LECTURE NOTE #11

PROF. ALAN YUILLE

1. Decision Trees

2.

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

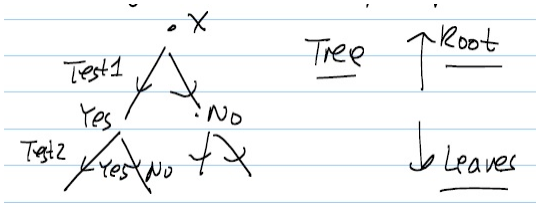

Figure 1

Notation:
Set of classified Data $< (x_\alpha, w_d) : \alpha \in \Lambda >$
Set of Tests $< T_j : j \in \Phi >$
Each test has response "T" or "F" $T_j(x_a) \in < T, F >$
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_a s.t. $T_\mu(x_a) = T$

Its child node μ_2 is for data x_a s.t. $T_\mu(x_a) = F$

3. 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
$P_\mu(w_j) = \frac{1}{|\Lambda_\mu|} \sum_{a \in \Lambda_\mu} \delta_{w_a, w_j}$
$\sum_{j=1}^{\mu} P_\mu(w_j) = 1 \qquad \mu$ is number of classes.

· Define an impurity measure (entropy) for node μ
$I(\mu) = -\sum_j P_\mu(w_j) \log P_\mu(w_j)$

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

4. 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

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

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

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

Where $I(\mu^j, \mu^j_2) = \frac{|\Lambda_{\mu^j,1}|}{\Lambda_\mu} I(\mu^j_1) + \frac{|\Lambda_{\mu^j,2}|}{\Lambda_\mu} I(\mu^j_2)$

Hence, for all leaf node μ calculate $\max_j I(\mu^j_1, \mu^j_2)$. Select the leaf node μ and test T_j which achieve this maximum.

5. Greedy Strategy

Start at root node μ_0

Expand root node with test that maximize the decrease in impurity - or maximize the gain in purity.

Repeat with leaf nodes until each node is pure.

Time Complexity: Learning algorithm is $O(|\phi||\Lambda|\{\log|\Lambda|\}^2)$

$|\phi|$ = no. of tests.

Run time $O(\log|\Lambda|)$. 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 (z) is arbitrary. You can extend the approach to having three, or more, children.

6.

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 error on the test dataset.

Better to stop splitting the data when the impurity reaches a positive threshold, it set a node to be a leaf $J(\mu) \leq \beta$

$*\beta : threshold$

Then at each leaf, classify data by majoring 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 data sets).

7. Spectral Clustering

Spectral clustering is a technique for segmenting data into non-overlapping subsets. It is used in many machine learning applications (e.g., von Luxberg 2007) and was introduced into computer vision by Shi and Malik (2000).

The data is defined by a graph with an affinity, or similarity measure, between graph nodes. The computation – to segment the data – can be performed by linear algebra followed by thresholding. Note that affinities relates to kernels (those that fall off with distance, like radial basis functions) used in machine learning. For some problems it is easier to define affinities between objects directly instead of obtaining them by the more standard method of specifying the objects by features and then calculating the distance between the features.

Spectral clustering is an alternative to probabilistic methods for segmentation. The main difference is that the probabilistic models define data at the nodes of the graph while spectral clustering defines data at the edges between nodes. (There are ways to relate the two approaches which will be discussed later).

For image segmentation a typical affinity between pixels i and j is defined by $w_{ij} = \exp\{-\gamma|I_i - I_j|\}\exp\{-\tau|x_i - x_j|\}$ where I_i, I_j are the intensities at pixels i, j and x_i, x_j are their spatial positions. Hence the affinity is high between neighboring pixels which have similar intensity values (small $x_i - x_j$ and small $|I_i - I_j|$) and the affinity is small between pixels which are far apart (large $x_i - x_j|$) or which have very different intensity values (large $|I_i - I_j|$). If this affinity is used, the spectral clustering will segment the data into subregions within which the intensity values changes slowly with position.

References.

- U. von Luxburg, A tutorial on spectral clustering, Stat.Comput.,2007
- Shi and Malik, Normalized cuts and image segmentation, PAMI, 2000

8. Basic Concepts

Let $G = (V, E)$ be an undirected graph with nodes $V = v_1, v_2, ..., v_n$. The graph has weighted edges $w_{ij} = w_{ji} \geq 0$ which are called affinities and are measures of similarity between nodes. Large w_{ij} means strong affinities, or bonds, between node i and node j.

