN are equivalent:

$$g_i(x) = p(\omega_i|x) = p(\omega_i)p(x|\omega_i)/p(x) = \frac{n_i}{n}\frac{k_i}{n_i V}/p(x) \propto k_i$$

Distance metric selection

- Baseline case - Euclidean metric
- Necessary to normalize features.
  - Define μ_j, σ_j, L_j, U_j to be mean value, standard deviation, minimum and maximum value of the j-th feature.

Name	Transformation	Properties of resulting feature
Autoscaling	$x'_j = \frac{x_j - \mu_j}{\sigma_j}$	zero mean and unit variance.
Range scaling	$x'_j = \frac{x_j - L_j}{U_j - L_j}$	belongs to $[0, 1]$ interval.

- Non-linear transformations incorporating features with large values:
  - $x'_i = \log(x_i)$
  - $x'_i = x^p$, $0 \leq p < 1$

Distance metric selection

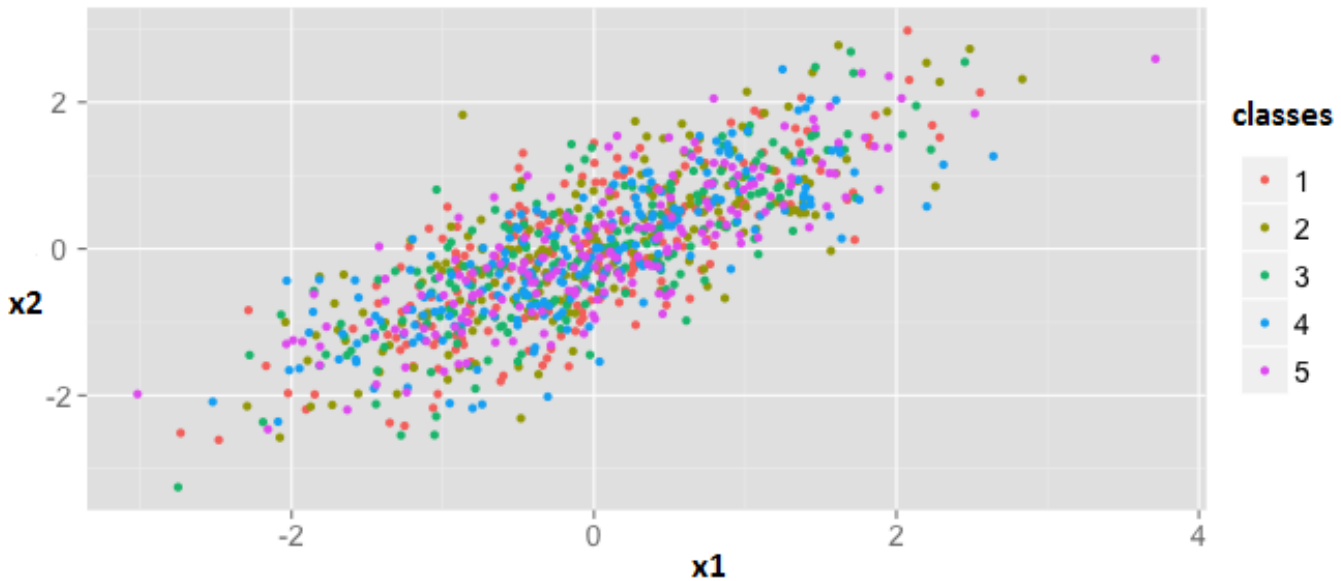
Metric	$d(x, z)$
Euclidean	$\sqrt{\sum_{i=1}^{D}(x_i - z_i)^2}$
L_∞	$\max_{i=1,2,\ldots D} \lvert x_i - z_i \rvert$
L_1	$\sum_{i=1}^{D} \lvert x_i - z_i \rvert$
Canberra	$\frac{1}{D} \sum_{i=1}^{D} \frac{\lvert x_i - z_i \rvert}{x_i + z_i}$
Lance-Williams	$\frac{\sum_{i=1}^{D} \lvert x_i - z_i \rvert}{\sum_{i=1}^{D} x_i + z_i}$
Cosine	$\frac{\sum_{i=1}^{D} x_i z_i}{\sqrt{\sum_{i=1}^{D} x_i^2} \sqrt{\sum_{i=1}^{D} z_i^2}}$

