Regression Trees: Split optimization Task

$$\min_{d,s}\left\{\min_{c_1}\sum_{\vec{x}\in R_1(d,s)}\left(\hat{p}(\vec{x})-c_1\right)^2+\min_{c_2}\sum_{\vec{x}\in R_2}\left(\hat{p}(\vec{x})-c_2\right)^2\right\}$$

Trees are prone to overfitting - limit depth with regularizer:

$$J_{reg}=\sum_{j=1}^{k}\int_{\vec{x}\in R_j}\left(p(\vec{x})-\hat{p}(\vec{x})\right)^2 d\vec{x}+\alpha\cdot k$$

k: number of splits

α: weighing parameter

Curse of dimensionality

- statistically all distances become large
- similarity measure (distance) breaks

Manifold Learning

Goal: reduce dimensionality

PCA: Project onto hyperplane with largest variance

MDS: "PCA on distances / dissimilarities"

PCA Concept

C: $p \times p$ covariance matrix

V: matrix of column eigenvectors

L: diagonal matrix with eigenvalue entries

$$C=\frac{X^TX}{n-1} \Rightarrow C=V L V^T \quad \text{(diagonalized)}$$

$\Rightarrow$ Perform SVD of X: $X=USV^T$ where S is a diagonal matrix

$$\lambda_i=\frac{s_i^2}{n-1}$$

PA Summary ② Manifold Learning 25.07.19

MDS Concept

$S = \{x_1, \dots, x_N\}$, $x_i \in \mathbb{R}^d$

$X = [x_1, \dots, x_N] \in \mathbb{R}^{d \times N}$

$D^2 = [d_{ij}^2]_{i,j \in [1,\dots,N]}$, $d_{ij}^2 = (x_i - x_j)^T (x_i - x_j)$

Given: D Compute: X

$$d_{ij}^2 = (x_i - x_j)^T (x_i - x_j) = x_i^T x_i + x_j^T x_j + 2 x_i x_j$$

$$\Rightarrow D^2 = \text{diag}(X^T X) \cdot \vec{1}^T + \vec{1}\, \text{diag}(X^T X)^T - 2X^T X$$

C: Centering Matrix $C = \left(I - \frac{1}{N} \vec{1}\vec{1}^T\right)$

$$-\frac{1}{2} C D^2 C = \dots = X^T X \quad \text{(centered distances)}$$

$$\Rightarrow X^T X \quad [\text{eigendecomposition}] = U \Sigma U^T$$

$$\Rightarrow X = \Sigma^{1/2} U^T$$

— done —

1. Determine the m largest eigenvalues & eigenvectors
2. Drop the remaining eigenvectors

⇒ gives us feature vectors (rows) in new, reduced space

Proportion of variance explained by p dimensions:

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{n-1} \lambda_i}$$

ISOMAP Algorithm

- non-linearity patch for MDS
- close distance: Euclidean distance

  others: distance via graph edges (shortest path)

  [sparse adjacency matrix]

  "geodesic distance"

Locally Linear Embedding

Idea: treat local neighborhood of a point as linear

Algorithm:

1.) Define the neighborhood (kNN / distance thresholding)

2.) Solve w_{ij} in the high dimensional space

$$\min \sum_i \left\| x_i - \sum_{j \in N(x_i)} w_{ij} x_j \right\|_2^2 \quad \text{s.t.} \quad \sum_{j \in N(x_i)} w_{ji} = 1$$

linear constraint

3.) Solve for $x_i' \in \mathbb{R}^{d'}$ $(d' \ll d)$

$$\sum_i \left\| x_i' - \sum_{j \in N(x_i)} w_{ij} x_j' \right\|_2^2 \quad \text{s.t.} \quad \frac{1}{N} \sum_i x_i' \cdot x_i'^{\top} = I$$

identity covariance

$$\sum_i x_i' = \vec{\emptyset}$$

zero mean

MODIFICATION of step 2

translation

$$2^*) \quad \sum_i \left\| x_i - \sum_{j \in N(x_i)} w_{ij} x_j \right\|_2^2 = \sum_i \left\| (x_i - t) - \sum_{j \in N(x_i)} w_{ij} (x_j - t) \right\|_2^2$$

$$\text{set } x_i = t \Rightarrow \sum_{i=1}^{N} \left\| \sum_j w_{ij} (x_j - x_i) \right\|_2^2 = \left\| M_i \vec{w}_i \right\|_2^2$$

