PA 2018 -01

Recap: PR

结构框图

- Recording(ϕ)
  - Sampling
  - quantization
- Preprocessing(ϕ')
  - smoothing
  - denoisiong
  - Histogram Equlazation
- Fearure Extraction(c)
  dim(f)>>dim(c)
  - edge filter
  - HOG
  - SIFT/SURF
- Classification
- Training

Interlude: Deep Learining

DL needs more data.
eg. 1/10000 hard to learn

low generise

overfitting

underfittting

100*100 data set class dog

75% train data, 25% test set

- count determine relative frequences of correct predictions
  eg accucy recall F1-Score

ROC croves

data $[0, 1]$ label = $[dog, notdog]$
ROC偏向左上的效果最好，对角线的是随机猜测。

Use AUC(area under the curve) to evalue it

Three point

Strict separation of test/train data

- foldas train/test

Cross validation

do not use train data to test

Overfitting

train to well

PA 2018 -02

Remark : Least squares decision boundary

KNN and DB
3-NN vs 1-NN

Bias-Variance tradeoff

PA Mind Map

Pattern analysis

- Density Estimation
  - Task: To estimate a continuous Probability density function from a discrete finite set of observations
  - eg:
    - MLE(Maximum Likelihood Estimation):
      - Assume that PDF is a Gaussian,Poisson,gamma,polynomial distribution
      - Estimate mean and covariance
      - More General: Assume any parametric function
    - MAP(Maximum a posteriori Estimation):
      - similar to MLE, but with prior knowledge
      - $\theta^* = argmax(\theta|x_1, x_2, x_3 \ldots . x_n)$,where θ denotes the parameter of the model (eg mean and covariance for a Gaussian), and $x_1 \ldots x_n$ denotes N observations/samples/feature points
      - $argmax_\theta \frac{p(\theta) * p(x_1, x_2, x_3 \ldots . . x_n|\theta)}{p(x_1, x_2, x_3 \ldots . . x_n)}$
      - assume independence of observations!
      - $argmax_\theta p(\theta) * \prod_{i=1}^{N} p(x_i|\theta)$
      - $argmax_\theta log p(\theta) + \sum_{i=1}^{N} log(p(x_i|\theta))$
- Mean Shift algorithm
- Clustering
  - Group observations according to vicinity in the feature space
- Manifold Learning
  - Dimensionality reduction
  - PCA(Principal component Analysis): Linear projection of the d-dimensional data onto a d_,-dimensional subspace that maximizes the covariance of the data.

up now is "Roughly analytical"

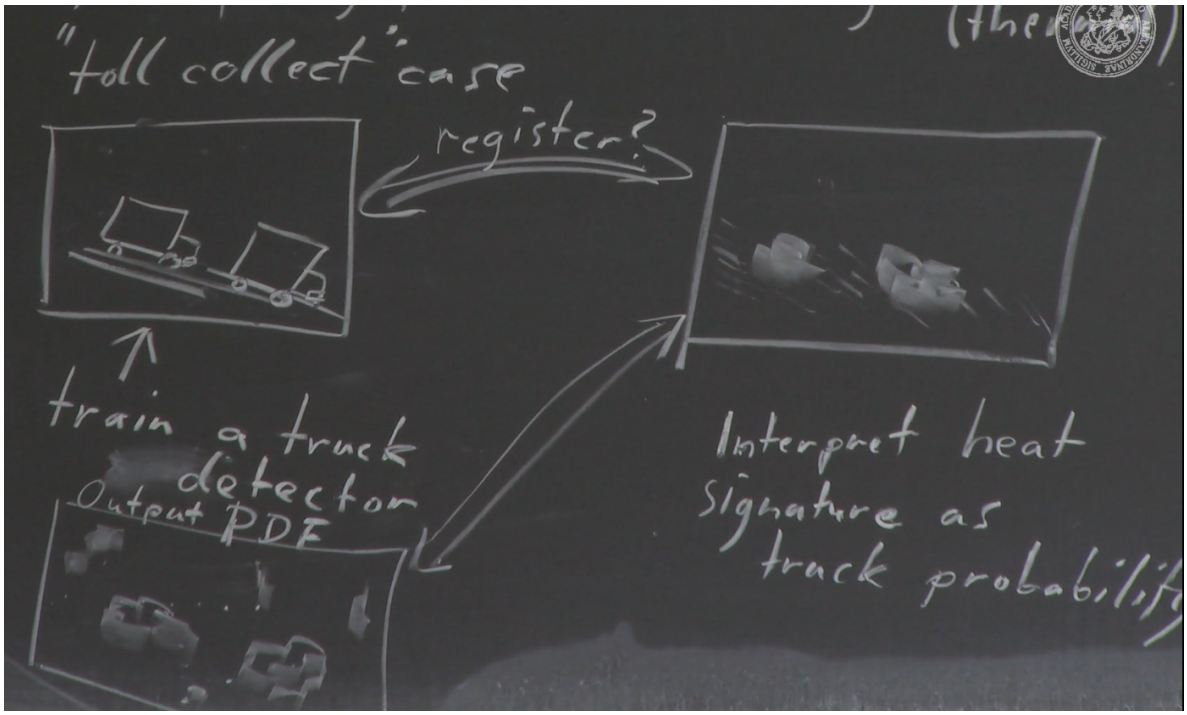
- Random Forests
  - Learning-based probabilistic tool to partition the feature space
- Graphical Models HMMs,MRFs
  - To model the sequence of data,
  - Hidden Markov Models

- Markov Random Fields

PA-2018 - 03: Non-parasitic density function

why might i want to estimate a probability density function (PDF) from some discreate observation?

- compute statistical measures, for example in Mutual information
  - image from modality 1
    - e.g. a photograph
  - image from modality 2
    - e.g. infrared image



- sample(draw) new observations with the same distribution as the actual(measurement)

Parzen Window Estimator

Idea: Given a set of discrete observations "smear" them out to obtain a PDF
Let $S = \{x_1, x_2, x_3, \ldots . x_N\}$ denote the set of observations.
Let PR denote the probability that x is falling into region R.

$$PR = \int_R p(x) \partial x \tag{1}$$

if we assume $p(x)$ is approximately constant in R,

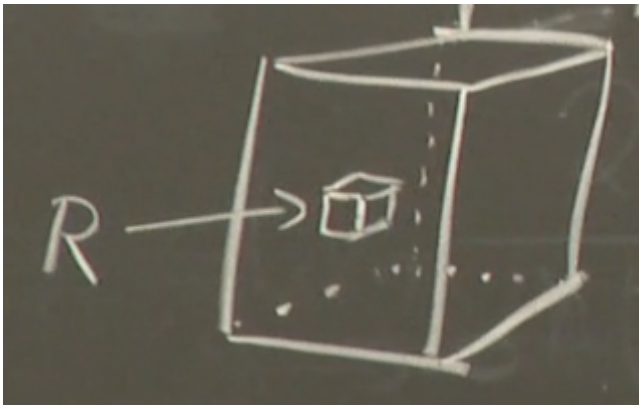
then $PR \approx p(x) * \int_R \partial x$

Here $\int_R \partial x = V_R$ is the "Volume" of R

$\Rightarrow PR = p(x) * V_R = \frac{k_R}{N} * V_R$

N is the number of features that fall in R over # of all features.

That called relative frequency feature in R.



For example, let R be a d-dimensional hypercube and let h denote the side-length of the hypercube$V_R = h^d$

The kernel window function is

$K(x_c, x) = \begin{matrix} 1 \\ 0 \end{matrix}$ if $\frac{x_{i,k} - x_k}{h} \leq \frac{1}{2}$

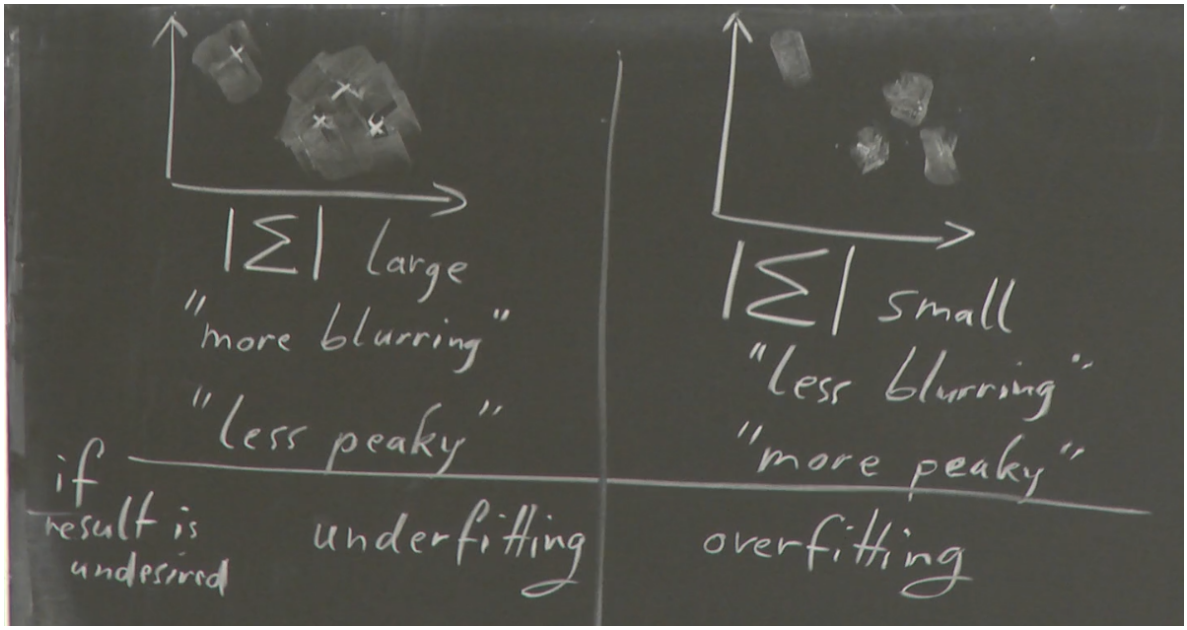
$$\Rightarrow RewriteP(x) = \frac{1}{N} * \sum_{i=1}^{N} K(x_i, x) \tag{2}$$

Alternatively, $K(x_i, x)$ can be any other kernel for example Gaussian.
$K_{\sum}(x_i, x) = \frac{1}{\sqrt{det(\sum) * 2\pi}} * e^{-\frac{1}{2}(x_i - x) \sum^{-1} (x_i - x)}$

where$\sum$ is covariance

Question tree

Question tree: how do we obtain $\sum$?
equivalent: how do we obtain h for the hypercube?
Qualitatively, how doses the result change if $|\sum|$ become larger or smaller?



Estimation of the covariance $\sum$ or the window width h can be done via ML estimation in the case of limited training data additionally with cross-validation

Cross-validation

Let $P^i_{\lambda,N-1}$ be the PDF defined by $S = \frac{S}{x_i}$
$\lambda = \sum, h$

Then we consider the objective function

$$\hat{\lambda} = argmax_\lambda \prod_{i=1}^{N} P^i_{\lambda,N-1}(x_i) \tag{3}$$

$P^i_{\lambda,N-1}$ is "Trained model for all samples except of x_i"
(x_i) is Test sample

$= argmax_\lambda \sum_{i=1}^{N} logP^i_{\lambda,N-1}(x_i)$ log-likelihood
For a differentiable kernel, we can now compute the gradient and look for an optimun, if the kernel is non-differentiable, we have to "brute force" the solution(Note: Gaussian=differentiable, Hypercube=non-diff.)

PA 2018 04

Small addendum to density estimation

Converting the unsupervised Density estimation task to a supervised task(from HTF sec 14.2.4)

$\Rightarrow$ Perform a regression on the joint sample space to model the density (relative to the reference density) (14.2.4)

Mean Shift Algorithm

Method for finding the modes of a PDF.
Applications are essentially applications of clustering:
- data exploration
- image segmentation,denoising,...
- dimensionality/complexity reduction

Let $S = x_1, x_2, \ldots, x_N$ denote N d-dimensional observations x_i. The multivariate kernel estimate is
$p(x) = \frac{1}{N} * \sum_{i=1}^{N} K_\lambda(x_i, x)$
The maxima of p(x) are characterized by the locations x where $\nabla p(x)$ vanishes.

$\nabla p(x) = \nabla \frac{1}{N} * \sum_{i=1}^{N} K_\lambda(x_i, x) = \frac{1}{N} * \sum_{i=1}^{N} \nabla K_\lambda(x_i, x)$

Let us assume that the kernel is radial symmetric
$K_\lambda(x_i, x) = C_d * k_lambda(||x_i - x||_2^2)$
Derive the kernel:
$\frac{\partial k_\lambda(s)}{\partial s} = k_\lambda^,(s)$ derivative of the kernel

$\frac{\partial s}{\partial x} = \frac{(x_i - x)^T (x_i - x)}{x} = 2(x_i - x)$ application of the chain rule

$\nabla p(x) = \frac{1}{N} * \sum_{i=1}^{N} C_d * k_\lambda^,(||x_i - x||_2^2) * (-2(x_i - x)) = 0 \Leftrightarrow \sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2) * x_i - \sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2) * x = 0$

$\Rightarrow \frac{\sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2) * x_i}{\sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2)} - x = 0$ Mean shift equation

To perform gradient ascent, we have to compute the gradient and to "walk upwards " along the gradient direction until the gradient vanishes

Algorithm:
1) compute the mean shift rector
$m(x^{(t)}) = \frac{\sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2) * x_i}{\sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2)} - x = 0$

2) update $x^{(t+1)} = x^{(t)} + m(x^{(t)}) = \frac{\sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2) * x_i}{\sum_{i=1}^{N} k_\lambda^,(||x_i - x||_2^2)}$

Why "MEAN SHIFT"?

If the kernel is chosen as the Epanechnikov kernel, then the update simplifies to computing the weighted "mean" of all feature vectors a sphere around x

Epanechnikov Kernel: $K_E(x) = c * (1 - x^T x)\ if\ x^T x\ \leq\ 1\ otherwise\ 0$

Paper: Comaniciu, Meer:"Mean Shift: A robust Approach Toward Feature Space Analysis"

PA 2018 - 05

Mean shift: Representative of the "Mode-seeking" algorithms = Related Application⇒clustering of Data

K-means:

- Obtain(guess) an initial distribution of cluster centers
- for each data point, identify the closest cluster center
- each cluster center is replaced by the coordinaterweise average of all data point that are closest to it
- repeat until convergence

Greedy algorithm / greedy search

it is a strictly local opination
Biggest advantage: speed
Biggest disadvantage: no brain

Two very common (dis-)similarity criterion for clustering's is the "within" or "intra" cluster distance W(C) ($\Rightarrow Eq.14.28$) and the "between" or "inter"-cluster distance B(C) ($\Rightarrow Eq.14.28$)
$\Rightarrow$ the k-Means algorithm minimizes the within-cluster distances(greedily)($\Rightarrow Eq.14.31 - 14.33$)

How can we determine a reasonable Parameter K?

