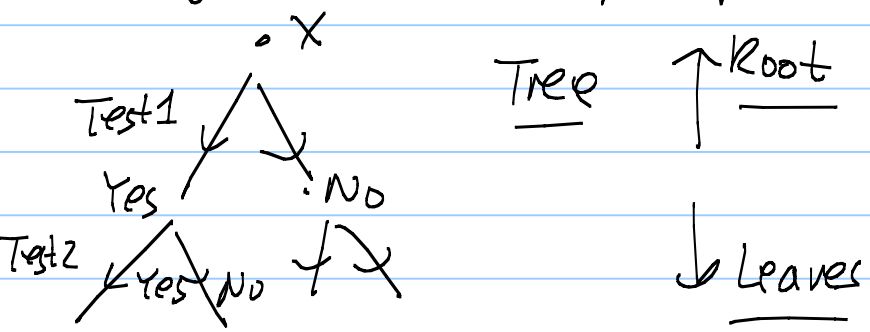
(1)

Spring 2013

Note Title 11/7/2006

Decision Trees (New Topic)

Game of Twenty Questions
Apply a series of tests to the input pattern



Notation: Set of Classified Data

$\{(x_\alpha, \omega_\alpha) : \alpha \in \Lambda\}$

Set of Tests $\{T_j : j \in \Phi\}$

Each test has response "T" or "F"

$T_j(x_\alpha) \in \{T, F\}$

Tree nodes $\{\mu\}$

Root node μ_0 at top of tree.

Each node either has two child nodes, or is a leaf node.

Each node $\{\mu\}$ has a test T_μ. Its child node μ_1 is for data x_α s.t. $T_\mu(x_\alpha) = T$

μ_2 " " " " " $T_\mu(x_\alpha) = F$.

(2)

Spring 2013

Decision Tree Notation

Define the data that gets to node μ recursively.

$$\Lambda_{\mu 1} = \{ x_a \in \Lambda_\mu \text{ s.t. } T_\mu(x_a) = T \}$$
$$\Lambda_{\mu 2} = \{ x_a \in \Lambda_\mu \text{ s.t. } T_\mu(x_a) = F \}$$

The root node contains all data $\Lambda_{\mu_0} = \Lambda$.

- Distribution of data at node μ.

$$P_\mu(\omega_j) = \frac{1}{|\Lambda_\mu|} \sum_{a \in \Lambda_\mu} \delta_{\omega_a \omega_j}$$

$$\sum_{j=1}^{M} P_\mu(\omega_j) = 1 \qquad M \text{ is no. classes}$$

- Define an impurity measure (entropy) for node μ

$$I(\mu) = -\sum_j P_\mu(\omega_j) \log P_\mu(\omega_j)$$

Note: if a node is pure, then all data in it belongs to one class.

Intuition: Design a tree so that the leaf nodes are pure — yield good classification

(3)

Spring 2013

Iterative Design.

- Initialize tree with the root node only. (so it is a leaf node).
- For all leaf nodes, calculate the maximal decrease in impurity by searching over all the tests.
- Expand the leaf node with maximal decrease and add its child nodes to the tree.

Decrease in impurity at node μ due to test T_j.

$$\Lambda^j_{\mu,1} = \{x \in \Lambda_\mu : \text{s.t. } T_\mu(x) = T\}$$
$$\Lambda^j_{\mu,2} = \{x \in \Lambda_\mu : \text{s.t. } T_\mu(x) = F\}.$$

Decrease in entropy $\Delta^j I(\mu) = I(\mu) - I(\mu^j_1, \mu^j_2)$.

where $I_{\mu^j_1,\mu^j_2} = \frac{|\Lambda_{\mu^j_1}|}{|\Lambda_\mu|} I(\mu^j_1) + \frac{|\Lambda^j_{\mu,2}|}{|\Lambda_\mu|} I(\mu^j_2)$,

Hence, for all leaf node μ calculate $\max_j I^j_{\mu_1,\mu_2}$. Select the leaf node μ and test T_j which achieves this maximum.