$\Rightarrow$ minimize $(M_i w_i)^{\top} (M_i w_i) + \lambda (1 - w_i^{\top} w_i)$ ... $\Rightarrow$ $M_i^{\top} M_i w_i = \lambda w_i$

eigenvector / -value $\Rightarrow$ solvable

Laplacian Eigenmaps

1.) Build adjacency graph
2.) Compute affinities between nodes
} Construct Similarity Matrix
3.) Perform Eigendecomposition
4.) Low-Dim Embedding

$$L = D - W$$

$$d_{ij} = \begin{cases} \sum_{k=1}^{N} w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$w_{ij} = \begin{cases} \text{weight between node } i \text{ and } j, & i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Heat Kernel

$$w_{ij} = e^{-\|x_i - x_j\|_2^2}$$

Binary affinity

$$w_{ij} = \begin{cases} 1 & \text{if } \|x_i - x_j\| \leq t \\ 0 & \text{otherwise} \end{cases}$$

Objective Function

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i' - x_j'\|_2^2 w_{ij}$$

$$= 2 \vec{x}'^T (D - W) \cdot \vec{x}'$$

(D − W): Graph Laplacian

$$\Rightarrow \boxed{\min \vec{x}'^T L \vec{x}' \quad \text{s.t.} \quad \vec{x}' D \vec{x}' = 1}$$

$$\Rightarrow D^{-1} L \vec{x}' = \lambda \vec{x}'$$

PA Summary (5) — Trees — 25.07.14

(also see 24.07 (3))

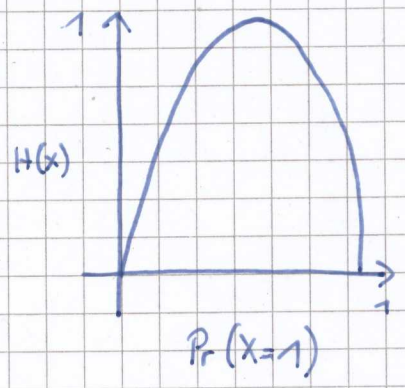
Decision Trees

Information Gain (for splitting at S_j)

$$I = H(S_j) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{|S_j|} \cdot H(S_j)$$

Entropy

$$H(S_j) = -\sum_{c \in C} p(c) \cdot \log(p(c))$$

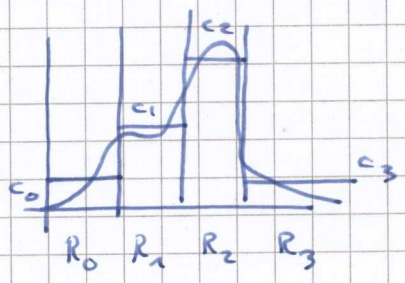


Binary Entropy Function

Regression Forests

Goal: Predict continuous label

Regression Tree: Estimation of c_j for each R_j



Split optimization task:

$$\min_{d,s} \left\{ \min_{c_1} \sum_{x \in R_1(d,s)} \left(\hat{p}(\vec{x}) - c_1\right)^2 + \min_{c_2} \sum_{\vec{x} \in R_2(d,s)} \left(\hat{p}(\vec{x}) - c_2\right)^2 \right\}$$

Prevent overfitting by using a regularizer:

$$J_{Reg} = \sum_{j=1}^{K} \int_{x \in R_j} \left(p(\vec{x}) - \hat{p}(\vec{x})\right)^2 d\vec{x} + \alpha \cdot K$$

K: number of splits

α: weighting parameter

Density Forests

- Each leaf node is a multivariate Gaussian

Information Gain – same as Regression

Entropy

$$H(S_j) = \frac{1}{2} \log\left((2\pi e)^d \, |\Lambda S_j|\right)$$

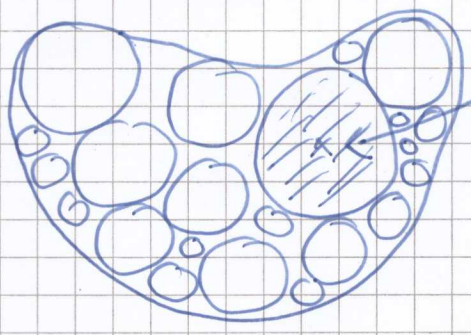
$|\Lambda S_j|$: determinant of the covariance of the data ("volume of a cluster")

⇒ Information Gain:

$$I(S_j, \mathcal{T}) = \log\left(|\Sigma(S_j)|\right) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{|S_j|} \cdot \log\left(|\Lambda(S_j^i)|\right)$$

Manifold Forests

1. Partition feature space via Density Forest training
2.) Use Regions as local neighborhoods → similarity matrix
3.) Apply Laplacian Eigenmaps



Density Tree partitioning ⇒ sparse similarity matrix

Affinity Model(s)

$w_{ij} = e^{-Q(x_i, x_j)}$ ⇐ Affinity

$Q(x_i, x_j)$ ⇐ distance function

Mahalanobis

$$Q(x_i, x_j) = \begin{cases} d_{ij}^T \left(\Lambda_{\ell(x_i)}\right)^{-1} d_{ij} & \text{if in same leaf } \ell(x_i) \\ \infty & \text{otherwise} \end{cases}$$

Binary

$$Q(x_i, x_j) = \begin{cases} 0 & \text{if in the same leaf} \\ \infty & \text{otherwise} \end{cases}$$