- low-dimensional dataset: Look at the data
- High-dimensional data set: Need a catamitical criterion
- K-means minimize $W(C)$,
- $W(C)$ decreases for increasing k, for $k = N : W(C) = 0$ "every point is in its own cluster"
- One trick to determine K is Tibshiranis "gap statistics" stop at the k where $G(K) \leq G(k+1) - s_{k+1}^{,}$ where $G(k) = log(w(C_k)) - log(W(C_1)))$ $G(K)$ negative number
- $S_{k+1}^{,}$ is the standard der. of the outcomes for $k+1$ clusters
- A second trick is to create a reference verve from the uniform distribution consider $W(C_k^{uniform})$
- $\Rightarrow$ compute the ratio of $W(C_K))$ over $W(C_K^{uniform})$ and pick the minimum (or an early minimum)

Example for agglomentive clustering

by "Efficient Graph-based Image Segmentation"

PA 2018 - 06

Spectral Clustering:

Step 0:

Given a set of points $S = s_1, s_2, \ldots, s_n$ in R^l

Step 1:

Compute affinities $A \in R^{n \times n}$, $A_{ij} = e^{\frac{-||s_i - s_j||_2^2}{2\sigma^2}}$, $A'_{ij} = 0$

Step 2:

Let $D = diag(\sum_j A_{1j}, \sum_j A_{2,j} \ldots \sum_j A_{n,j}$ of the row affinities, and set $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Step 3:

Compute eigen decomposition of L.
Let $X = (x_1, x_2 \ldots x_k), X \in R^{n \times k}$ denote a matrix consisting of the k "largest eigenvectors" $x_1, \ldots x_k$ of L

k eigen vectors associated with the largest eigenvalues

Step 4:

Normalize X : $Y_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^2}$

Step 5:

Perform k-means clustering with k centers on the points that are the rows of Y

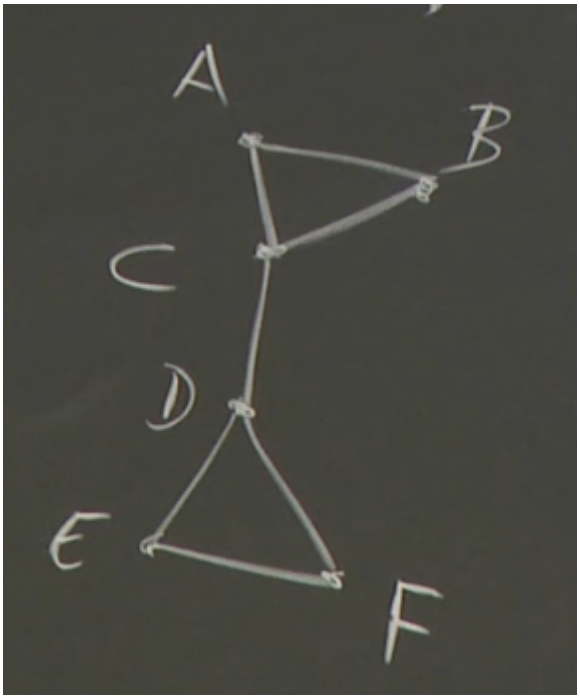
Step 6:

The points in the original space are assigned to the k clusters of their transformed version in %Y%.

Quick, informal note on the Graph Laplacian:

A tool that likes graph theory linear algorithm and probability theory.

Let`s look for example at a graph.



Popular graph algorithm:
Maximum Flow between two nodes?
Shortest Path between two nodes?

The set of mathematical tools that can be used on a graph is somewhat limited,. what exactly should we calculate on a list of nodes an edges ?

$\Rightarrow$ Spectral Graph Theory opens a door to linear algebra by treating a graph`s adjacency matrix as an actual matrix in an algebraic sense $\Rightarrow$ we can for example compute the eigenvalues to find out invariants about he underlying graph.

adjacency matrix: A usefully representation of the adjacency matrix is the Graph Laplacian L.

Remarks on the "Ideal Case" (Sec. 3. 1)

- The set of eigenvalues of a block matrix $\hat{L} = \begin{bmatrix} L^{(11)} & 0 & 0 \\ 0 & L^{(22)} & 0 \\ 0 & 0 & L^{(33)} \end{bmatrix}$ is identical to the set of $eigenvalue(L^{(11)}), eigenvalue(L^{(22)}), eigenvalue(L^{(33)})$(can be treated independently)
- The largest eigenvalue of a graph Laplacian is 1, the seconded largest is lower than 1 $\Rightarrow$ $\hat{L}$ has exactly 3 eigenvalues that are 1. These are the 3 largest ones that we are looking for.
- The associated 3 eigenvector are orthogonal and thus span a 3-dimensional subspaces. Not, however, that the exact choice of eigenvectors is not unique.

The points are clustering around the rotation axes span the subspace These axes are spanned by 90° . angles between them. Therefore. k-means can easily cluster the point in that subspace.

Sec. 3.2 clusters that lie close to another:

The distribution of eigenvalues indicates how well the data structure in to k cluster: consider the eigenvalue gap $\sigma = |\lambda_k - \lambda_{k+1}|$

Cheeger constant:

$$h(S_i) = min_I \frac{\sum_{j \in I, k \notin I} A_{jk}}{min\{\sum_{j \in I} d_j^{(i)}, \sum_{k \notin I} d_k^{(i)}\}}$$

PA 2018 - 07

Manifold Learning:

Goal/Purpose:

Reduce the dimensionality of the data while preserving its structure

Example:

consider a noisy plane in 3-D:

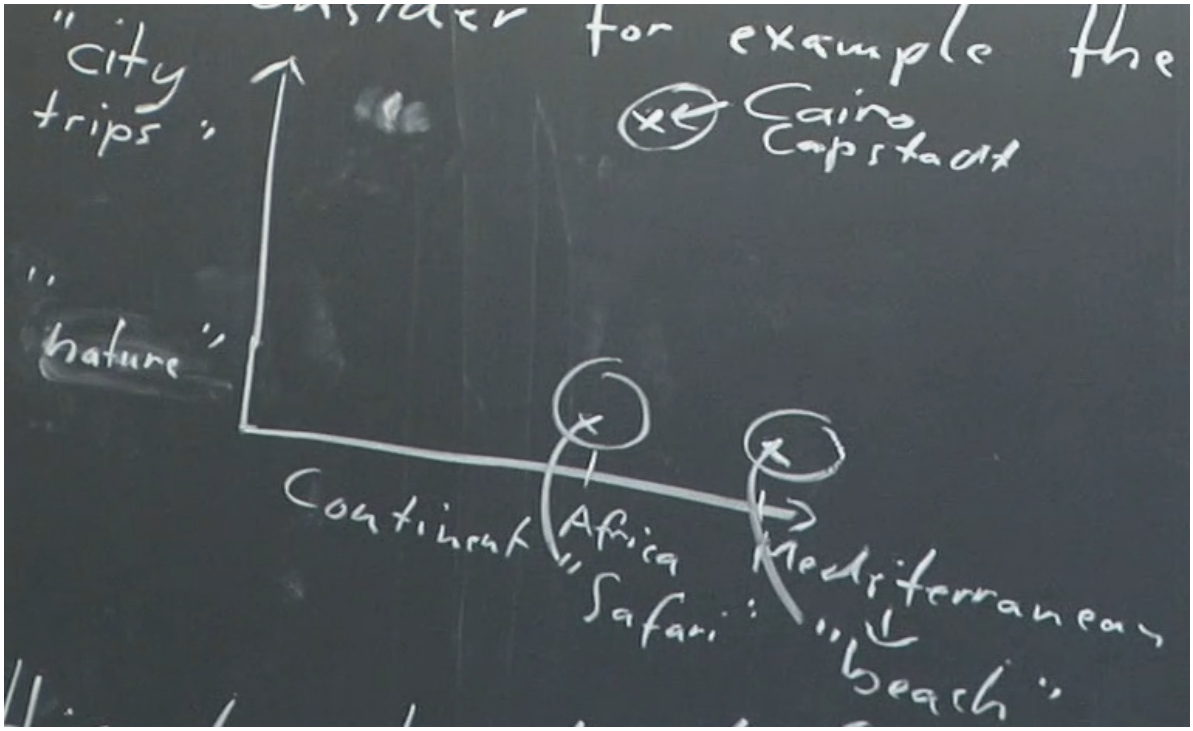
Noisy: unimportant

Plane: is the structure to preserve

###
In many real-world tasks, the structure to preserve, and the low-dim. manifold, is not so super-sharply defined.
Consider for example the manifold of tourist pictures.

Semantic Image Classification

From millions of pixels in millions of images to very few (<10) dimensions $\rightarrow$ rather extreme task.



In this class, I would like to see manifold learning as the task of finding a lower-dimensional representation of structure, essentially by ignoring the noise.

Why should we reduce the dimensionality?

- to avoid the "Curse of Dimensionality"

Curse of Dimensionality :
Distance metrices lead to "wash out" in higher dimensional spaces, i.e. to lose their discriminative power.

To see this, consider N features $x_1, x_2, \dots x_N \in R^d$, where $0 \leq x_{i,k} \leq 1$,where k^{th} component of i^{th} Vector
To capture a fraction r of uniformly distributed features, wee need to consider a fraction $r * V$ of the Volume V of the feature space

This corresponds to a d-dimensional hypercube with edge length

$$e_d(r) = r^{\frac{1}{d}} = \sqrt[d]{r} \tag{1}$$

For example, to find 1% of the features in 10-dim. space,

$$e_{10}(0.01) = (0.01)^{\frac{1}{10}} = 0.63 \tag{2}$$

to find10% in 10-dim. space:$e_{10}(0.1) = (0.1)^{\frac{1}{10}} = 0.8$

the **median** distance of the nearest neighbor to the origin of a d-dimensional space with N sampler is :

$$d(d, N) = (1 - (\frac{1}{2})^{\frac{1}{N}})^{\frac{1}{d}} \tag{3}$$

- d(10,5000) = 0.52

Most of the data points are close to the boundary

PCA (Principal Component Analyses):

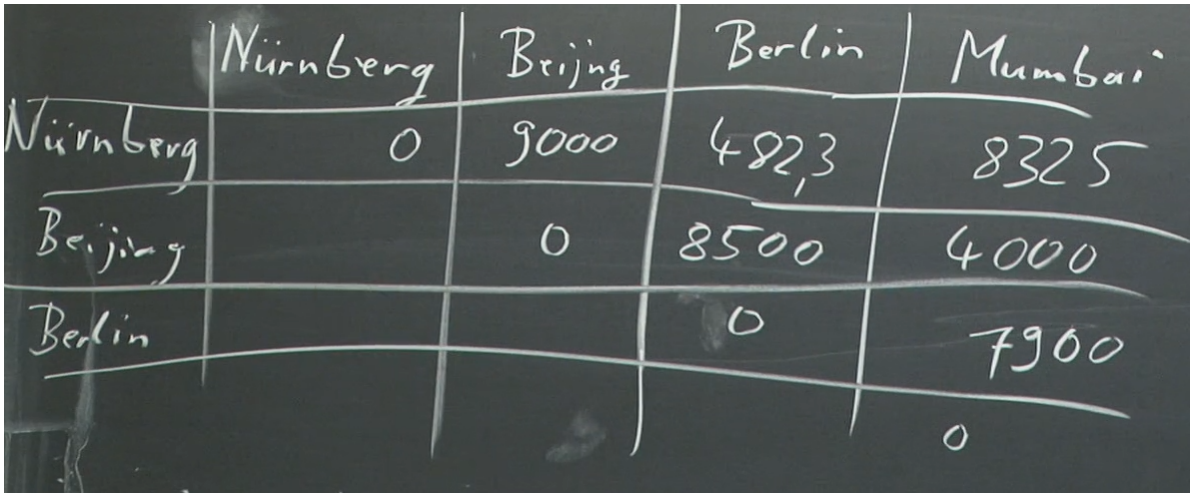
Find a linear mapping $\Phi : R^d -> R^{d1}, d^{'} << d$ that maximizes the variance in each dimension

- objective function: $J = \sum_{i,j=1}^{N} (\Phi * x_i - \Phi * x_j)^T * (\Phi * x_i - \Phi * x_j) + \lambda(\Phi^T \Phi - 1)$
- PCA requires an Eigenvalue decomposition of the covariance matrix PCA generates a low dim orthogonal basis.

Multidimensional Scaling (MDS)

Goal:

To compute a low-dimensional representation of data points where we only know the distances (or dissimilarities) between them.



- can we reconstruct a map from the distances?
  (- note that there are close links to kernel PCA!!!)
  see the elements of statistical learning, search for MDSE

Let $X = (x_1, x_2, x_3) \in R^{d \times N}$ denote the feature vectors (as usual)
Set $B = X^T X$
Let furthermore denote $D^2 = [d_{i,j}^2]_{i,j \in 1....N}$ where

$$d_{i,j}^2 = (x_i - x_j)^T (x_i - x_j) \tag{4}$$

(Euclidean distance, could be exchanged)
Task: given D^2, compute X

$$d_{i,j}^2 = (x_i - x_j)^T (x_i - x_j) = x_i^T x_i + x_j^T x_j - 2 x_i^T x_j \tag{5}$$

Assume that $x_i, \ldots., x_N$ are zero-mean. i.e. $\sum_{i=1}^{N^D} = 0$

The distance matrix is

$$D^2 = diag(X^T x) * l^T + l * diag(X^T X)^T - 2 * X^T X, \tag{6}$$

where $l = (1.///.l)^T \in R^N$
"Magic matrix"/Centering matrix $C = (I - \frac{1}{N} l * l^T)$

- Multiplying D^2 by the centering matrix from left & right and weighting the result by $-\frac{1}{2}$ yields:

$$-\frac{1}{2} C D^2 C = -\frac{1}{2}(I - \frac{1}{N} l * l^T) * (diag(X^T X) l^T) + l * diag(X^T X) - 2 * X^T X) * (I - \frac{1}{N} * l * l^T) \tag{7}$$

Term (1): = 0
Term (2): = 0
Term (3):
$-\frac{1}{2}(I - \frac{1}{N} l * l^T) * -2 * X^T X) * (I - \frac{1}{N} * l * l^T) = I * X^T - \frac{1}{N}(l * l^T) X^T * (X - \frac{1}{N} X * l * l^T) \rightarrow X^T X = B$

- Factorize B to obtain X via SBD:
  $B = U^T \Sigma V$
  B is symmetric $\Rightarrow U^T \Sigma^{\frac{1}{2}} * \Sigma^{\frac{1}{2}} * V$

Note-08

Manifold Learning (continued)

Why manifold learning?

goal: reduce the dimensionality of the data but the structurer of the data is preserved

MDS

"PCA on distances(dissimilarities)"

$$C = (I - \frac{1}{N}1^T 1) \tag{1}$$

$$SVD(\frac{1}{2}C^{-T}D^2C) \Rightarrow U^T\Sigma^{\frac{1}{2}} = X \tag{2}$$

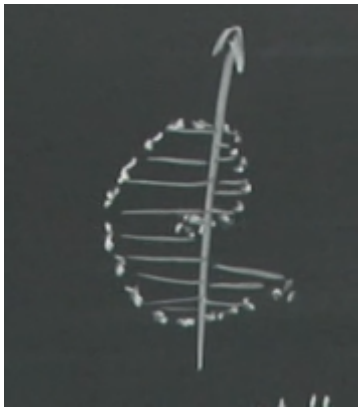
which Σ = zero out singular values for demined

Another item to relax from the PCA formulation:

Can we perform a local projection instead of a global one, Example case (not nicely solvable with PCA)

eg:

"swiss roll"



MDS also has difficulties here!

because the globally applied distances can not distinguish local structures on the manifold

"ISOMAP"(isometric Feature Mapping)

Perform MDS, but on graph distances instead of some global metric like the Euclidean distance.

