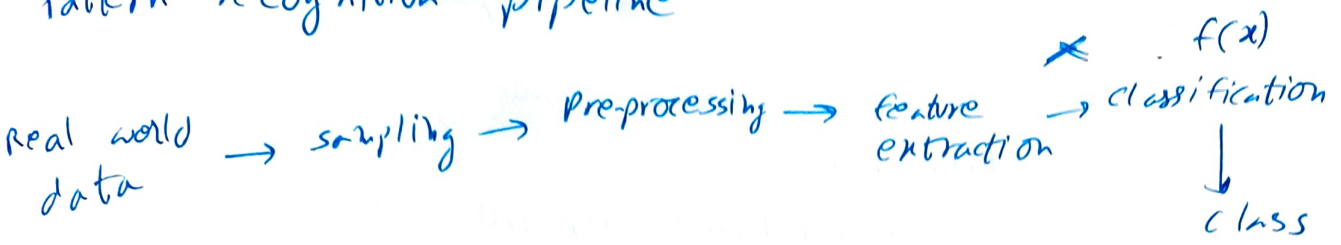
Pattern recognition pipeline



$x \rightarrow$ feature vector

this is classified using a classifier or function $f(x)$ which op. the class y after classification.

Assumption: similar objects map to similar features.

supervised learning

Training sample is a tuple (x_i, y_i).
- x_i: feature vector
- y_i: expected class/desired prediction.

Unsupervised learning

work with unlabeled data. $(x_i) \rightarrow$ operates only on i/p x_i.

- summary of distribution

Parameters
- choice of param. is training

Hyper-parameters
- a parameter that parametrizes the choice of parameters
- d is hyper-parameter

Linear regression

$$y = \beta^T \tilde{x} = \sum_{i=0}^{d} \beta_i \tilde{x}_i$$

$\tilde{x}_i = (1, x_1, \ldots x_{d-1})^T \rightarrow$ plane defined in some space.

↳ offset

Flexibility requires more localization, more parameters & are less robust.

Model selection prob: trade-off b/w choice of parameters/hyperparam. and flexibility.

Local models perform poorly in high dimensional spaces.

Prob. distributions are used to represent data in feature spaces.

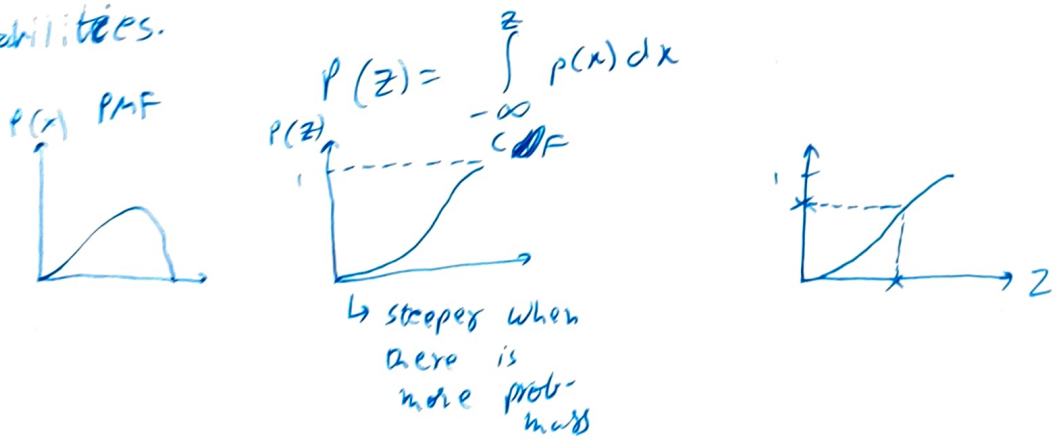
operations on distributions

- fitting a distribution: represent data as distribution
- sampling: create new data points that follow the distribution
- factorizing: technique to reduce complexity like joint distributions.

$$P(y|x) = \frac{P(x|y)\ P(y)}{P(x)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

- logistic regression fits single regression curve to data
- bayesian " " distribution of curves.

CDF $P(z)$ is a running sum that accumulates the probabilities.



drawing a uniform number is always possible.