Mahalonobis distance

Mahalonobis distance

Generalization of Euclidean distance to correlated variables
$x \sim F(\mu, \Sigma)$, $y \sim F(\mu, \Sigma)$:

$$d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$



Mahalonobis distance - interpretation

- Mahalonobis distance is the distance from observation x to origin according to distribution $F(\mu, \Sigma)$ measured in terms of mean and covariance only.

$$d(x|F) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

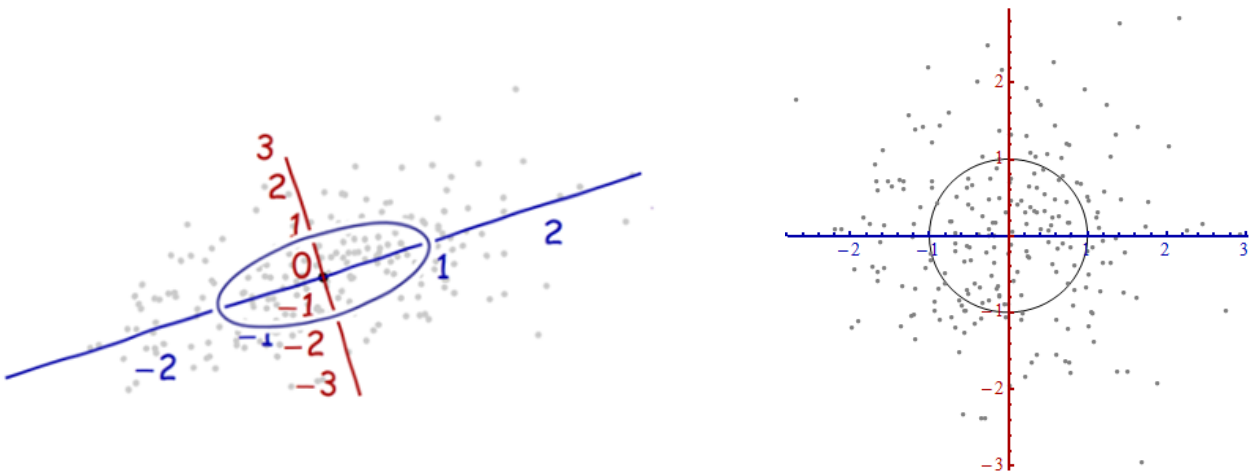
- $z = \Sigma^{-1/2}(x-\mu)$ where $\Sigma^{-1/2}$ is inverse to the square root matrix of Σ.
- z is normalized x: $\mathbf{E}[z] = \mathbf{0}$, $\mathbf{cov}[z] = I$, I is identity matrix.

$$\begin{aligned} cov[z] &= \mathbf{E}[zz^T] = \mathbf{E}[\Sigma^{-1/2}(x-\mu)(x-\mu)^T(\Sigma^{-1/2})^T] \\ &= \Sigma^{-1/2}\Sigma\Sigma^{-1/2} = I \end{aligned}$$

Mahalonobis distance - interpretation

- Mahalonobis distance is the norm of normalized x:

$$d(x|F) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)} = \sqrt{z^T z} = |z|$$



- For independent $x \sim F(\mu, \Sigma)$, $y \sim F(\mu, \Sigma)$: $x - y$ will have zero mean and covariance 2Σ.
- Mahalonobis distance gives scale and correlation aware size of this difference: $d(x, y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}$

Modifications of algorithm

- Weighted voting
  - closest neigbours vote with weights inversely proportional to their rank or distance to tested pattern
- Performance optimizations:
  - LAESA
    - distances from test pattern to training patterns are limited from below
    - using these limits from below most of training patterns are removed from the search
  - KD-trees, ball-trees
    - recursive feature space partitioning using nested hyper-rectangles or hyper-spheres.
    - on training - training set is recursively partitioned.
    - on testing - more efficient nearest neigbours search becomes possible, requiring $O(\ln n)$ instead of $O(n)$ operations.

Reduction of training objects

- Reduction of training patterns
  - editing: remove untypical patterns, representing noise.
  - condensing: remove redundant patterns which don't contribute to classification quality, only subset of patterns that correctly classify all training dataset are left

Editing algorithm

- Editing removes patterns that are untypical.
- These patterns may be:
  - outliers (true observations, obtained from non-typical conditions)
  - erroneous data caused by improper measurements or data transformations
- Untypical observations affect decisions for neighbouring area around them, so besides increasing efficiency editing may increase accuracy.