Graph distance:

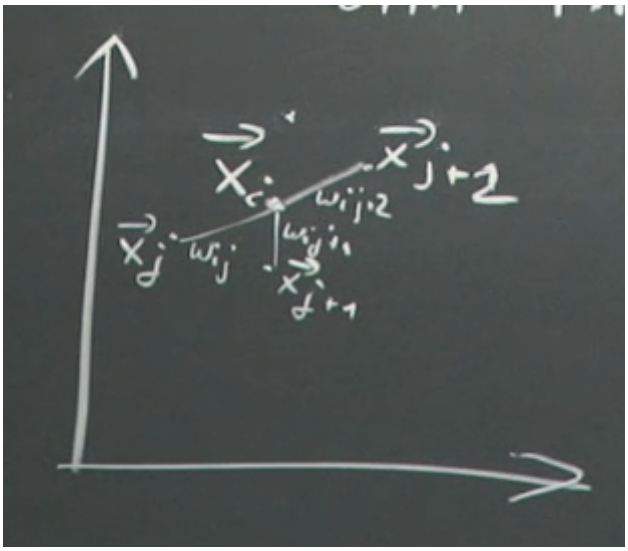
- Define a local neighborhood, compute Euclid distance to these neighbors $\Rightarrow$ Spare adjacency matrix

- Compute all-pairs-shortest paths to obtain all missing distances
  $\Rightarrow$ this is also called a "geodesic" distance

Locally Linear Embedding (LLE)

Idea

- Represent every sample as a linear combination of its neighbors
- Search for a lower dimensional representation with the same / similar linear combination of neighbors



- 1) Search for weight w_{ij}:

$$min\Sigma||x_i - \Sigma_{j\in N(x_i)} w_{ij} x_j||_2^2 \tag{3}$$

subject to $\Sigma_{j\in N(x_i)} w_{ij} = 1$

- 2) In the lower dimensional space:

Solve for $x^{`}$ in

$$min\Sigma||x_i - \Sigma_{j\in N(x_i)} w_{ij} x_j||_2^2 \tag{4}$$

subject to $\frac{1}{N}\Sigma_i x_i x_i^T = I$
covariance matrix is a identity
$\Sigma x_i = 0$

Note: For the linear constraint $\Sigma_{j\in N(x_i)} w_{ij} = 1$, we need a linear programming solver.

$\Rightarrow$ Let`s consider the earlier problem with a quadratic constraint $\Sigma_{j\in N(i) w_{ij}^2} = 1$

$\Rightarrow$ computer the derivative, solve a linear system

Let`s work on the solution of the first part of the problem.
The objective function is invariant to translations:

$$\Sigma_{i=1}^N ||x_i - \Sigma_{j\in N(x_i)}||_2^2 = \Sigma_{i=1}^N ||x_i + t - \Sigma_{j\in N(x_i - t)}||_2^2 \tag{5}$$

for an arbitrary but fixed i, we set $t = -x_i$

$$\Rightarrow ||x_i - x_i - \Sigma_j w_{ij}(x_j - x_i)||_2^2 = ||x_i - x_i - \Sigma_j w_{ij}(x_j - x_i)|| \tag{6}$$

$$||w_{i1}(x_1 - x_i) + w_{i2}(x_2 - x_1) + \ldots ||_2^2 = ||M_i * w_i|| \tag{7}$$

$$\Rightarrow minimize(M_i w_i)^T (M_i w_i) + \lambda * (1 - w_i^T w_i) \tag{8}$$

Note: $M_{ij} = 0$ if $x_j \not\in x_i$

$$\frac{\partial}{\partial w_i}(w_i^T M_i^T M_i w_i + \lambda(1 - w_i^T w_i) = 2M_i^T M_i w_i - \lambda 2 w_i \not= 0 \tag{9}$$

$M_i^T M_i * w_i = \lambda w_i$

Eigenvalue / Eigenvector problem ⇔ take eigenvector that belongs to smallest non=zero eigenvalue

2 step: solve second object function with the calculated weights w_i for x_i (solution is similar)

Laplacian Eigenmaps

- Build an adjacency graph form the feature neighborhood
- Compute affinities between neighbored noted(weight)
- Perform eigen decomposition of the Graph Laplacian of the weights
- Low-dimension embedding

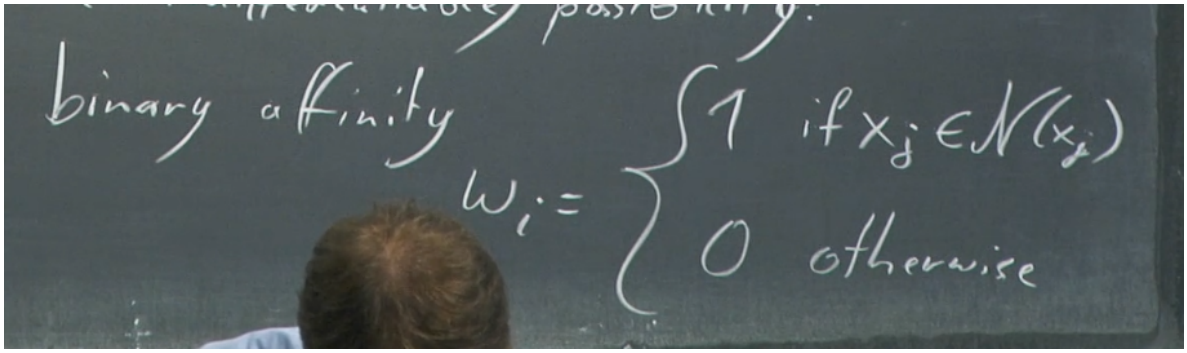
Step 1:

- Everything with in field (Euclidean?) distance d is a neighbor
- The k nearest samples are neighbors

Step 2：Pick any function that decreases with increasing distance 1 specific proposal heat kernel

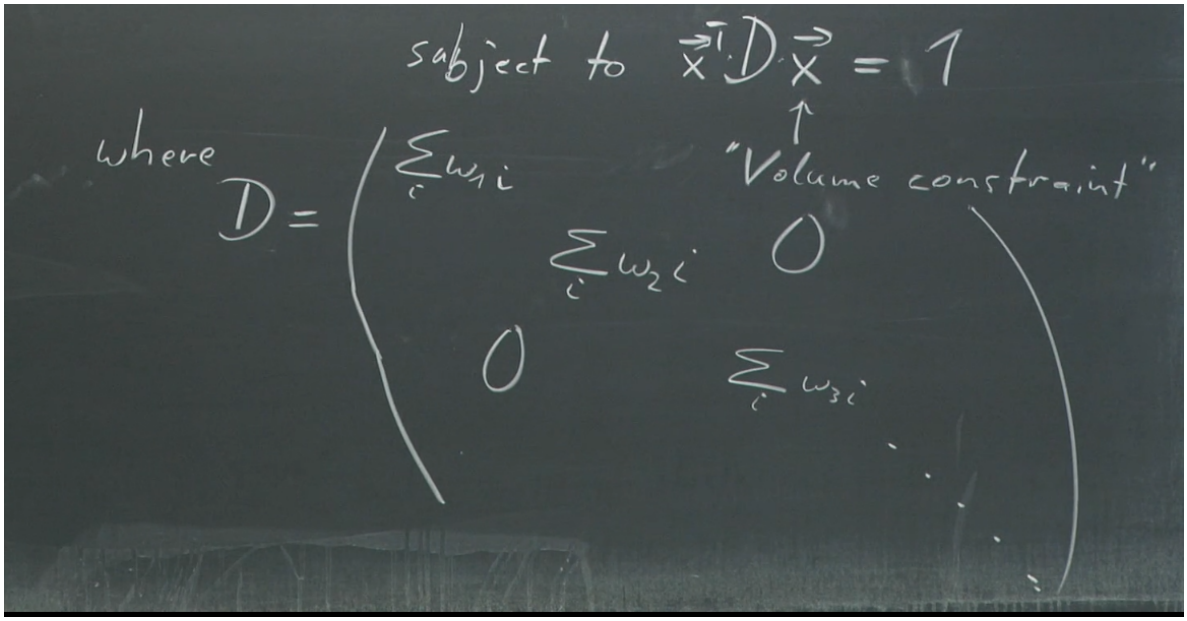
$$w_i = e^{\frac{-||x_i - x_j||_2^2}{t}} \tag{10}$$

another (non-differentiable possibility):



Step 3: theoretical derivation

minimize $\Sigma_{i=1}^N \Sigma_{j=1}^N (x_i^{'} - x_j^{'})^2 w_{ij}$



Rewrite the objective function:

$$\Sigma_{ij}(x_i' - x_j')^2 w_{ij} = \Sigma_{ij}(x_i'^2 - x_j'^2 - 2x_j' x_j')w_{ij} \tag{11}$$

$$= \Sigma_{ij}(x_i'^2 w_{ij} + \Sigma_{ij}(x_j'^2 w_{ij} - 2\Sigma_{ij} x_i' x_j' w_{ij} \tag{12}$$

$$= 2\Sigma_{ij} x_i'^2 w_{ij} - 2\Sigma_{ij} x_i' x_j' w_{ij} \tag{13}$$

$$= 2(x'^T D x' - x' W_{x_i'}) \tag{14}$$

$$= 2x'^T (D - W)x \tag{15}$$

Minimize $x'^T L x'$ subject to $x'^T D x' = 1$

$$\Rightarrow \frac{\partial}{\partial x}(x'^T L x' + \lambda(1 - x'^T D x') \tag{16}$$

$$= 2Lx' - \lambda D x' \neq 0 \tag{17}$$

$$\Rightarrow D^{-1} L x' = \lambda x' \tag{18}$$

Note-09

$$p(x) = \sum_{i=1}^{N} \beta \mathbb{N}(\mu; \Sigma; x) \tag{1}$$

β is the weighting term

$\mathbb{N}$ is the single Gaussian

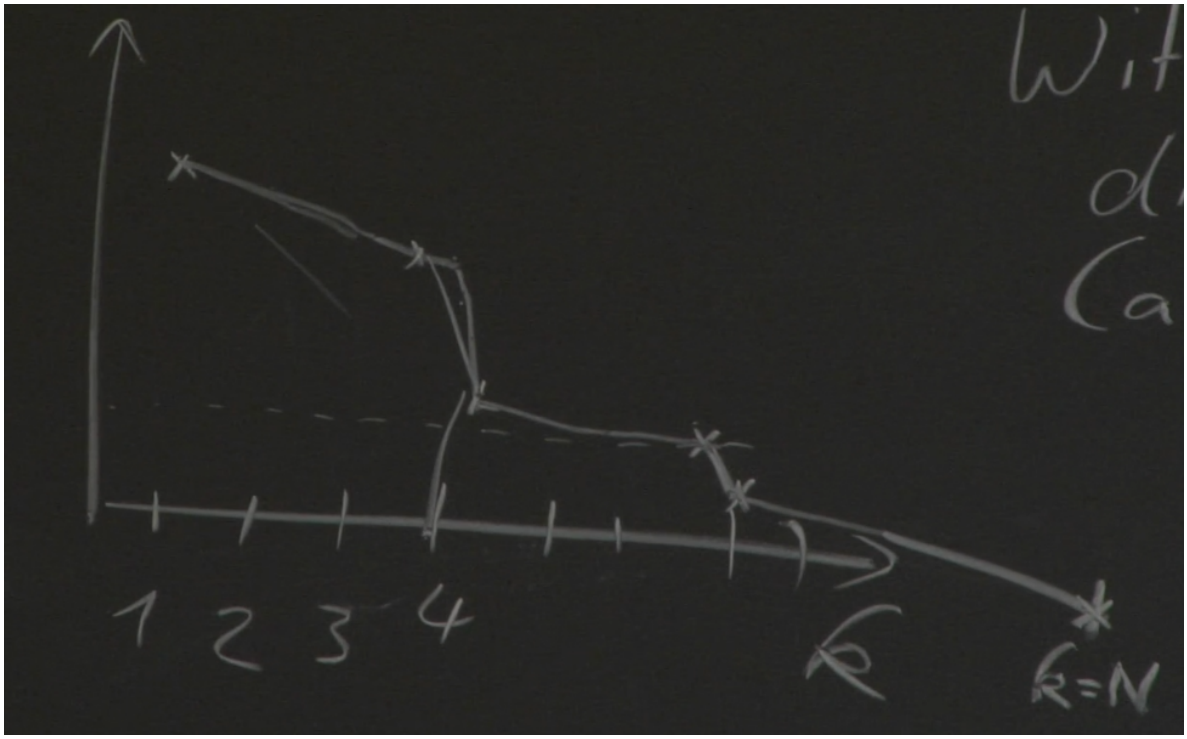
μ, Σ is the parameters that need to be filled

N is the Model selection problem: choose N

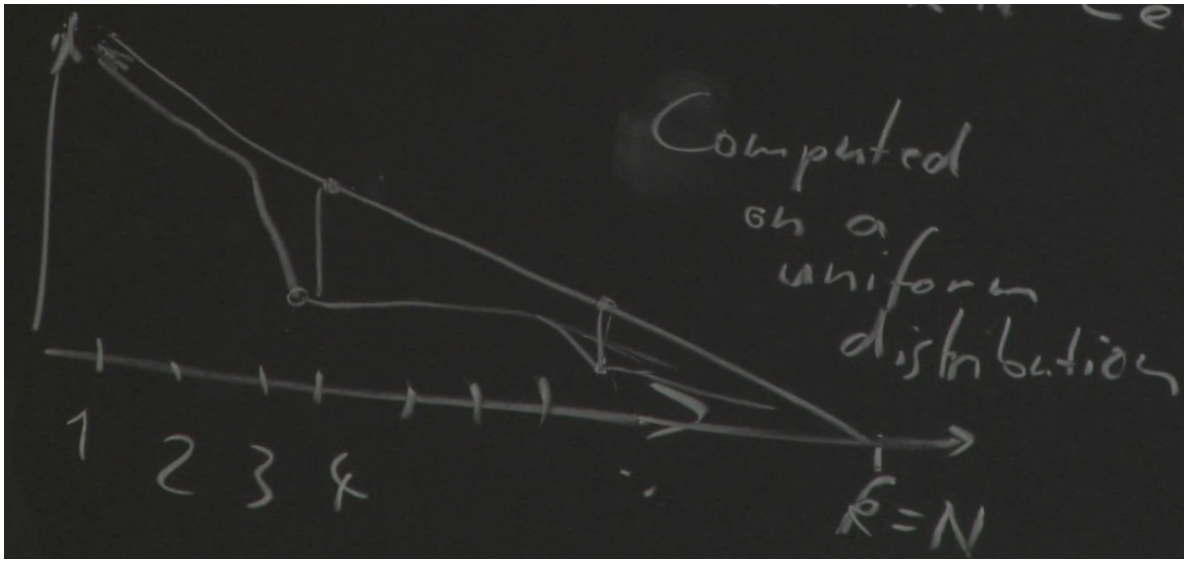
For clustering - Let's say k-means clustering - how did we tackle the model selection problem (i.e. select the number of clusters)

Gap statistics:

Within - cluster distance (average. distance. of each point to its cluster center)