Sampling Algo

1. discretize the domain of PDF $p(x)$
2. Linearize $p(x)$ if it is multidimensional
3. Calc CDF $P(z)$ of $p(x)$
4. Draw a uniformly distributed num. in b/w 0 & 1.

$$z^* = \underset{z}{\operatorname{argmin}}\ P(z) \geq u$$

Sampling: pics a random number in the domain of distribution based on the prob. described by it. i.e. picks random number based on prob. described by curve in PDF.

Density Estimation

Creates a PDF from samples. It is opposite of sampling.

Parametric density estimation → from pattern recognition.

$$\theta^* = \arg\max_{\theta} p(x_1 \cdots x_n \mid \theta) \rightarrow \text{Max likelihood estimator.}$$

1. Parametric DE req. good func. representation
2. Non-parametric DE can operate on arbitrary distribution,
   ↳ Ex: Histograms.

Number of bins to choose for histogram is a hyper-parameter.

Non-parametric DE

1. kernel-based
2. nearest neighbour

Let $p(x)$ be a PDF in D-dim space, ΔR a region around x

prob. mass in R is $p = \int_R p(x)\,dx$

Assu1: $$p = \frac{\#\text{ points in } R}{\text{total \# of points}} = \frac{K}{N}$$ p → relative freq.

Assu2: R is small

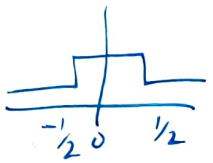
$$p = \int p(x)\,dx = p(x) \int_R dx = p(x) \cdot V$$ ↳ volume

Assu. 1+2 $\Rightarrow$ $$p(x) = \frac{k}{N \cdot V}$$

Parzen Window estimator

Fixes V & leaves k/N. [k varies]

$$k(u) = \begin{cases} 1 & \text{if } |u_i| \le \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$



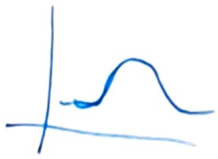
$$K(x) = \sum_{i=1}^{N} k\left(\frac{x - x_i}{h}\right)$$ ↳ Scales the box size. = volume of region

$$\boxed{p(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h^D} k\left(\frac{x - x_i}{h}\right)}$$

Kernel removes discretization error of fixed-distance histogram bins but is still blocky.

gaussian kernel

$$P(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\pi h^2}^{D/2} \cdot \exp\left(\frac{-\|x - x_i\|_2^2}{2h^2}\right)$$



new estimate will look less boxy

Any kernel can be used which satisfies

$$k(u) \geq 0$$

$$\int k(u) = 1$$

K-NN DE

Fixes K & varies V. Calculates V from distance of K-nearest neighbors

Model selection Prob

optimizing hyperparameters / model selection

- hyper-parameters should be trained on validation set (held out part of training set) because on training set the most complex model wins as it achieves lowest error.
- CV (cross validation) is used only for supervised tasks as it requires an objective function for ML. (maximum likelihood)
- Density estimation is unsupervised.
- Use CV when training data is ltd.

KNN & parzen window estimators are memory based i.e. need to store all samples. & also are non-parametric DE.

Bias & variance

[Applicable to both supervised & unsupervised learning]

Motivation behind hyperparameter optimisation is to aim for generalization to new data.

For kernel DE, pitfalls are

- Too large kernel: covers all space with prob. mass but density is too uniform.
- Too small kernel: closely represents data but might assign too low prob. in areas without training data.

- optimal kernel size represents structure of training data & any unobserved data to some extent.

Bias-Variance trade off. → cross validation helps in finding right balance.

Bias → square of average deviation of an estimator from ground truth.

$$\left[\sum_{i=1}^{N} \frac{(y_i^* - y_i)}{N}\right]^2 \longrightarrow ?$$

Variance → variance of the estimates.
expected squared deviation from the estimated mean.

high bias → model undercomplexity: doesn't fit to data.

high variance → model overcomplexity: models structure of data & its noise.

Higher model complexity = lower bias & higher variance.

larger kernel support / k increases bias & lowers variance

smaller " " / k lowers bias & increases variance.