The degree of a vertex v_i is defined as: $d_i = \sum_{j=1}^{n} w_{ij}$. The degree matrix of the graph is defined by $\mathbf{D} = diag\{d_1, d_2, ..., d_n\}$.

For two subsets A, B (which do not need to be disjoint) of V, $w(A, B)$ is defined by:

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

The size of a subset $A \subseteq V$ has two definitions.

$$|A| = number\ of\ vertices\ in\ A \quad \text{(unweighted volume)}$$

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

8.1. **Examples of affinities.**

$$w_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Shi and Malik's definition of w_{ij} is given by:

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

where r, σ_x, σ_I are parameters.

9. The Graph Laplacian Matrix (ref. Chung, Spectral graph theory, AMS-1997)

This section defines graph Laplacians on the graphs. In the special case where the affinities w_{ij} takes values $\{0, 1\}$ then the connected components of the graph can be found from the eigenvectors with zero eigenvalues. This enables us to segment the graph into its connected components by linear algebra. If we allow the w_{ij} to take continuous values, as will happen for computer vision applications, then we can estimate a segmentation of the data by using the eigenvectors of the Laplacian with sufficiently small eigenvalues. There are several different Laplacians (normalized and unnormalized) which will give different segmentations.

9.1. **Unnormalized graph Laplacians.** Given an undirected, weighted graph $G = (V, E)$, its Laplacian matrix is defined to be

$$L := D - W \tag{3}$$

where $D = diag(d_i)$ and $W = (w_{ij})_{n \times n}$ with $n = |V|$. Note that L does not depend on w_{ii} (which cancels between D and W).

Why do we call this matrix the "Laplacian"? From Figure 2, we see that it is similar to the standard discretization for the Laplacian differential operator $-\nabla^2 u = -(u_{xx} + u_{yy})$ in the special case where $w_{ij} = 1$ for nearest neighbor pixels and $w_{ij} = 0$ otherwise:

Here are some useful properties of L which will be useful for spectral clustering:

1. $f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2, \forall f \in \mathbb{R}^n$
2. L is a symmetric positive semi-definite matrix (this follows from 1).
3. The smallest eigenvalue is 0, the corresponding eigenvector is the vector $\mathbb{1} = (1, 1, ..., 1)$.
4. L has n non-negative eigenvalues. $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_k < \lambda_{k+1} \leq \cdots \leq \lambda_n$.
5. If $0 = \lambda_1 = \lambda_2 = \cdots = \lambda_k < \lambda_{k+1} \leq \cdots \leq \lambda_n$, then G has k-connected components $A_1, \ldots, A_k$ ($V = \bigcup_{i=1}^{k} A_i$). The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of these components.

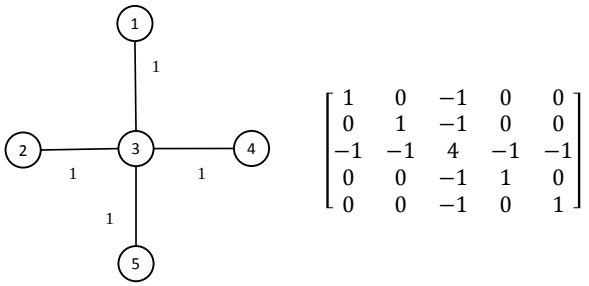

FIGURE 2. A simple graph and its corresponding Laplacian

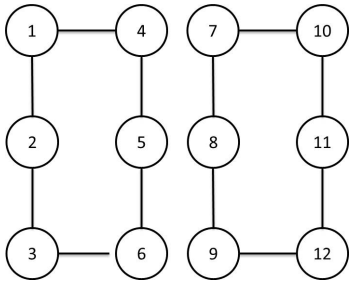

FIGURE 3. A simple graph G with two connected components

Here is a simple example of Property 5. The graph $G = (V, E)$ in Fig 3 has two connected components, A_1 and A_2. $W = \{w_{ij}\}$ of G is defined in Equation (6).