Spring 2013

Greedy Strategy

- Start at root node.
- Expand root node with test that maximizes the decrease in impurity — or maximizes the gain in purity.
- Repeat with leaf nodes. until each leaf node is pure.

Time Complexity: Learning algorithm is $O(|\phi||N|\{\log|N|\}^2)$ $|\phi|$ = no. of tests.

Run Time $O(\log|N|)$. Very Rapid

Notes: the design strategy is very greedy. There may be a shorter tree if you learn the tree by searching over a sequence of tests.

The number of children (2) is arbitrary. You can extend the approach to having three, or more, children

(5)

Spring 2013

There are alternative impurity measures (e.g. the Gini index)

$$I(\mu) = \sum_{i,j\,(i\neq j)} P_\mu(\omega_i) P_\mu(\omega_j) = 1 - \sum_j P_\mu^2(\omega_j)$$

Expanding the tree until all nodes are pure risks overgeneralizing. It will give perfect performance on the training dataset, but will usually cause errors on the test dataset.

Better to stop splitting the data when the impurity reaches a positive threshold, i.e. set a node to be a leaf if $I(\mu) \leq \beta$ — threshold

Then at each leaf, classify data by majority vote.

Cross Validation Strategy: learn the decision tree with different impurity thresholds β. Select the tree, and hence the β, which has best validation (consistency between training & test datasets.

(6) Spectral Clustering. Spring 2013

Note Title 11/23/2011

Undirected graph $G = (V, E)$ $V = \{v_1 \dots v_n\}$

weighted edges $w_{ij} = w_{ji} \geq 0$ big w_{ij} means big similarity.

Degree of a vertex v_i: $d_i = \sum_{j=1}^{n} w_{ij}$

$D = \text{diag}(d_i)$ degree matrix

For two disjoint subsets of V

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

Size of a subset $A \subseteq V$:

$|A|$ = number of vertices in A (un-weighted)

$\text{vol}(A) = \sum_{i \in A} d_i$ (weighted volume)

Similarities to Graphs $(x_1, \dots x_n)$ data points with similarities $s_{ij} \geq 0$

ε-neighborhood graph: $w_{ij} = \begin{cases} 1, & \text{if } s_{ij} > T_\varepsilon \text{ threshold} \\ 0, & \text{otherwise.} \end{cases}$

k-nn: $w_{ij} = \begin{cases} s_{ij}, & \text{if } v_i \text{ is a k-nn of } v_j \text{ or vice versa} \\ 0, & \text{otherwise} \end{cases}$

fully connected graph: $w_{ij} = s_{ij}$

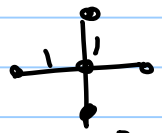
Example (Shi & Malik): v · set of pixels in an image

$$w_{ij} = e^{-\frac{\|I_i - I_j\|^2}{v^2}} \times \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{\sigma_x^2}}, & \text{if } \|x_i - x_j\| < r \\ 0, & \text{otherwise} \end{cases}$$

Graph Laplacian matrix

$L = D - W$

Why "Laplacian"



$$L = \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}.$$

Laplacian operator $-\nabla^2 u = -(u_{xx} + u_{yy})$

Properties of L ($f \in \mathbb{R}^n$):

(1) $f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$

(2) L is symmetric, positive semi-definite

(3) Smallest eigenvalue is 0, eigenvector $\mathbb{1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

(4) $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$

(5) If $0 = \lambda_1 = \dots = \lambda_k < \lambda_{k+1} \leq \dots \leq \lambda_n$, then G has k-connected components and $\mathbb{1}_{A_1} \dots \mathbb{1}_{A_k}$ are eigenvectors

$$\mathbb{1}_{A_1} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \}\, k_1$$

(7) Spring 2013

Normalized Graph Laplacian

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$
$$L_{rw} = D^{-1} L = I - D^{-1} W.$$

Properties:

(1) $f^T L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$

(2) (λ, u) eigenpair of L_{rw} (λ, w), $w = D^{1/2} u$, eigenpair of L_{sym}

(3) (λ, u) eigenpair of L_{rw}: $Lu = \lambda Du$ (generalized eigenvalue)

(4) $(0, \mathbb{1})$ eigenpair of L_{rw}: $(0, D^{1/2}\mathbb{1})$ eigenpair of L_{sym}.