Variance & bias require an error metric to be properly understood & quantified. [Ex metrics are MSE & MAE (Mean Absolute Error)]

Ex: unsupervised methods → reconstruction error for PCA
→ sum of squared dist. from point to μ " k-means clustering.

Random Forests & Variants

decision trees

→ training data

↳ discretise space into reg. intervals

Histogram → partitions space into equi-distant bins.

(3, 2, 2, 0, 1) (1 2, 1 3, 1)

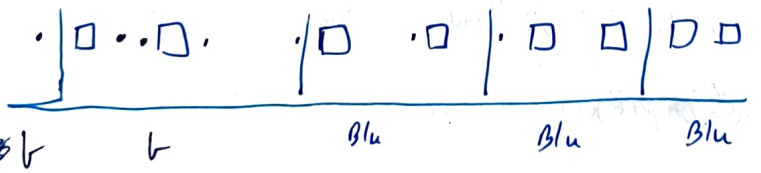
With Max. likelihood

classify bins as (r, (50-50), r, b, (50-50))

- histogram not flexible. Red & blue appear to form 2 continuous dists, but histogram doesn't represent them fully.

Tree-based representation

Can split the domain into uneven patterns/bins.



Size of bin is derived by an optimisation problem in a data-driven approach.

Forest → number of trees together is a forest.

It smoothens the boundaries.

1. tree-based sample representation

2. compare kernels & tree based methods
   - do not need to store data points
   - divide sample space dynamically with objective func.

3. CART [Classification & Regression Trees]
   - Random Forests (RF)
   - density forests

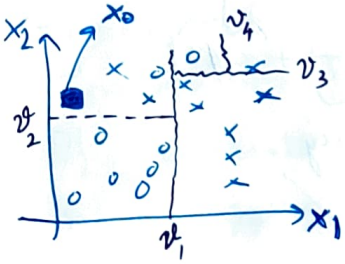
RFs are actively used & not outdated / replaced by Deep neural N/ws.

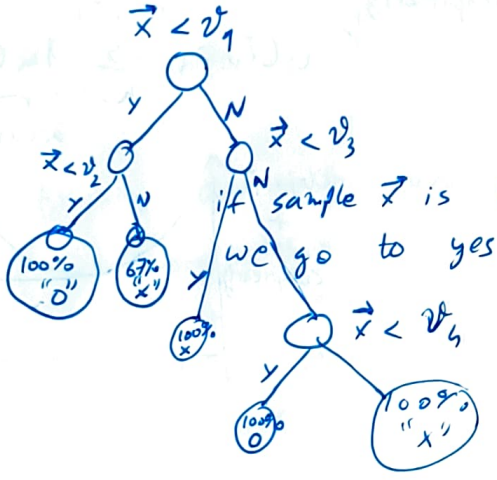
AIM: Understand CART & RF as tools for hierarchical sample space partitioning.

CART

- recursively divide sample space
- perform classification or regression locally in each partition.

Ex





if sample $\vec{x}$ is less than threshold we go to yes else to N.

$x_0 < v_1 \rightarrow Y$

$x_0 < v_2 \rightarrow N$

$x_0 =$ 67% "X" // → classification returns locally observed labels. similar to K-NN.

Idea is similar for regression tree.

splits 1-d space into different parts.

Const function value as regression estimate.

~~random forests~~ fixed value/segment is enough as averaging multiple trees gives a smooth o/p.

Training

1. each node has 2 parameters
   1. axis-aligned split along
   2. one dimension

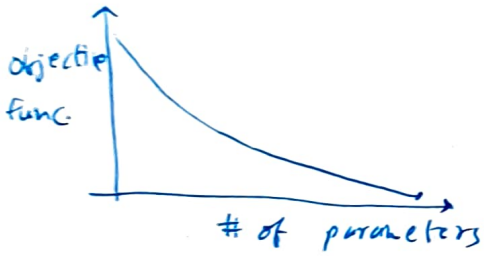
   axis & where/threshold from prev. example.
2. tree has max. height [Hyper-parameter] → can prevent over fitting.
3. Steps
   i) grow tree greedily from root to leaves
   - at each node perform exhaustive search for best split w.r.t an obj. func. $Q_m(T)$
   - stop splitting only if few samples are remaining.