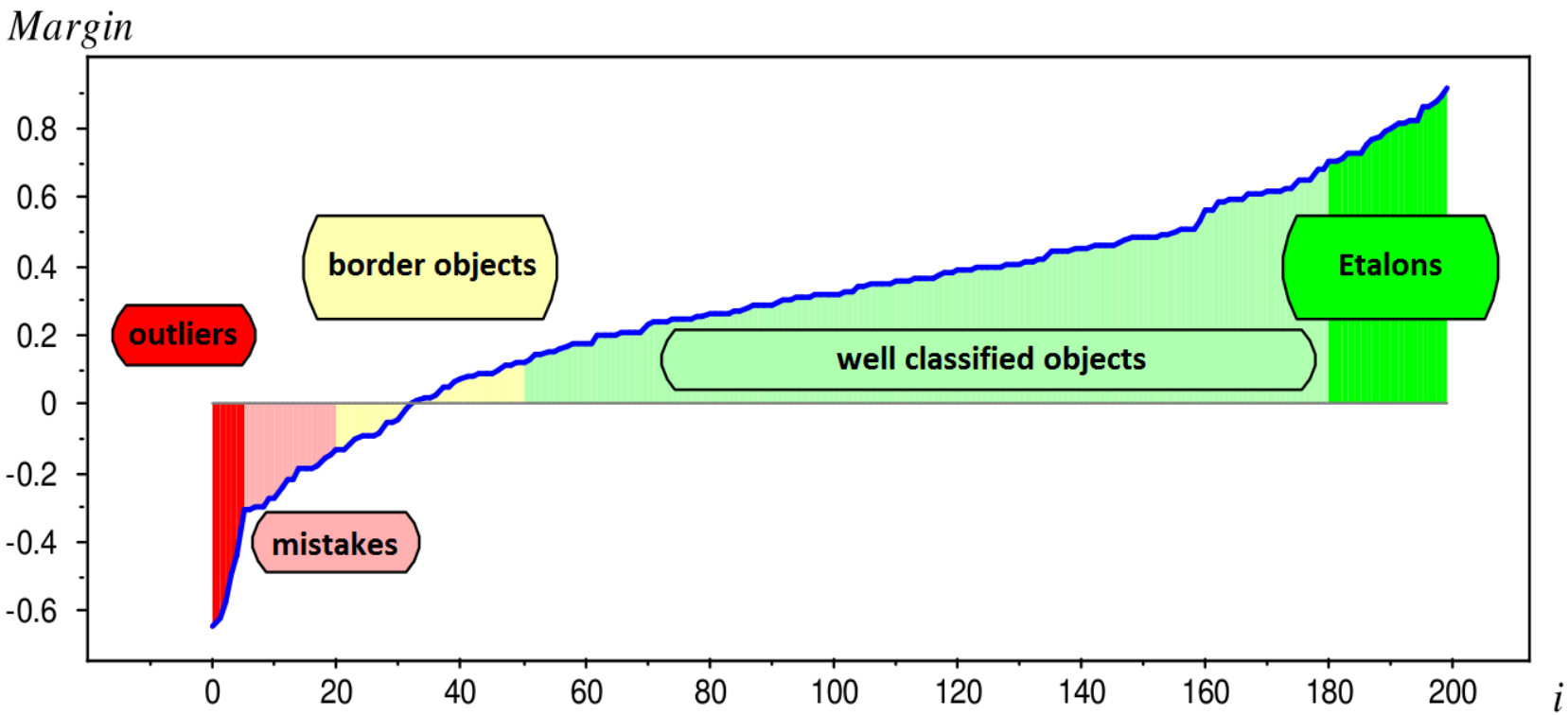
Condensing algorithm

- Condensing removes redundant observations, lying in high concentrations of patterns with the same class.
- Used mostly for efficiency purposes.
- Algorithm:
  1. divide training set into two sets: $A = \{x_1\}$ and $B = \{x_2, x_3, ...x_N\}$
  2. for each x in B:
     - if x is classified incorrectly using data only from A, then remove x from B and transfer it to A.
  3. if no transfers were made, then exit; otherwise return to step 2.