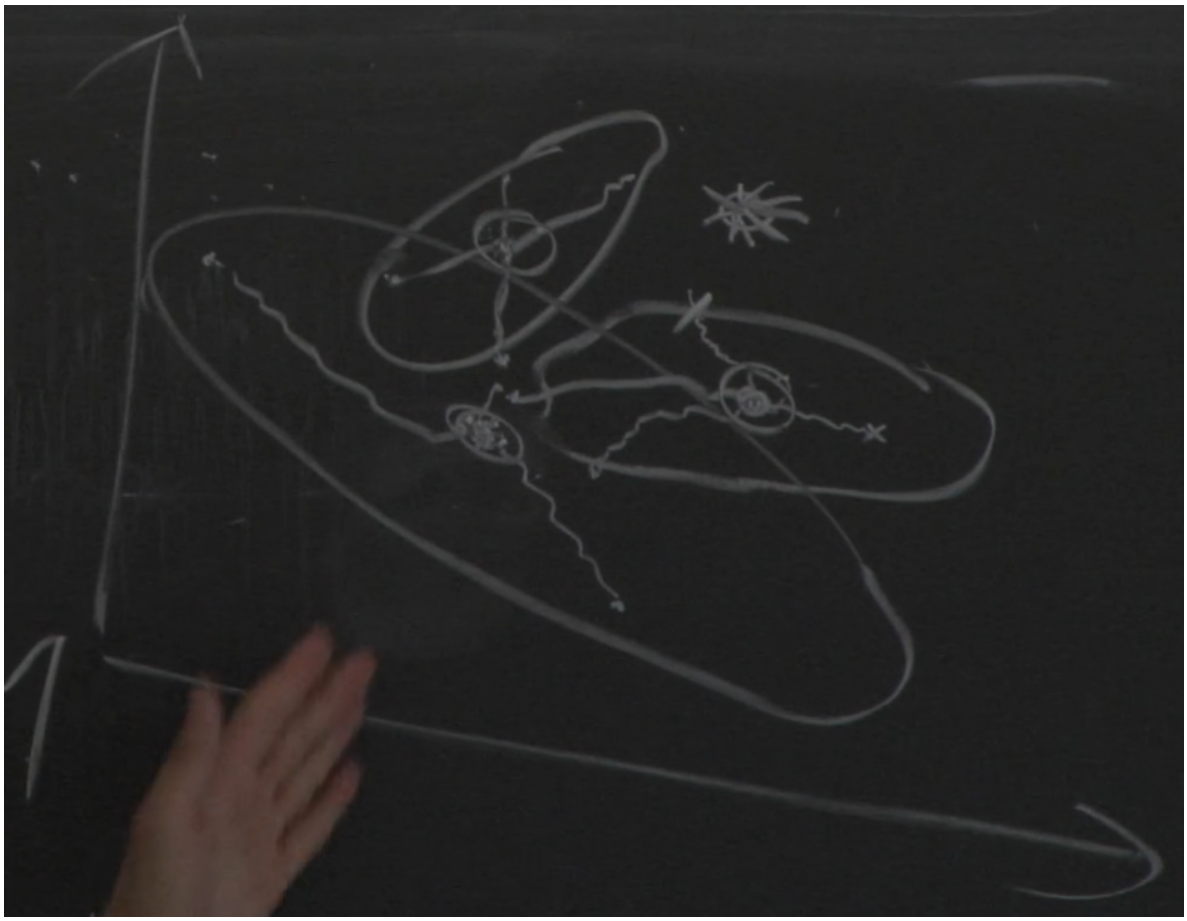


Normal gap statistics



Computer on a uniform distribution

Mean shift:



Cross-Validation can be used to determine the quality of a chosen Parzen window sized for density estimation

Remove 1 sample from the input set. For a reasonable set of window sizes $s = \lambda_1, \lambda_2, \lambda_3 \dots \lambda_w$,evaluate $1 - p_{\lambda_i x)j}(x) \leftarrow p(x)$,computed for window λ_i,and without

$\Rightarrow$ repeat this for all samples $x_1, .. x_N$

$\Rightarrow$ Choose the window size λ_i with the smallest interpolation error

Model Selection Problem for Manifold Learning

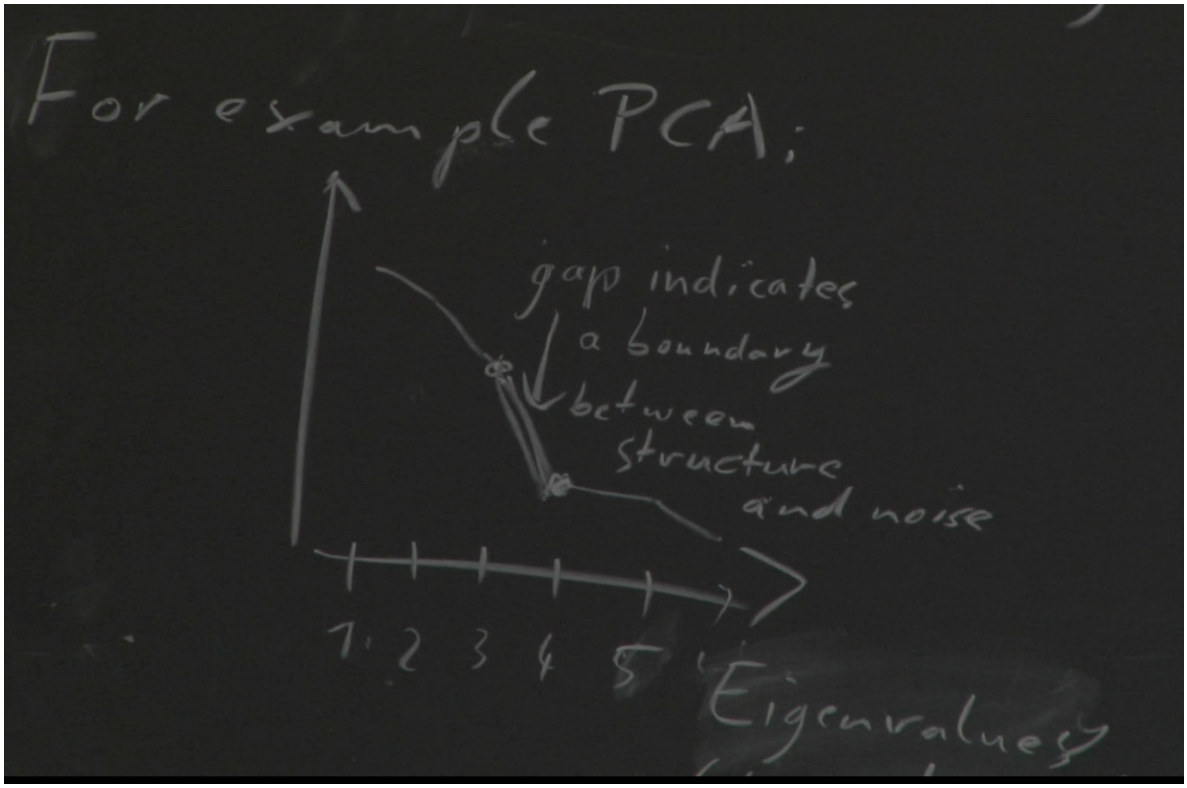
What is a good (intensive!) dimensionality of the data?

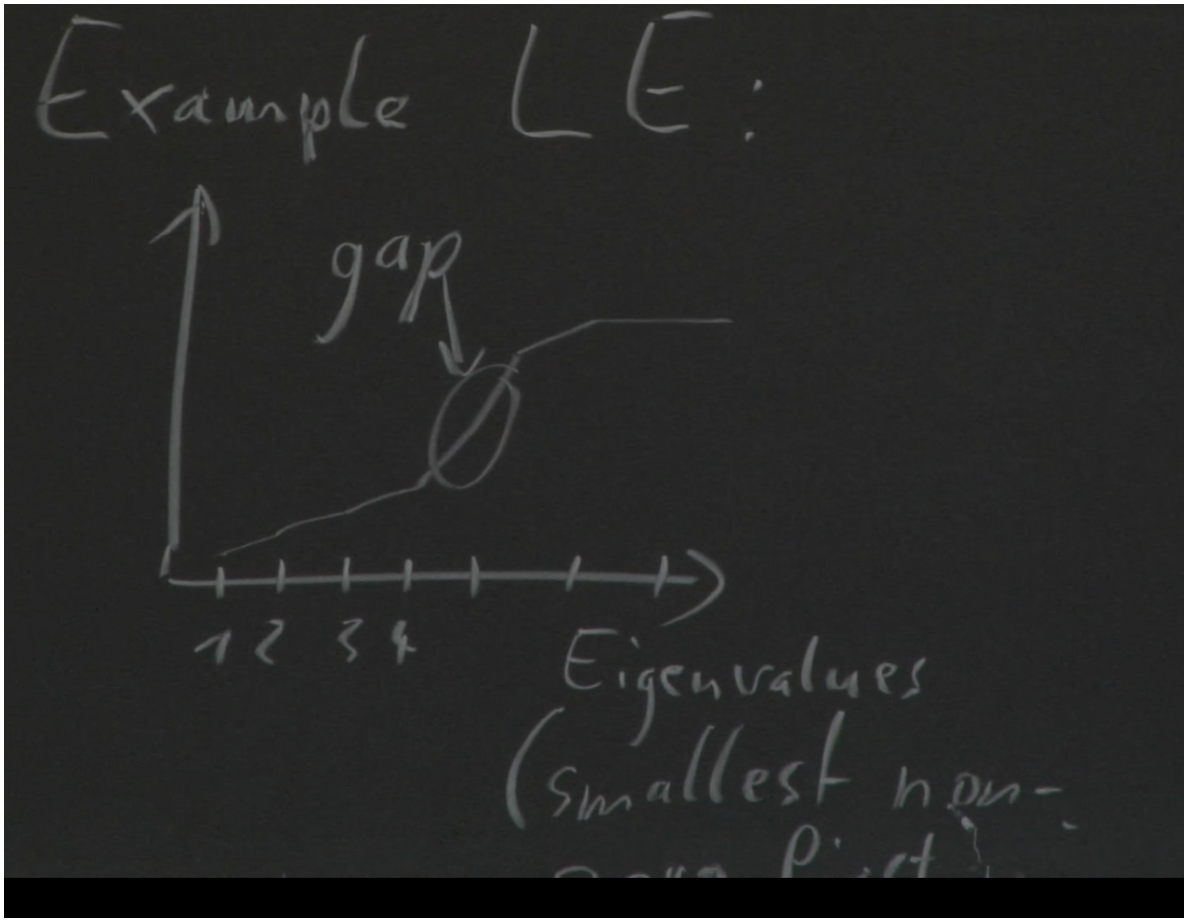
$\Rightarrow$ All methods that we considered are based on some Eigencomposition.

The distribution of eigenvalues indicates (quantifies) the quality of the model similar to the within-cluster distance for the k-means clustering.

A "good" dimensionality is indicated by a gap in the distribution of eigenvalues.

For example PCA:





Random Forest

Random Forest as a learning-based tool for feature space partitioning

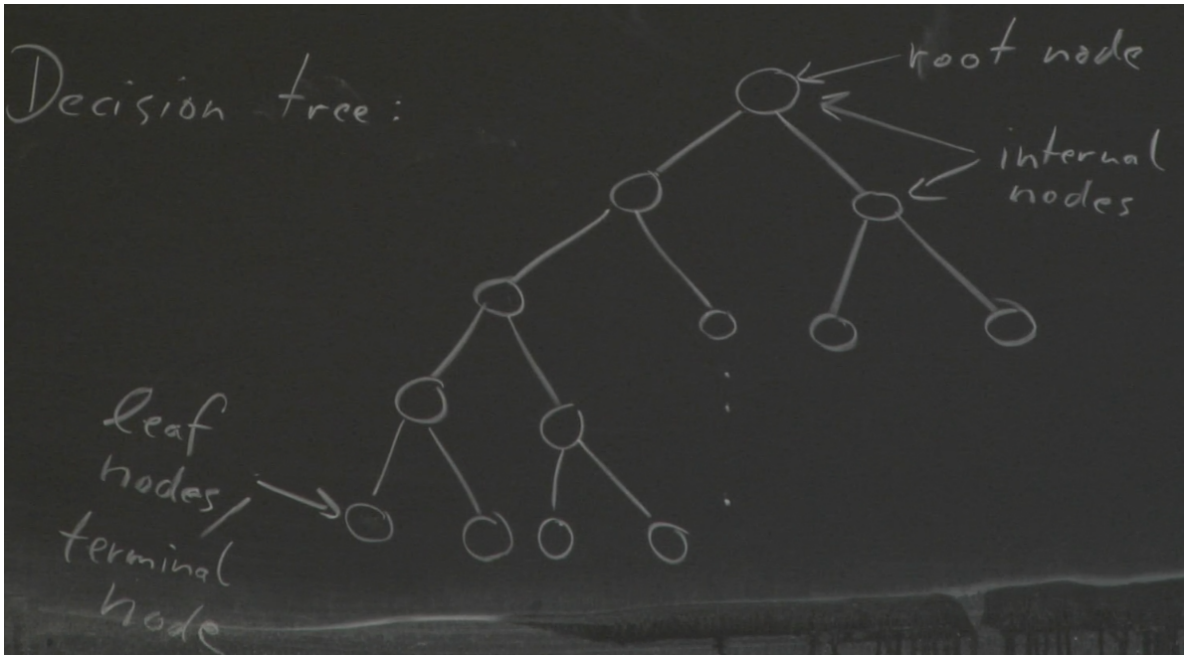
You can use your partitioning feature space form many different tasks' specifically

- classification
- regression
- density estimation
- manifold learning
- semi-supervised learning

Forest: an ensemble of trees

Decition forest: ensemble of decision trees

Decition tree: binary tree



General workflow:

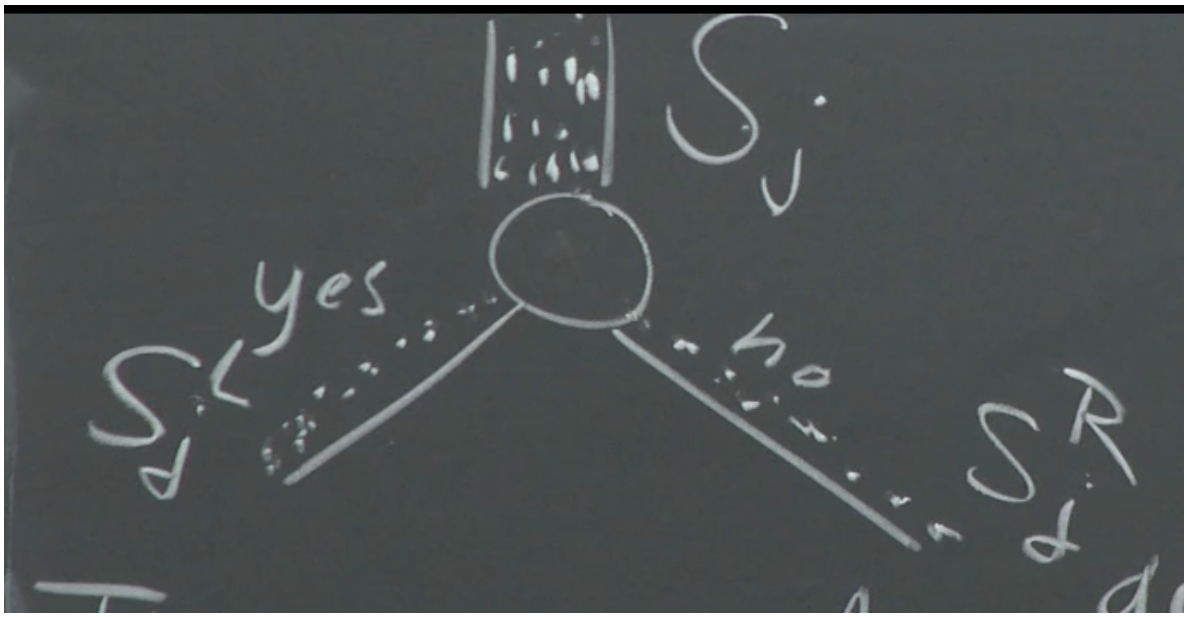
- start at root node
- Answer for each internal node with y or n to a question
- For y, proceed to left successor/child, for n processer to right successor
- Terminal node rise the result

Note-10

Testing: for a sample x, apply question at root node, follow up on left or right successor depending on whether the answer is yes or no

use this node as new root node repeat until leaf is reached

Training: Determine the functions ("yes no question") at each internal node, and the leaf output.