(5) L_{sym}, L_{rw} are positive semi-def and $0 \le \lambda_1 \le \dots \le \lambda_n$

(6) If $0 = \lambda_1 = \dots = \lambda_k < \lambda_{k+1} \le \dots \le \lambda_n$, then G has k-connected components.

The eigenspace of $\lambda = 0$ is spanned by $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$ for L_{rw} and $D^{1/2}\mathbb{1}_{A_i}$ for L_{sym}

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number of clusters k

(1) Construct $G = (V, E)$ and W from S.

(2) Compute the un-normalized Laplacian $L = D - W$

(3) Compute the first k eigenvectors of L

or solve the general eigenvalue prob. $Lu = \lambda Du$ for L_{rw}

or compute the first k eigenvectors of $L_{sym} = D^{-1/2} L D^{-1/2}$ (for L_{sym})

(4) Let $U = [u_1 \dots u_k] \in \mathbb{R}^{n \times k}$
$= \begin{bmatrix} y_1^T \\ \vdots \\ y_n^T \end{bmatrix}$, $y_i \in \mathbb{R}^k$

(5) Cluster the points $\{y_i\}$, $i = 1 \dots n$ using e.g. k means

Output Clusters $A_1, \dots, A_k$, with $A_i = \{j \mid y_j \in C_i\}$

Data point: In the new representation $\{y_i\}$ clustering is much easier.

Main point: In the new representation $\{y_i\}$ clustering is much easier.

(8) Graph Cut point of view

Spring 2013

$\bar{A}_i$ is complement of A_i

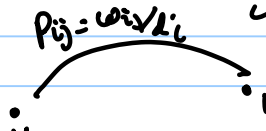
$$\text{cut}(A_1 \dots A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i)$$

$$\text{Ratio-cut}(A_1 \dots A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|}$$

$$N\text{cut}(A_1 \dots A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{\text{vol}(A_i)} = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{|A_i|}$$

Relaxations: Ratio-cut — unnormalized spectral clustering
Ncut — normalized spectral clustering.

Random Walks:



unique stationary distribution if graph connected and non-bipartite

$\pi = (\pi_1, \dots \pi_n)^T$ with $\pi_i = \frac{d_i}{\text{vol}(V)} = \frac{d_i}{\sum_{j=1}^{n} d_j}$.

Transition matrix: $P = D^{-1}W$ (note $L_{rw} = I - P$)
— (λ, u) eigenpair of L_{rw} $(1-\lambda, u)$ eigenpair of P
smallest eigenvectors of L_{rw} ⇔ largest eigenvectors of P.

$N_{cut}(A, \bar{A}) = P(\bar{A}|A) + P(A|\bar{A})$, if we start the random walk for $X_0 = \pi$ where $P(B|A) = P(X_i \in B | X_0 \in A)$

Commute distance (resistance distance)

c_{ij} = expected time it takes the random walk to travel from vertex v_i to vertex v_j and back
⇝ Instead of looking for the single shortest path it takes into account several reasonable paths.

$$c_{ij} = \text{vol}(V)\,(l^+_{ii} - 2l^+_{ij} + l^+_{jj}) = \text{vol}(V)(e_i - e_j)^T L^+ (e_i - e_j),$$
where $L^+ = \text{pinv}(L)$

$\sqrt{c_{ij}}$ can be considered as Euclidean distance on the vertices of the graph.
Construct an embedding which maps the vertices v_i of the graph on points $z_i \in \mathbb{R}^n$ such that the Euclidean distances between the points z_i coincide with the commute distances on the graph.