Generalization of editing, condensing

- Consider training set: $(x_1, c_1), (x_2, c_2), ...(x_n, c_n)$, where x_i is i-th feature vector and c_i is corresponding class from set of class indexes $\mathbf{C} = \{1, 2, 3, ...C\}$.
- Define margin $M(x_i, c_i) = g_{c_i}(x_i) - \max\limits_{c \in \mathbf{C} \setminus \{\mathbf{c_i}\}} g_c(x_i)$.
  - Margin is negative <=> object x_i is incorrectly classified.q
  - For correctly classified objects margin measures confidence of classification.
- Editing procedure (outliers removal) is equivalent to removing objects with lowest margin.
- Condensing procedure is equivalent to leaving highly representative objects with highest margin and objects lying on the decision border with margin close to zero.

Classification of objects using margin



Weighted voting

Let training set $x_1, x_2, ...x_N$ be rearranged to $x_{i_1}, x_{i_2}, ...x_{i_N}$ by increasing distance to the test pattern x:
$d(x, x_{i_1}) \leq d(x, x_{i_2}) \leq ... \leq d(x, x_{i_N})$.
Define $z_1 = x_{i_1}$, $z_2 = x_{i_2}, ...z_K = x_{i_K}$.
Usual K-NN algorithm can be defined, using C discriminant functions:

$$g_c(x) = \sum_{k=1}^{K} \mathbb{I}[z_k \in \omega_c], \quad c = 1, 2, ...C.$$

Weighted K-NN algorithm uses weighted voting scheme:

$$g_c(x) = \sum_{k=1}^{K} w(k,\, d(x, z_k))\mathbb{I}[z_k \in \omega_c], \quad c = 1, 2, ...C.$$

Commonly chosen weights

Index dependent weights:

$$w_k = \alpha^k, \quad \alpha \in (0,1)$$

$$w_k = \frac{K+1-k}{K}$$

Distance dependent weights:

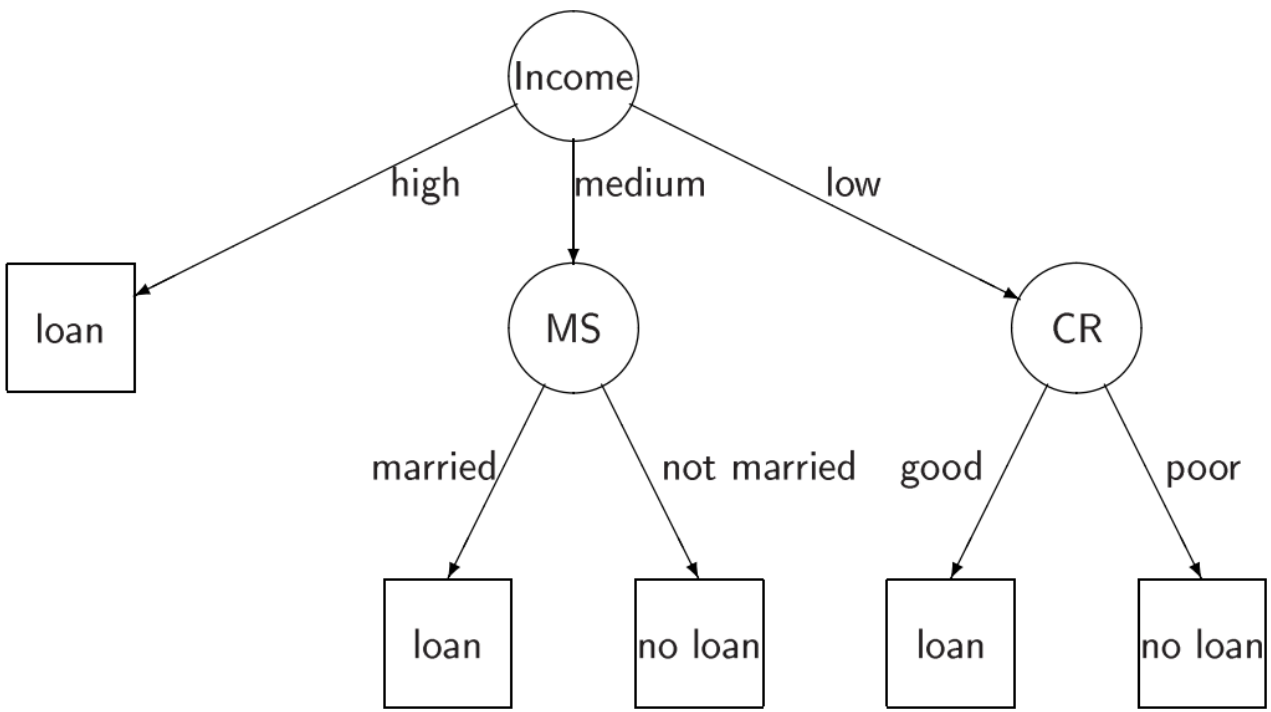
$$w_k = \begin{cases} \frac{d(z_K,x)-d(z_k,x)}{d(z_K,x)-d(z_1,x)}, & d(z_K,x) \neq d(z_1,x) \\ 1 & d(z_K,x) = d(z_1,x) \end{cases}$$