The question for a good node function is: what is a good "splitting functions" for all incoming features?

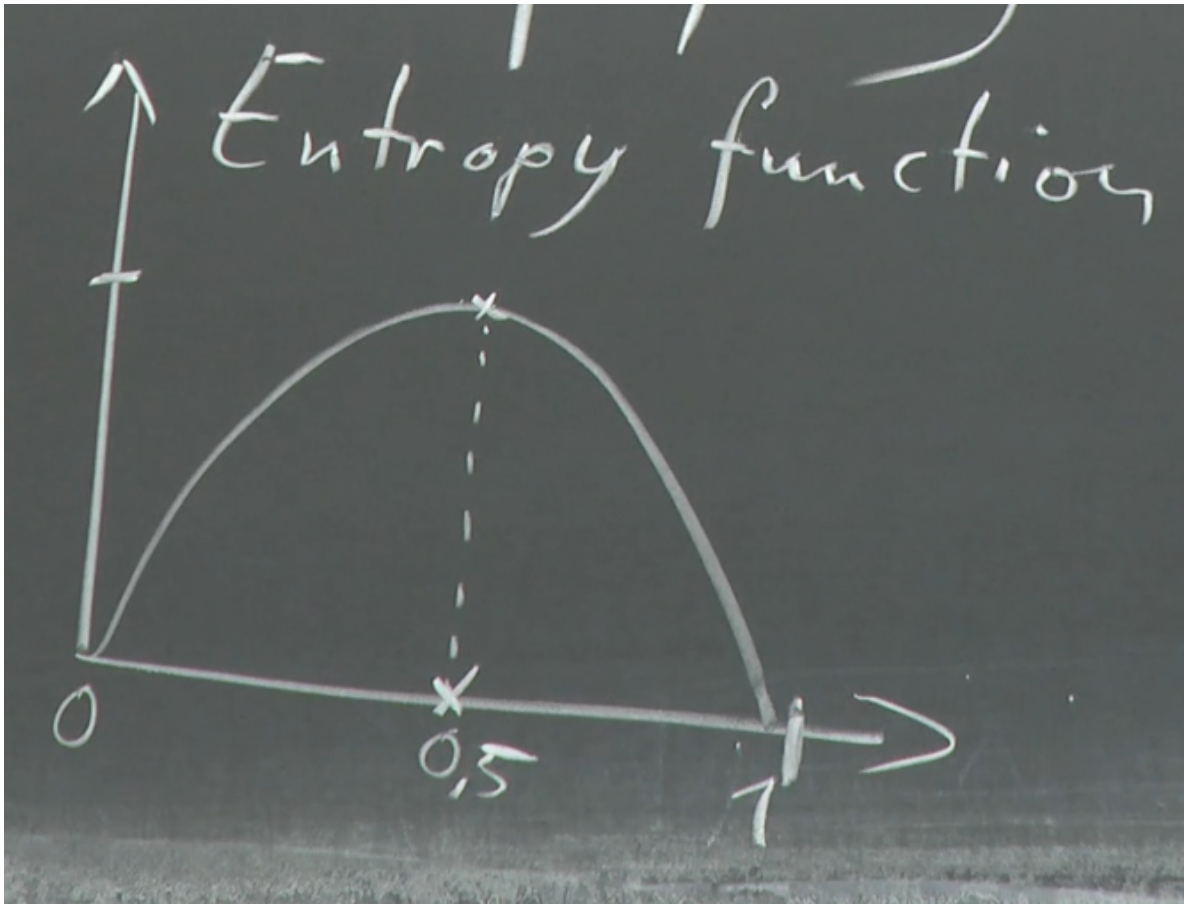
$\Rightarrow$ Maximize the information Gain for our tasks

$$I(S_j) = H(S_j) - \sum_{i \in L,R} \frac{S_j^i}{S_j} H(S_j^i) \tag{1}$$

Where $H(S_j)$ denotes the entropy of set S_j, For example , for the case of classification

$$H(S_j) = -\sum_{c \in C} p(c) * log(p(c)) \tag{2}$$

Where C is the set of all class labels, and p denote the relative frequency of label s in set S_j



The binary entropy function is lowest if the input is always 0 or always 1 "not surprising"

Translated to our classification task the entropy is lowest if all class labels are identical in a subset S_j^{LorR}

$\Rightarrow$ Lower entropy in S_j^{LorR} $\Rightarrow$ larger information gain

Generally speaking, for classification, we are looking for splitting functions that separate our labeled training data as cleanly as possible.

Plain decision trees are quite prone to overfitting (consider a greedy selection of always the best information gain)

$\Rightarrow$ Breinman's idea: add randomization information in the train process, This leads to a "weak learner" $\Rightarrow$ a tree that is not optimal, but "quite ok"/better than learnings

Repeat this a couple of 100s or 1000s time to obtain a forest of weak decision tree. Average their weak opinions $\Rightarrow$ if the weak learner are statistically independent, the ensemble vote converges to the true answer.

Options for randomization:

Randomization is typically set for one decision tree, We can randomize:

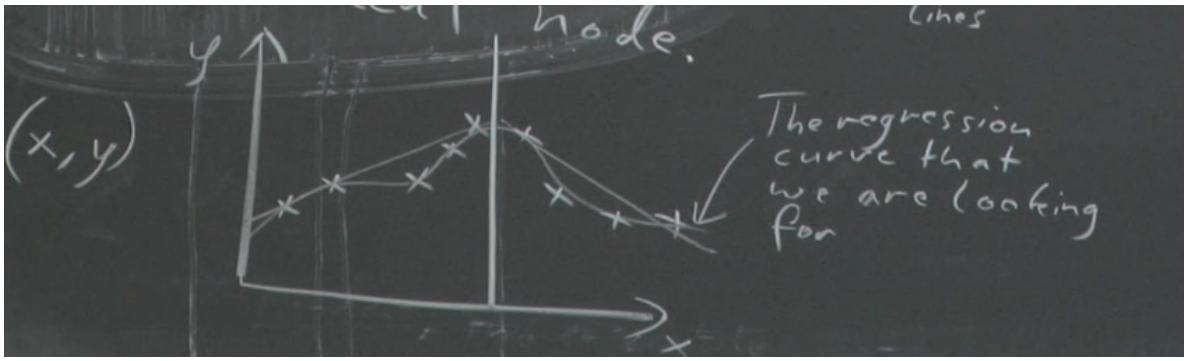
- The subset of samples for training the tree ("bagging")
- The subset of feature dimensions for the decision functions
- The parameters of the decision function

Note-11

Random Forest

"Regression Forest"

- Redefine information gain for continuous labels y
- Fit a local curve model to the samples in leaf node.



$\Rightarrow$ Again, by averaging multiple trees, the discontinuities smooth out, and the final regression curve is "smooth enough" (for our numerical needs)

Information gain for regression: $I(S_j) = H(S_j) - \sum_{i \in L,R} \frac{S_j^i}{S_j} H(S_j^i)$

$$H(S) = -\frac{1}{|S|} \sum_{x \in S} \int_y p(y|x) * log(p(y|x)) dy \tag{1}$$

Let $p(y|x)$ be a Gaussian distribution:

$$p(y|x) = N(y; y(x); G_y^2(x)) \tag{2}$$

$\Rightarrow$ At each position x, we have a distribution of possible values (see Bishop's book on how to perform a probabilistic line)

"Density Forest"

$\Rightarrow$ use the Random Forest splitting mechanism to estimate a PDF.

- Idea: Fit to the samples in each leaf node a Gaussian function The Gaussian is bounded at the boundaries of leaf, and has discontinuities the there.
  $\Rightarrow$ Average over many trees to smooth out the discontinuities

Adjust the objective function: $I(S_j)$ is again identical to classification

Entropy $H(S)$ for a multivariable Gaussian distribution

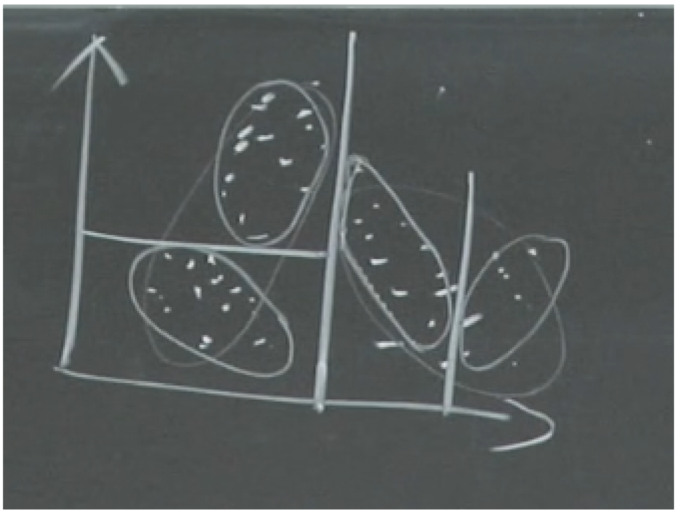
$$H(S) = \frac{1}{2} * log((2\pi e)^d) * |\Lambda(S)| \tag{3}$$

where d is the dimension of the Gaussian, and Λ its covariances,
$\Rightarrow$ to evaluate $H(S)$, you effectively fit a Gaussian to the sample in S.

remark: We can plug $H(S)$ into $I(S_j)$ and obtain

$$I(S_j) = log(|\Lambda(S_j)| - \sum_{i \in L,R} \frac{|S_j^i|}{S_j} * log(|\Lambda(S_j^i)|)) \quad (4)$$

⇒ Goal" Split to obtain a nice, compote Gaussian on side



"Manifold Forest"

Goal: Manifold Learning

Idea: Fit a density forest to the data use the "uncertainty" of the splitting boundaries across the trees to determine how tightly two data points are coupled.

Apply Laplacian Eigenmaps on these affinities

Affinity model:

Let $l(v)$ denote the leaf partition function that assigns each feature vector to a leaf.

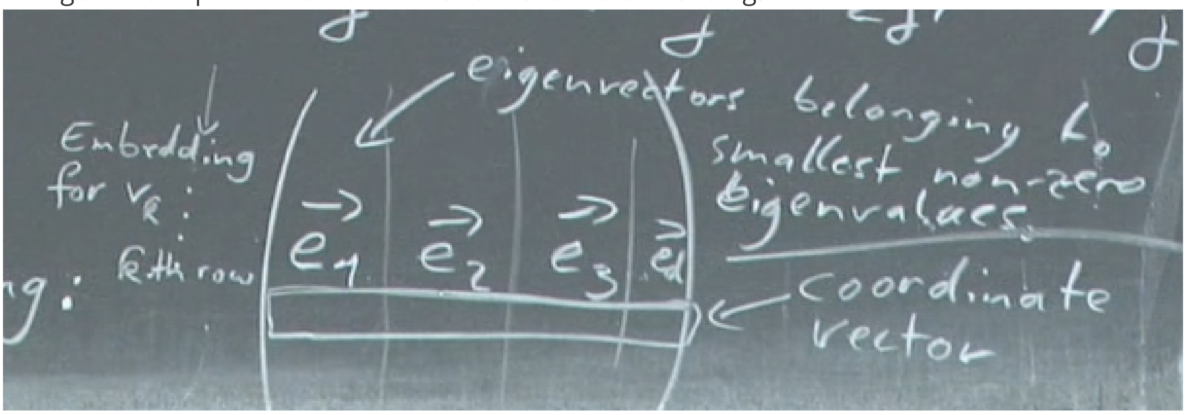
let k denote the number of training samples, the affinity in tree t between points v_i and v_j is

$W_{ij}^t = e^{-Q^t(v_i,v_j)}$ $$, Where Q^t is a distance function Choices for Q : −Mahalanobis affinity :![image − 20210302014312931](C : \Users\Administrator\AppData\Roaming\Typora\typora − user − images − 20210302014312931.png) − Gaussian affinity − Binary affinity :![image − 20210302014351160](C : \Users\Administrator\AppData\Roaming\Typora\typora − user − images − 20210302014351160.png) − Final affinity : $W = \frac{1}{T} * \sum_{t=1}^{T} W^t$ $$ ⇒ Compute Laplacian Eigenmaps on W (5)

L = I- \Gamma{-\frac{1}{2}}W\Gamma{-\frac{1}{2}}
$$

where L is Graph Laplacian, $\Gamma = diag(\sum_j W_{1j}, \sum_j W_{2j}, ...\sum_j W_{Nj},)$

⇒ Eigen decomposition Obtain Lower-dimensions embedding:



Note-12

Random Forests: A flexible tool to describe sample neighborhoods

Single tree: Performs a partitioning

Ensemble of tree "often" the partitioning

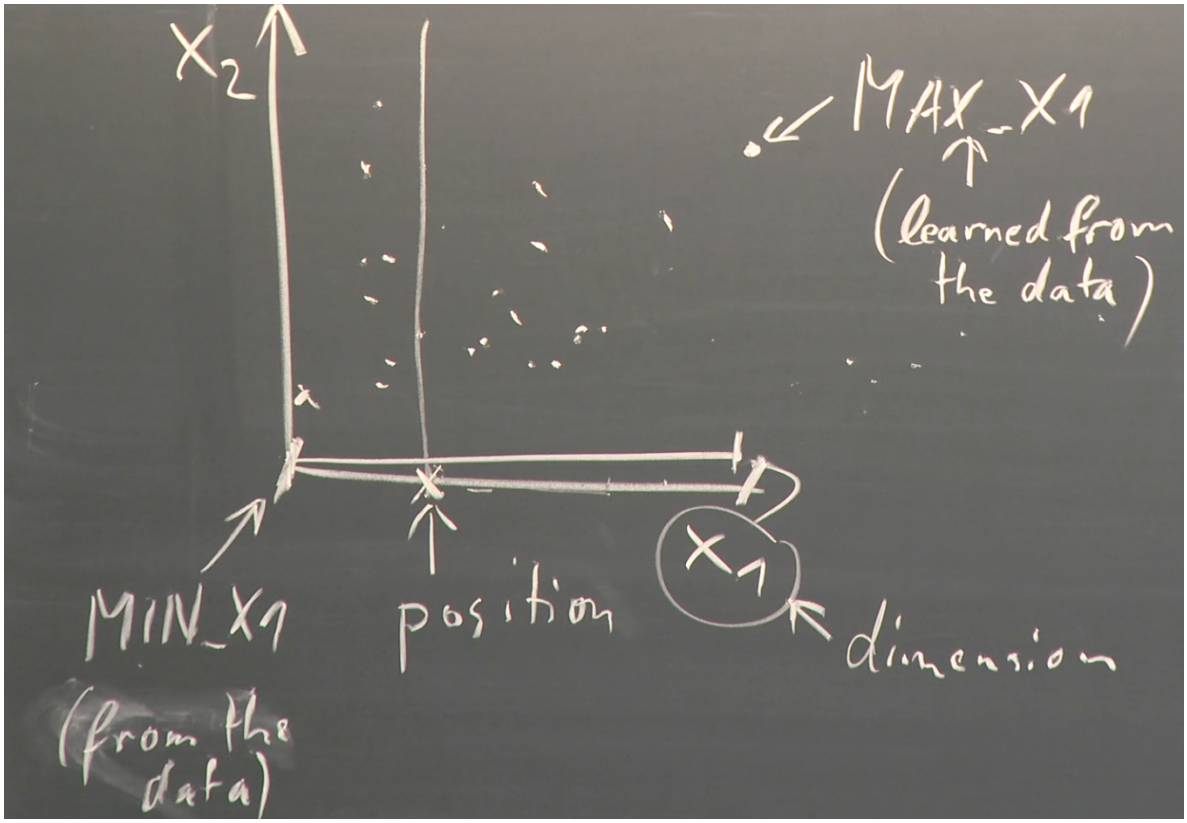
from 0-1 labels to continues probability