$$w_{ij} = \begin{cases} 1 & e_{ij} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Let $L = D - W$ and calculate the eigenvalues and eigenvectors of L. It follows that the first two eigenvalues λ_1, λ_2 are 0, and the corresponding eigenvectors (u_1 and u_2) correspond to the connected components A_1 and A_2.

$$A_1 = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$
$$A_2 = \{v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$$
$$u_1 = (1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)$$
$$u_2 = (0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)$$

Note that the eigenvalues are degenerate and so that eigenvectors will be of form $cos\theta u_1 + sin\theta u_2$ and $sin\theta u_1 - cos\theta u_2$ where θ is any value. An additional algorithm is needed to find u_1 and u_2 as described in the next section.

9.2. **Normalized graph Laplacians.** There are two matrices which are called normalized graph Laplacians in the literature. Both matrices are closely related to each other and are defined by:

$$L_{sym} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{1/2} \quad \text{(Ng, Jordan, Weiss, 2002)} \tag{5}$$

$$L_{rw} := D^{-1} L = I - D^{-1} W. \quad \text{(Shi, Malik, 2000)} \tag{6}$$

The normalized Laplacian matrix satisfy the following properties:

(1) $f^T L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}})^2, \forall f \in \mathbb{R}^n$
(2) (λ, u) is an eigenpair of L_{rw} if and only if $(\lambda, w = D^{1/2} u)$ is an eigenpair of L_{sym}.
(3) (λ, u) is an eigenpair of L_{rw} if and only if λ and u solve the generalized eigenproblem $Lu = \lambda D u$.
(4) $(0, \mathbb{1})$ is an eigenpair of L_{rw}, and $(0, D^{1/2}\mathbb{1})$ is an eigenpair of L_{sym}.
(5) L_{sym} and L_{rw} are positive semi-definite and have n non-negative eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_n$.
(6) If $0 = \lambda_1 = \cdots = \lambda_k < \lambda_{k+1} \leq \cdots \leq \lambda_n$, then G has k-connected components $A_1, \ldots, A_k$. And the eigenspace of eigenvalue $\lambda = 0$ is spanned by the $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ for L_{rw} and $D^{1/2}\mathbb{1}_{A_1}, \ldots, D^{1/2}\mathbb{1}_{A_k}$ for L_{sym}.

10. Spectral Clustering Algorithms

Input: Affinity matrix $S \in \mathbb{R}^{n \times n}$, and number of clusters to construct k.

(1) Compute the unnormalization Laplacian $L = D - W$.
(2) Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
    or Solve the general eigenvalue problem, i.e. $Lu = \lambda D u$, to get $u_1, \ldots, u_k$ (for L_{rw}).
    or Compute the first k eigenvectors $u_1, \ldots, u_k$ of $L_{sym} = D^{-1/2} L D^{-1/2}$(for L_{sym}).
(3) Let

$$U = [u_1, \ldots, u_k] \in \mathbb{R}^{n \times k} \tag{7}$$

$$= \begin{pmatrix} y_1^T \\ \vdots \\ y_n^T \end{pmatrix}, y_i \in \mathbb{R}^k \tag{8}$$

that is, y_i is the vector corresponding to the i-th row of U.
(4) Cluster the k-dimension points $(y_i)_{i=1,\ldots,n}$ using e.g. k-means, into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$, with $A_i = \{j | y_j \in C_i\}$.
Main point: In the new representation y_i, clustering is much easier. For example, suppose $n = 5$ and there are two connected components — nodes $1, 2, 3$ and nodes $4, 5$. Then the zero eigenvectors will be of form $(cos\theta, cos\theta, cos\theta, sin\theta, sin\theta)$ and $(-sin\theta, -sin\theta, -sin\theta, cos\theta, cos\theta)$, where θ is an angle (this is because we know the zero eigenvectors must lie in the subspace

spanned by $(1,1,1,0,0)$ and $(0,0,0,1,1)$, because of property 6 of the laplacian, and the eigenvectors must be orthogonal). Then if we set $k = 2$ we find that the clusters are $(cos\theta, -sin\theta)$ and $(sin\theta, cos\theta)$. Then the first three points are associated to the first cluster (i.e. the first connected component) and the last two points are associated to the second cluster (second connected component).