$$w_k = \frac{1}{d(z_k,x)}$$

Table of Contents

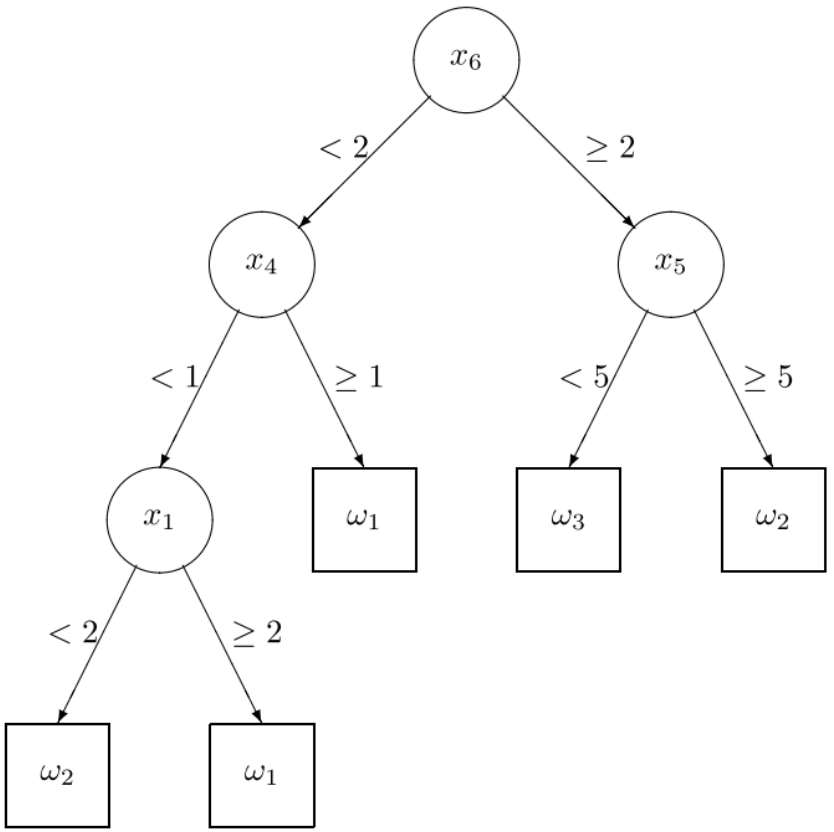
Decision trees definition

- Nodes are labelled with features
- Edges are labelled with values
- Multistage decision process
- Top-down greedy approach
- Piece-wise constant solution