Learned from the data, depending on the tree/forest parameters

- Depth of a tree
- Number of tree
- early abortion criteria where growing the tree(not discussed)
- Randomization parameters: # of splitting functions to test % of samples to select # of features dimensions to select
- Parametric form/type of the splitting set

Tree depth and # of trees are related when it comes to overfitting / underfitting

Generally, deeper trees need also larger forests to counter overfitting



Sample decision functions:

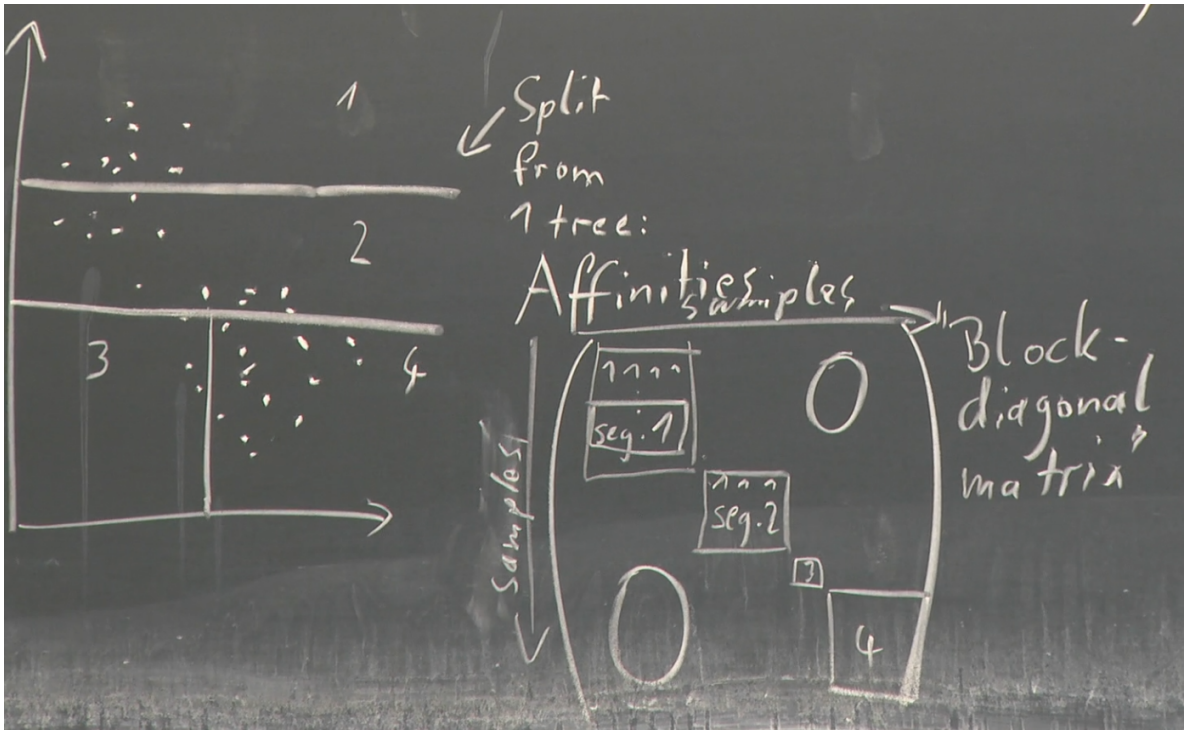
Randomly draw uniformly distributed values for "dimensions" and "Positions" within their respective value range.

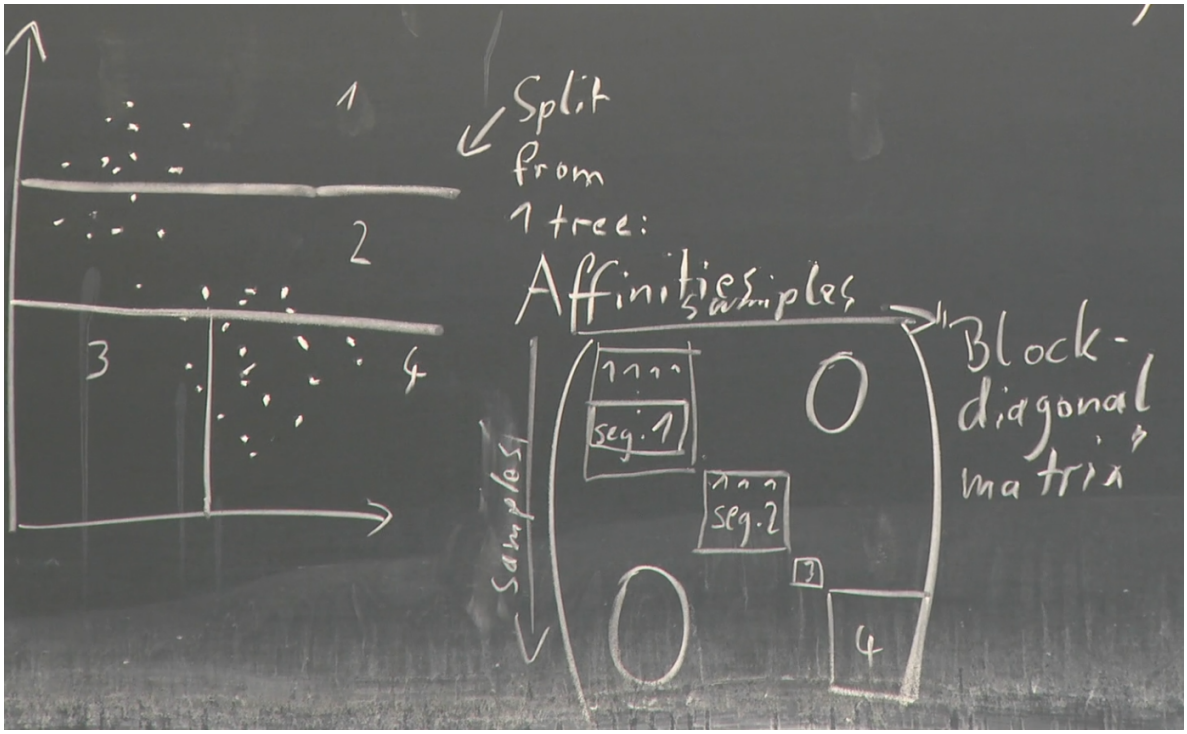
How many decision functions do we need to test in total for training a single tree?

Height H, no early stopping

A brief sketch on the Manifold Forest:

A manifold forest is Laplacian Eigenmaps that use the partition function of a density forest to define its affinity





The "loosely packed" links can be stretcher a little bit when looking for a lower dimensional manifold.

Nice: the "rigidness" is deduced from the variance in the tree training

Note-13

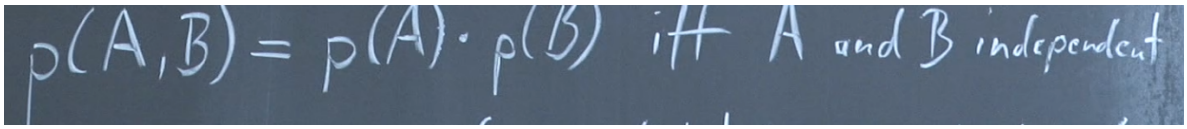
Probabilities Graphical Models

Hidden Markov models

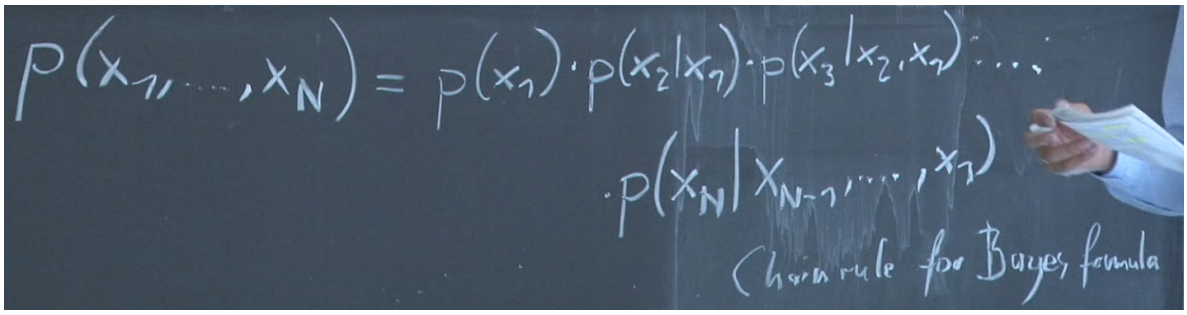
Markov Random Field

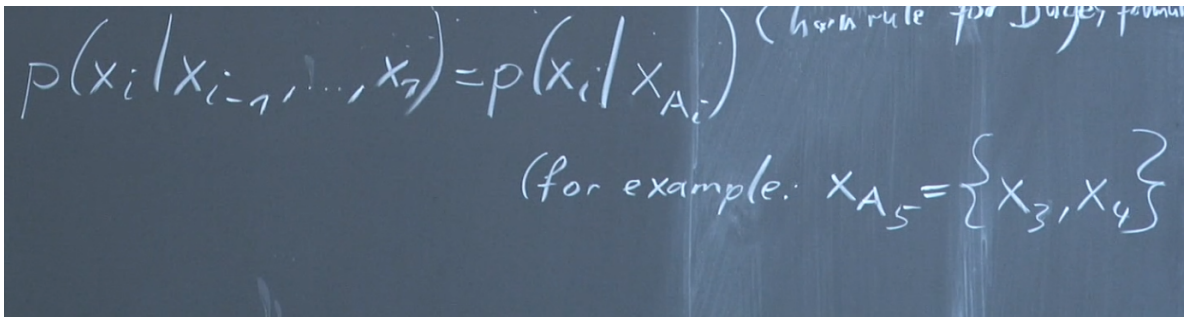
Introduction
Goal: to effectively and efficiently work with a huge joint probability function $p(x_1, x_2, x_3 \ldots . x_n,)$
A head-on model for $p(x_1, x_2, x_3 \ldots . x_n,)$ is almost certainly intractable, consider for example the space of all spoken sentons on the space of all pictures.

$\Rightarrow$ A key component in a probabilistic model are the independence assumptions.



A compact Bayesian network is a distribution in which each factor depends only on a small number of "ancestor variables"





Graphical representation:

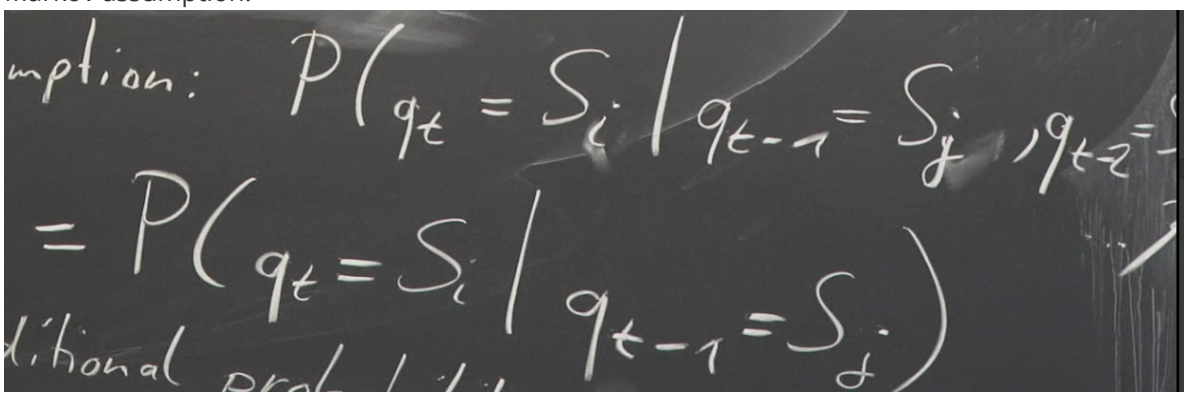
Express a conditional probability a directed edges in a graph where the variables $x_1, x_2 \ldots x_N$ are the node

Note-14

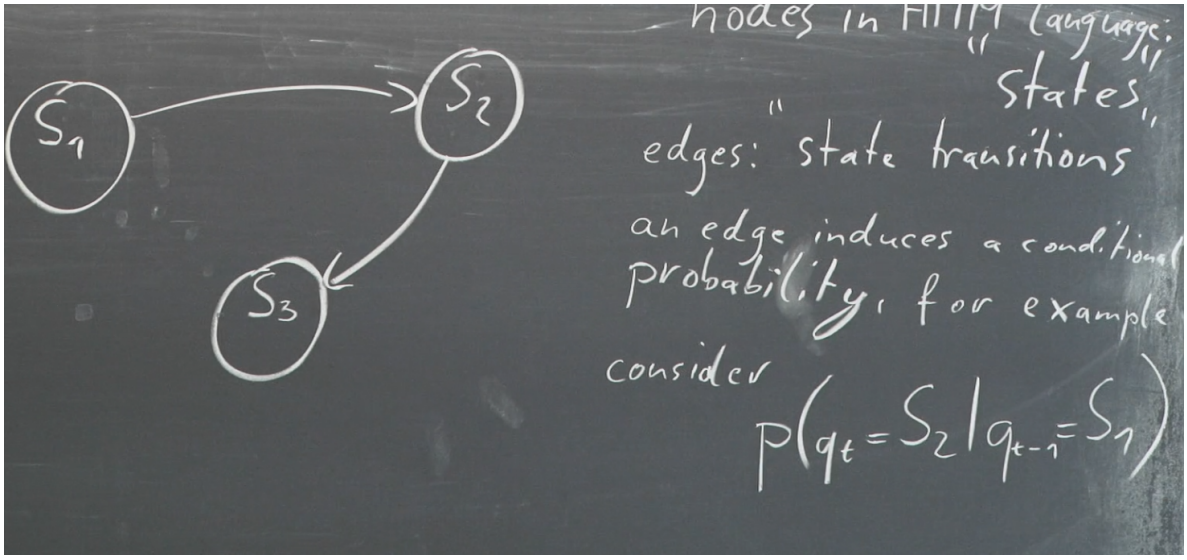
Hidden Markov Model

Probabilistic graphical model, The graph is directed

Markov assumption:



each conditional probability consists of only one ancestor



Most popular application: Speech processing and More general: sequential data.

Overall application: Ticket booking system

Construct a dialogue ⇒ prior for the confidence of next sentons

"Hidden" is the actual state: We only observe the phonemes (or more generally the output symbols)m and are interested in the question how well a HMM matches that sequence, but we don't care about the exact visited states

Markov property: Joint probabilities can be factorized into s terms, because every probability conditioned on states depends only on the last state

Note-15

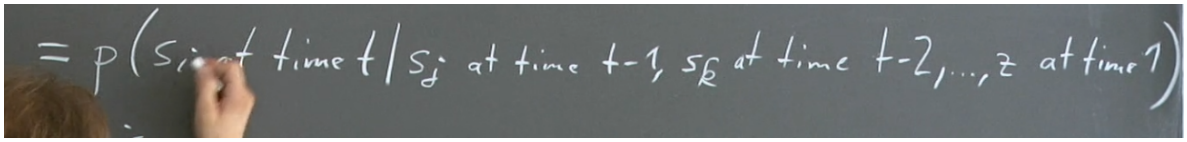
HMM

Given: Sequence of samples/observations

Desired goal: model for representing , for example to compute a probability that describes a certain label

Approach: Use a probabilistic model, expressed in a directed graph

The Markov property implies that the



that is, the current state only depends on the previous state, and not on the whole previous path, $\Rightarrow$ this makes inference tractable!

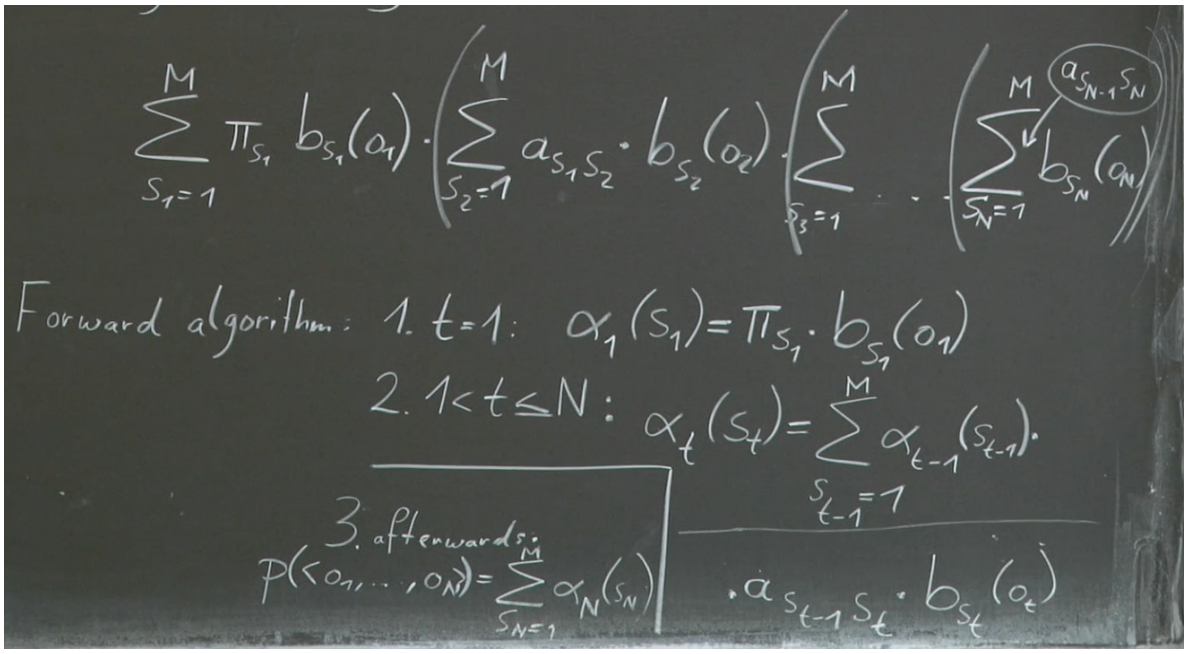
3 Methods to work with HMMs:

1) Matching: find $p(o_1, o_2 \dots o_N|\lambda)$: Forward algorithm Backward algorithm

2) Most likely state sequence for given observation:

3) Training: find $\lambda = (A, B, \pi)$ given some training samples Baum-Welch-Formula

Naïve approach is to expensive(exponential$O(M^N)$), but there is a Dynamic Programming solution.



Backward algorithm:

1.) $\beta_N = 1$

2.) $1 < t < N$: $\beta_t(s_t) = \sum_{s_{t+1}=1}^{M} a_{s_t s_{t+1}} \cdot b_{s_{t+1}}(o_{s_{t+1}}) \cdot \beta_{t+1}(s_{t+1})$

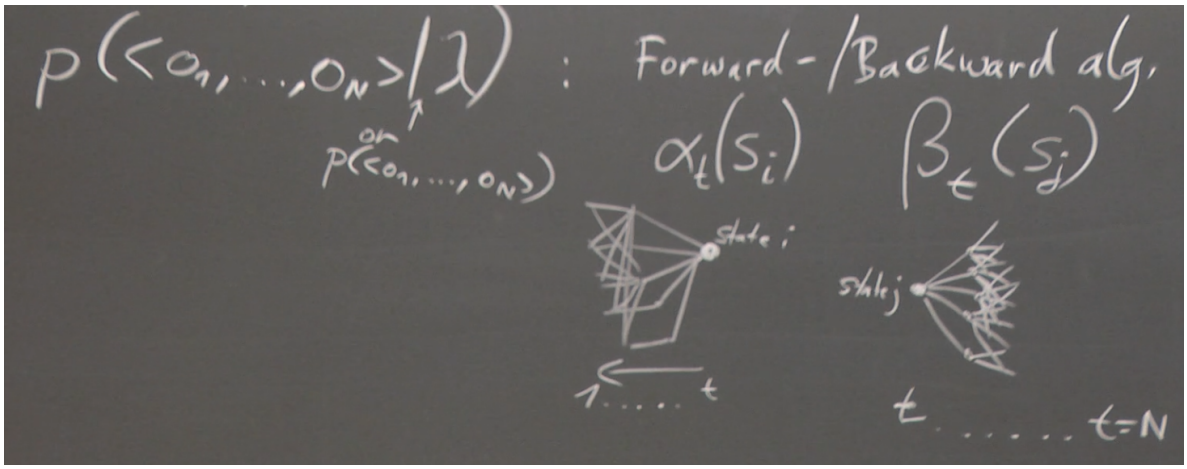
3.) $t = 1$: $\beta_1(s_t) = \sum_{s_{t+1}=1}^{M} \pi_{s_1} b_{s_1}(o_1) \cdot \beta_t$

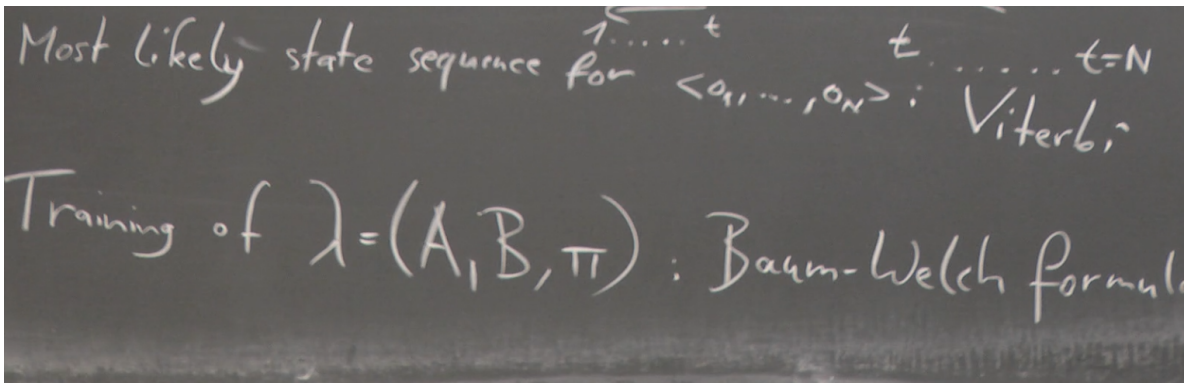
Note-16

HMM (Continued)

Three Methods:

1) Solution to Task 1: Forward-/Backward algorithm





2) Solution to Task 2: Viterbi algorithm:

Most likely state sequence:

Method is identical to Forward algorithm, with two except

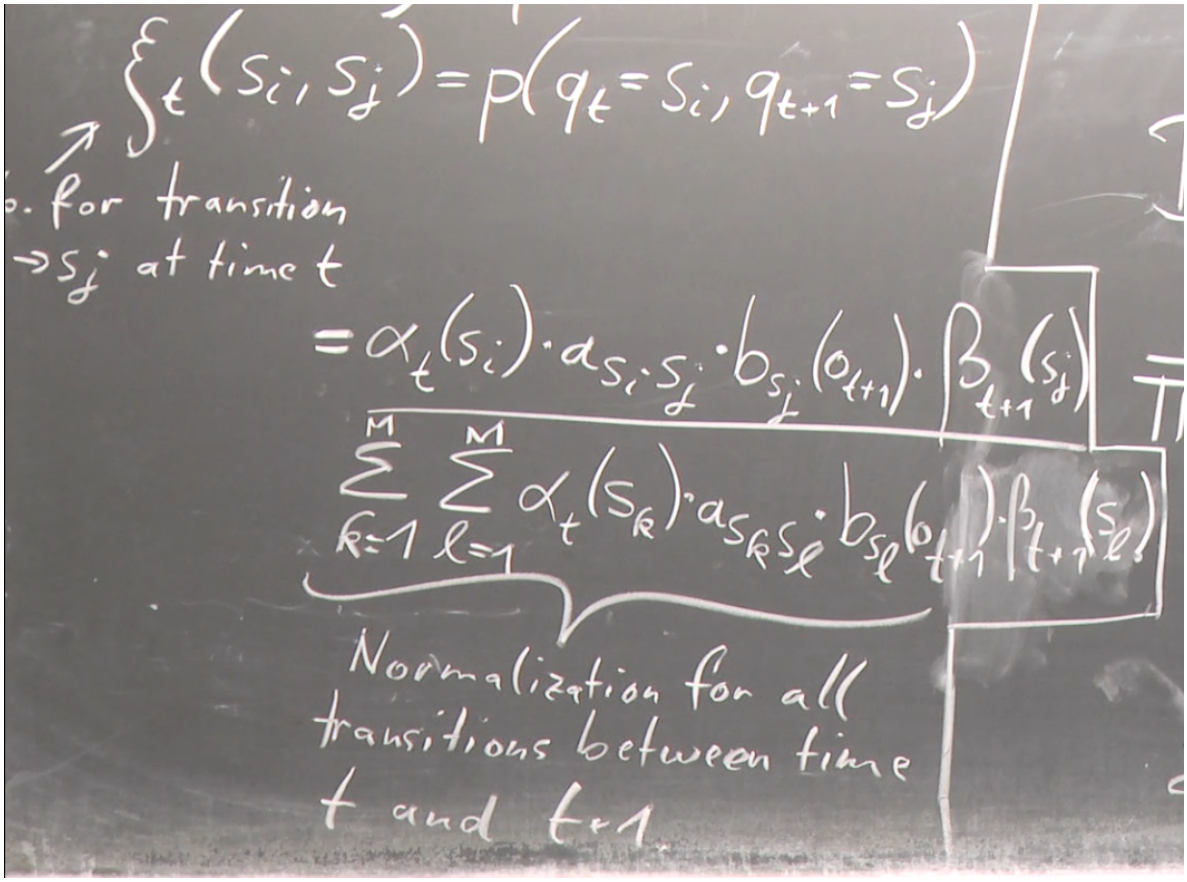
1) Replace every sum by max

2) Technical detail: store the actual path in a second list

3) Solution to task 3 (Training):

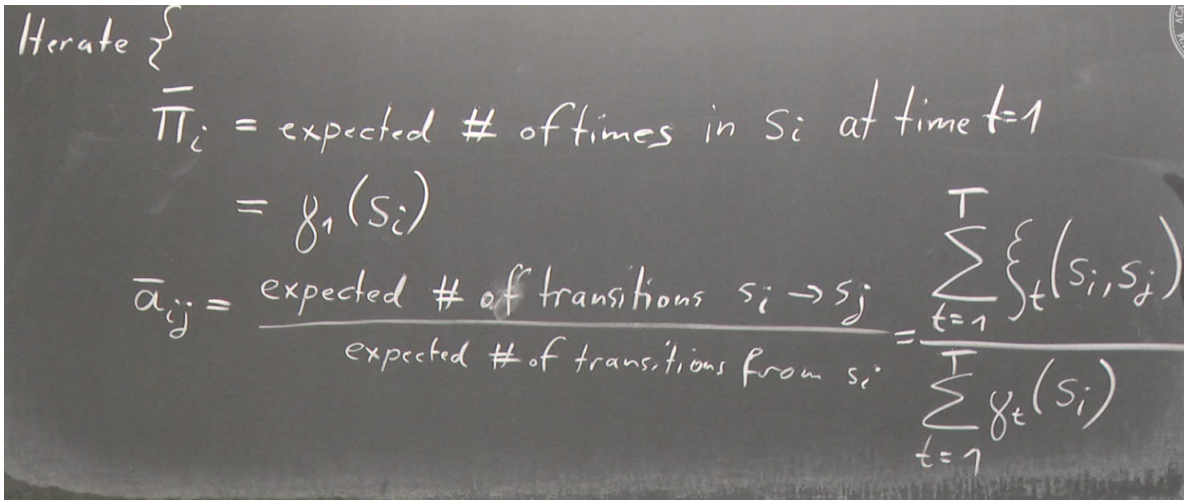
Determine A, B, π: No analytical solution $\Rightarrow$ iterative

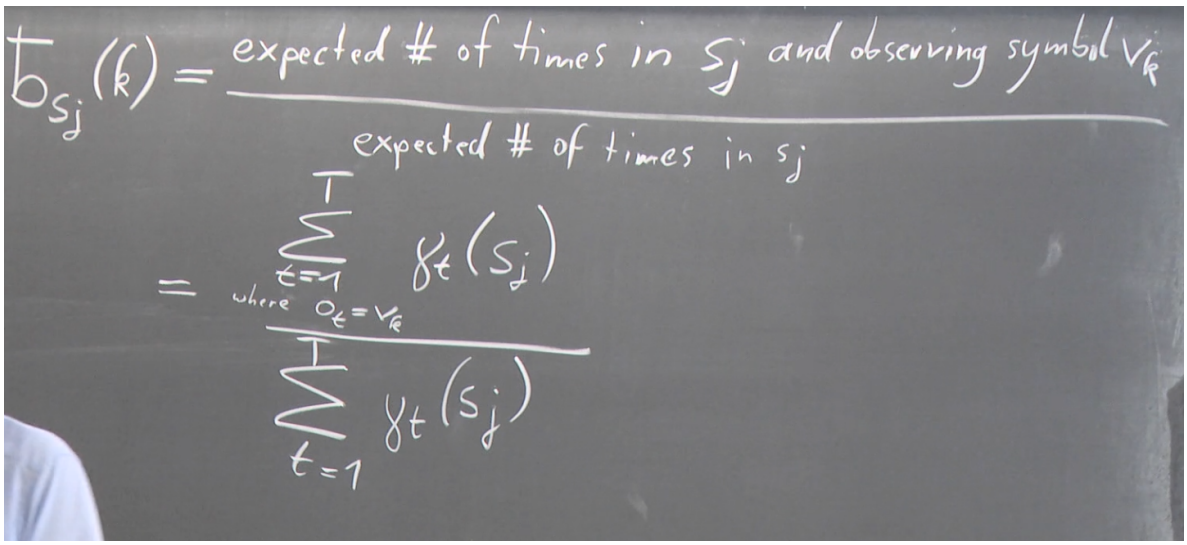
Auxiliary quantities:



Baum-Welch Formulas:

EM-style iteration:




$b_{s_j}(k) = \frac{\text{expected \# of times in } s_j \text{ and observing symbol } v_k}{\text{expected \# of times in } s_j}$
$= \frac{\sum_{t=1}^{T} \gamma_t(s_j) \text{ where } o_t = v_k}{\sum_{t=1}^{T} \gamma_t(s_j)}$

Note-17

Markov Random Fields

Probabilistic graphical model, on an undirected graph.