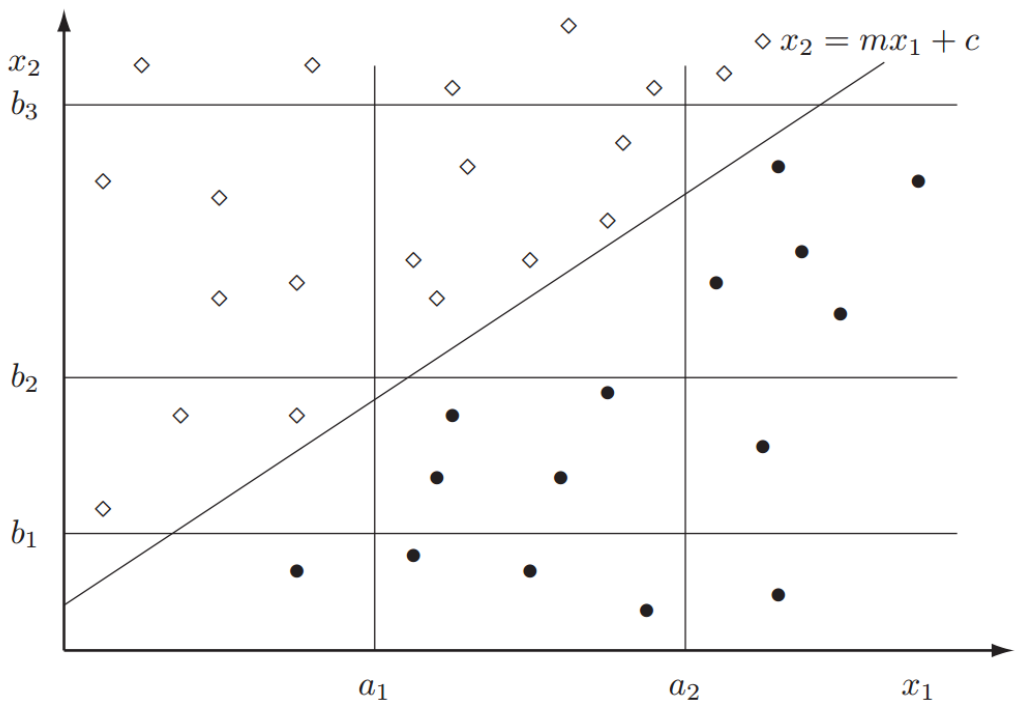
Simple splits

- Most commonly feature space is divided into hyper-rectangles - piecewise constant solution.
- Interpretable



General splits

- Many splits might be needed for general boundary.
- More complex splitting rules are possible, based on
  - linear combination of features
  - non-linear combination of features
- Less interpretable



Properties

- Decision trees are:
  - Interpretable
  - Handle both discrete (binary, nominal, ordinal) and continuous variables
  - Incorporate feature selection
  - Not online - after new data may require full rebuilding.

Specification of decision tree

- Select splitting rule
  - select splitting variable
  - select number of outgoing edges
    - binary tree (always two outgoing edges)
    - edge corresponding to each unique value of nominal variable
  - select values subset for each split
    - threshold or sequence of thresholds for continuous or ordinal variable
    - split into K children of nominal variable, taking N values: K^N variants

Specification of decision tree

- Determine which nodes are terminal
  - Bias variance:
    - pure nodes lead up to large overtrained tree
    - non-pure nodes may lead to biased model
  - Approaches
    - stopping rule
    - pruning
  - Assign class labels to terminal nodes
    - based on misclassification rate
    - based on misclassification cost

Node impurity function

- Node class probabilities:

$$p(\omega_j|x \in u(t)) = p(\omega_j|t) \approx \frac{N_j(t)}{N(t)}$$

- Impurity function:

$$I(t) = \phi(p(\omega_1|t), ...p(\omega_C|t))$$

- $\phi(q_1, q_2, ...q_C)$ is defined for $q_j \geq 0$ and $\sum_j q_j = 1$.
- ϕ attains maximum only when $q_j = 1/C$ for all j.
- ϕ attains minimum when $\exists j : \; q_j = 1, \; q_i = 0$ for $i \neq j$.
- ϕ is symmetric function of $q_1, q_2, ...q_C$.

Common impurity functions

- Gini criterion
  - probability to make classification mistake using discrete distributions of class probabilities

$$I(t) = \sum_i p(\omega_i|t)(1 - p(\omega_i|t)) = 1 - \sum_i [p(\omega_i|t)]^2$$

- Entropy
  - measure of uncertainty of discrete random variable

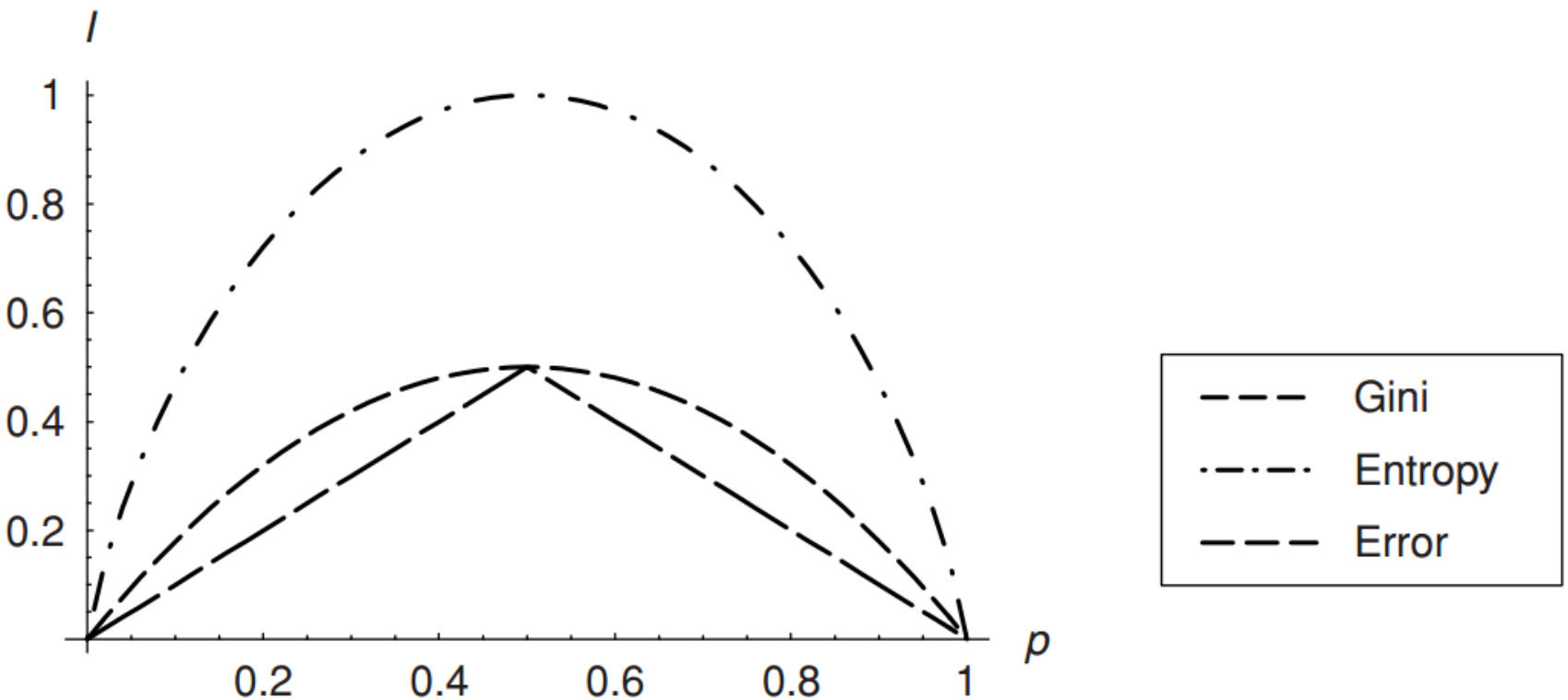
$$I(t) = -\sum_i p(\omega_i|t) \ln p(\omega_i|t)$$

- Classification error
  - error when classifying using most frequent class

$$I(t) = 1 - \max_i p(\omega_i|t)$$

Common impurity functions

Impurity functions for two-class case:



Splitting rule

- Select the split maximizing impurity gain:

$$\Delta I(t) = I(t) - \sum_{i=1}^{S} I(t_j) \frac{N_j(t)}{N(t)}$$

* where S is the number of splits, $t_1, \ldots t_S$ are child nodes of t and $N(t)$ is the number of observations falling into node t.
- If $I(t)$ is entropy $\Delta I(t)$ is called information gain.

Termination procedure

- Stopping criterion:
  - Approaches:
    - stop when change in impurity is below threshold
    - stop when number of nodes is above threshold
  - Advantages
    - simple
    - efficient
  - Disadvantages
    - needs to specify threshold
    - suboptimal: future splits may lead to greater decrease in impurity (example)
- Pruning
  - grow until node is almost pure
    - example when complete purity impossible - coincidence
  - simplify the tree replacing inner subtrees with their roots.

Assigning labels to terminal nodes

- Let $\lambda(\omega_i, \omega_j)$ denote the cost of misclassifying object, belonging to class ω_i, as belonging to class ω_j
- Then minimum cost solution is:
- $$c = \arg\min_{\omega} \sum_{i:\, x_i \in u(t)} \lambda(c_i, \omega)$$
- For zero-one loss the solution is the class yielding the smallest number of misclassifications