The graph structure is not limited to modeling 1-D sequnce (like in an HMM), but can cover arbitray neighborhood relations.

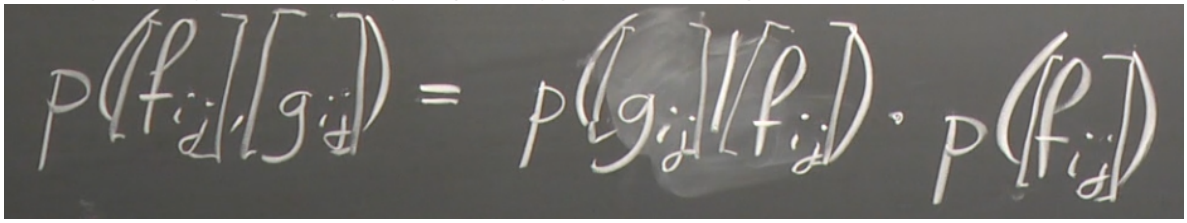
Important special case: consider the pixel grid of an image as a graph:

Each hidden variable depends on :

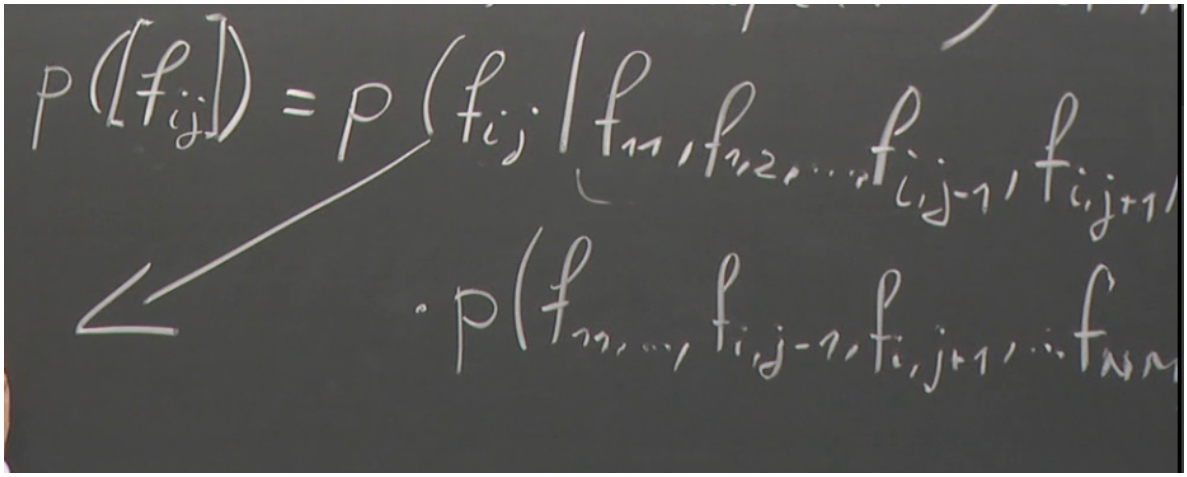
- the associated observation
- its neighborhood of other hidden variables

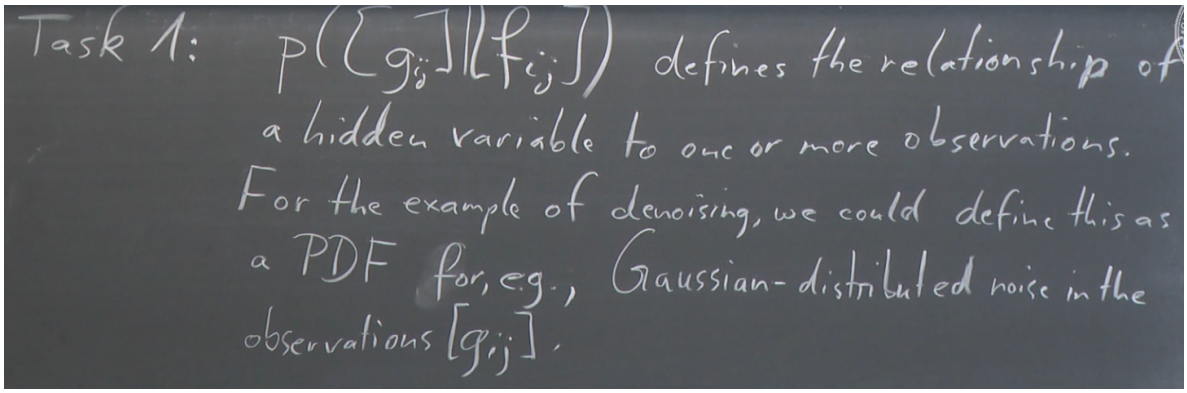
It is independent of all other quantities ⇒ "the trick" to make inference over p(fij,gij)

More Specifically, we model the joint probility p(f11,f12..fNN..gNM)



The Markov assumption limits the statistical dependency of the hidden variables:





This term is also called "data term"

describes relationship to the data/observations

$P([f_{ij}])$ defines the relationship of hidden variables within a local neighborhood.

For the denoising example, this PDF can capture the fact that the ideal image $[f_{ij}]$ is smooth, i.e. neighboring pixels

Note-18

MRF: Markov Random Field

Probabilistic graphical model

Markov Property: Probability of a node labeling only depends on the neighbors of a node

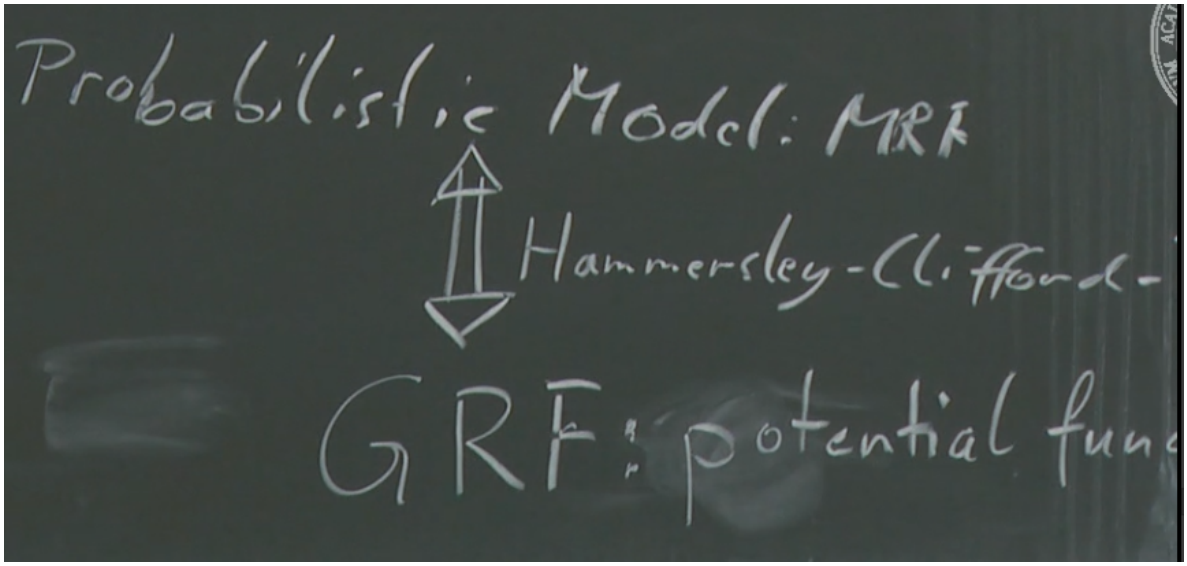
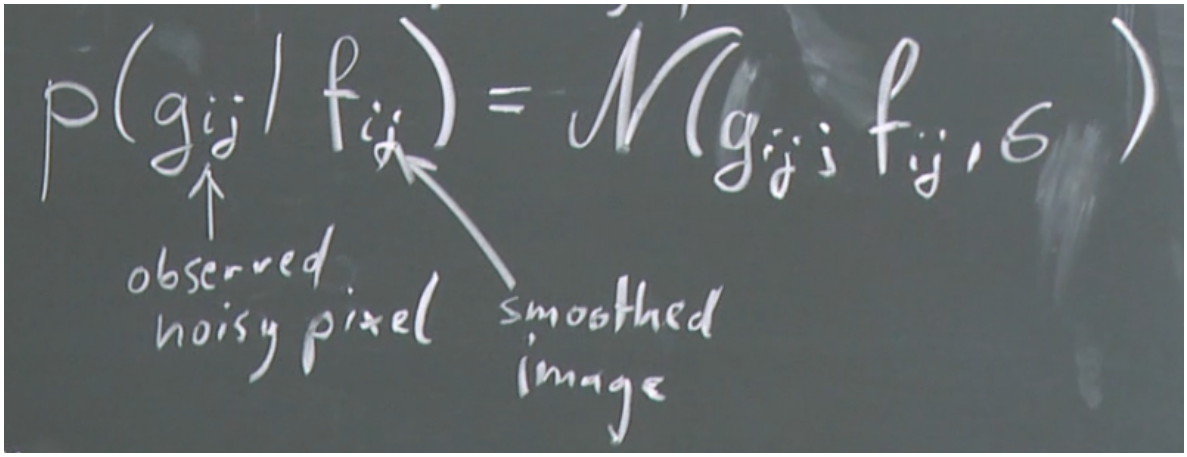
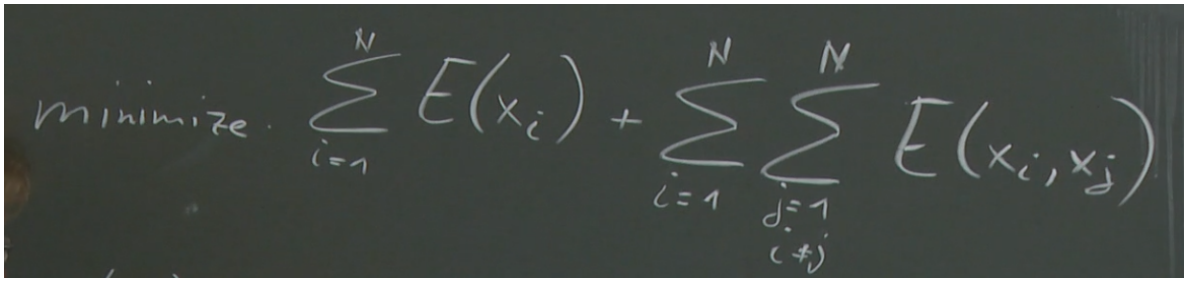


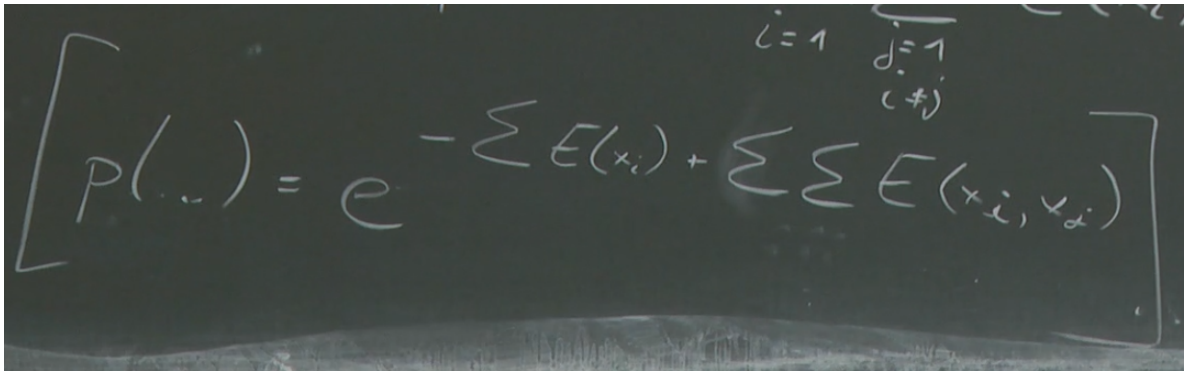
Image smoothing example: "data term"/unary potential



If we only consider the energy terms, the unary potentials are $E(x_i)$,

where x_i denotes the i-th hidden variables, and the pairwise potentials are $E(x_i, x_j)$,





An efficient solution for a finite set of labels can be computed via graph cuts

$\Rightarrow$ the core of the label assignment is a binary labeling via graph cut

This is what we discuss below

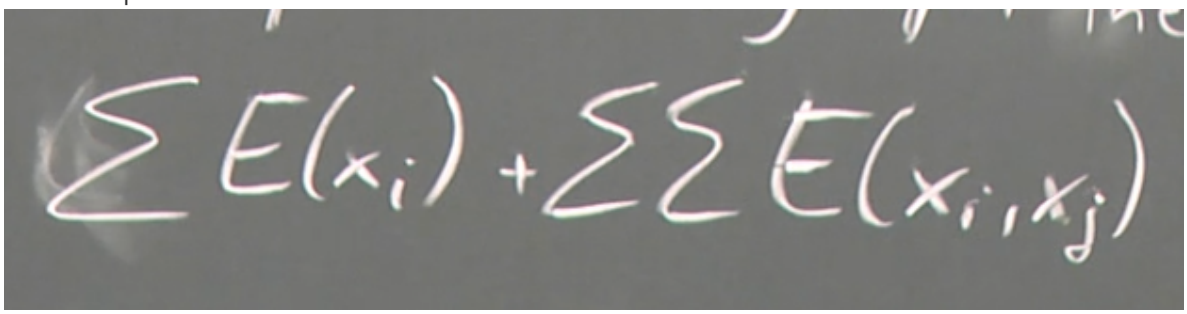
Idea: Identify the nodes in a graph with the hidden variables x_i, identify sources with label 0, drain t

Task: Define the graph to pology and weights such that the mincut separates the nodes into 0's and 1's, with an optimal solution to the energy minimization problem

Assignment of edge weights

1) MinCut works on part of a graph the same way of the par

$\Rightarrow$ decompose

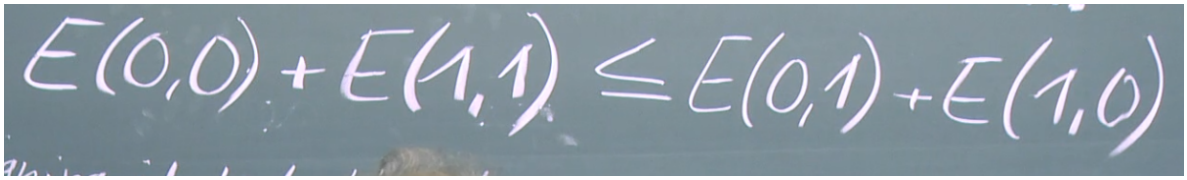


into individual terms

2) Part of the graph E(xi)

3) Part of the graph for

The conversation GRM $\rightarrow$ MinCut can always be done (and the minCut solution is always equivalent to the GRF) If the so-called sub modularity condition holds:



(assigning identical labels to neighbors may not be more expensive/worse assigns different labels)