CART

- Define $R(t) = \frac{M(t)}{n}$, $t \in \tilde{T}$

$$R(T) = \sum_{t \in \tilde{T}} R(t)$$

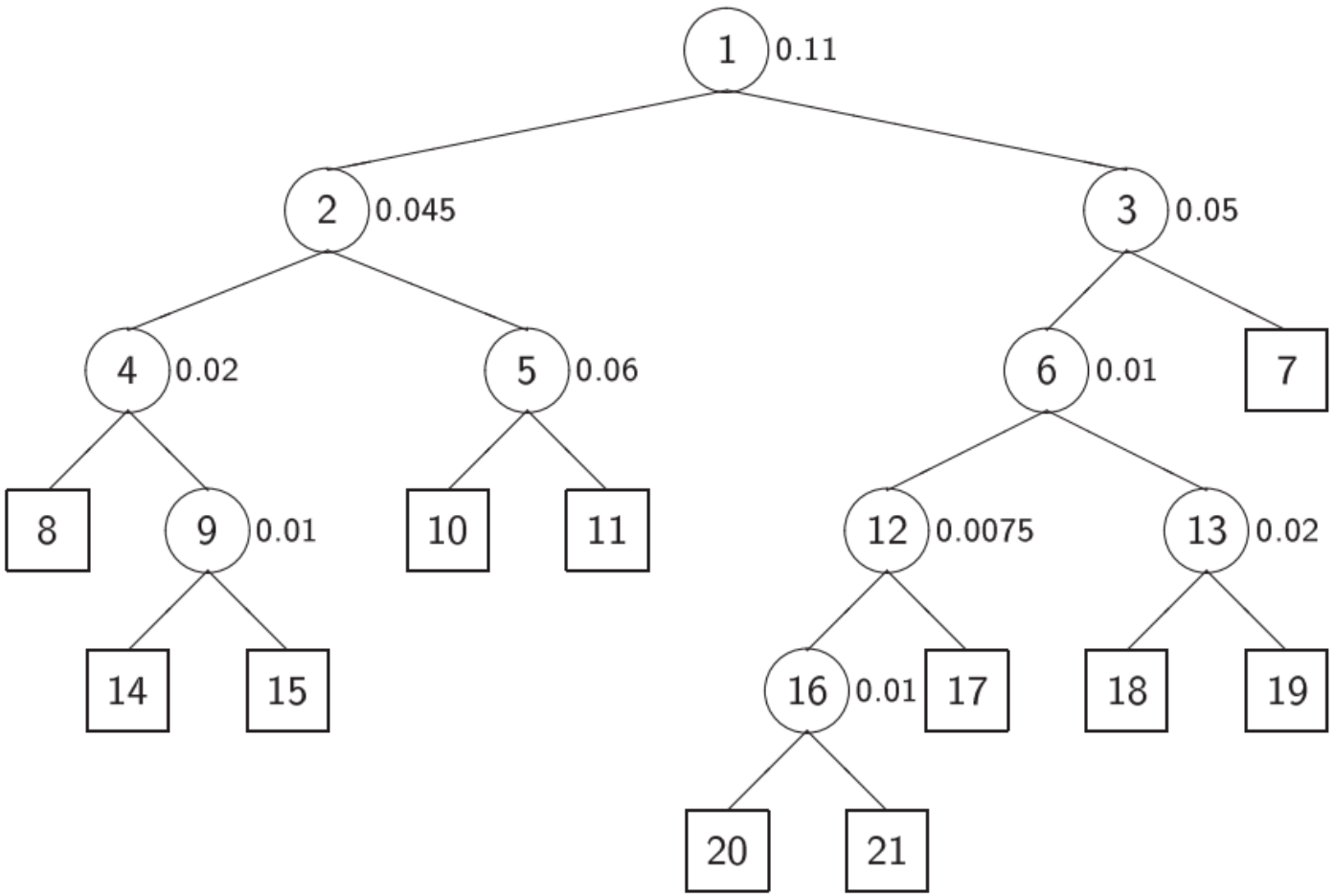
$$R_\alpha(T) = \sum_{t \in \tilde{T}} R_\alpha(t) = R(T) + \alpha|\tilde{T}|$$

- Condition when tree T_t, having root t, has equal cost with its root node:

$$\alpha_t = \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}$$

- We can build a sequence of nested subtrees from original tree to its root by successively replacing subtrees, having minimal α by their roots (and corresponding recalculation of $R(T_t)$, $R_\alpha(T_t)$ and α_t for all ancestors).

Demonstration



Demonstration

	α_k	$\lvert\tilde{T}^k\rvert$	$R(T^k)$
1	0	11	0.185
2	0.0075	9	0.2
3	0.01	6	0.22
4	0.02	5	0.25
5	0.045	3	0.34
6	0.05	2	0.39
7	0.11	1	0.5

Missing data

Missing data dealt with:

- surrogate splits (second best, third best predictor)
- predicting missing data
- labelling missing data with “missing” label