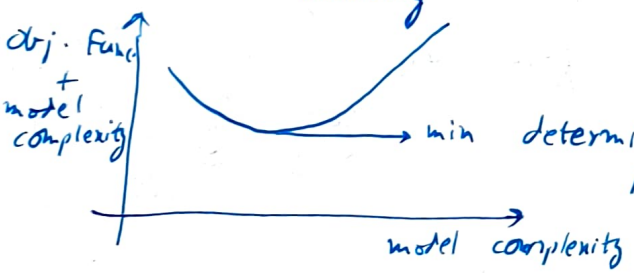
   ii) Prune tree: reduce tree size.
   merging splits which do not help solve the task.

   pruning cost function is tradeoff b/w model accuracy & model complexity.

$$C_\alpha(T) = \sum_{m=1}^{|T|} Q_m(T) + \alpha |T|$$

- α → complexity parameter
- T → subtree
- m → index of region
- $|T|$ → no of terminal nodes
- α → hyper-parameter





Objective functions

Regression

$$Q_m(T) = \sum_{x_i \in R_m} (y_i - \bar{y}_m)^2$$

- $\bar{y}_m$ → average of all samples in R_m
- R_m → samples in region m
- y_i → sample label

-ve cross entropy: classification

$$q_m(T) = \sum_{k}^{K} p_{mk} \ln p_{mk}$$

- K → no of classes
- p_{mk} → relative freq of class k in R_m

Classification : Gini index

$$Q_z(T) = \sum_{k}^{K} p_{mk}(1 - p_{mk})$$

Decision for best splitting candidate

$$\min \frac{|S_l|}{N} Q_l(T) + \frac{|S_r|}{N} Q_r(T)$$

$N \rightarrow$ total samples

$l \rightarrow$ left

$r \rightarrow$ right

$S_{l/r} = \{x_i \in R_{l/r}\}$

minimises loss

searches for candidate split with min. loss

cross entropy & gini-index are better than misclassification rate as the reward is lower for low purity & high reward for high purity.

Prediction in leaf node

classification : relative freq.

regression : average of data points.

RF → more modern than classical CART.

↳ tendency to overfit.

RF is ensemble of CARTs.

caveat → Statistical dependence

1 tree is weak learner, ensemble is strong.

ensemble averages votes of individual trees. RF, an ensemble of randomized CART to counter overfitting.

Randomising components

- bagging : training on random subset of training data
- random subspace projections in each node
  do not do exhaustive search.
  only select d-dimentions for best split
- replace exhaustive search with randomly drawn split candidates.

~~Part (c) (ii) [Rationale behind randomization]~~

Randomising required for (trees are less correlated)

- reducing overfitting
- RF are less sensitive to noise
- enhances generalisation
- resilience to noisy data

What is correlation?

To decrease variance in RF:

- increase n.o. of trees; errors caused by individual trees cancel out, Avg. out prediction of more individual models
- use more data; model can learn more accurately
- tune hyper-parameters; prevent overfitting to data

Rationale behind Randomization.

- adds statistical independence & decorrelates trees.
  - averaging votes of N independent & i.i.d learners → identical independent
    with accuracy >50%, reduces variance.

$$\sigma^2_{\text{i.i.d ensemble}} = \frac{1}{N}\sigma^2$$

increasing N no. of trees reduces variance

  - i.i.d. is too optimistic. If there is correlation ρ

$$\sigma^2_{\text{i.d. ensemble}} = \rho\sigma^2 + \frac{1-\rho}{N}\sigma^2$$

$\rho\sigma^2$: socket variance.

as $N \to \infty$ $\sigma^2_{id} = \rho\sigma^2$ atleast.

Other randomisation steps reduce the correlation ρ.

Configuration of randomisation are additional hyper-parameters: bagging y/n, subspace dimension, no. of candidate splits.

Engineering

- Cross validation without separate validation set.
  Use out-of-bag samples for validation
- more trees increase variance & prevent overfitting.
- deeper trees overfit
- parallelisation is easy.
- RF can be used for multiclass classification unlike SVM [support vector Machine]
- decision boundaries of RF are smoother. & increase robustness
- trees & forests have explainable decisions.

RF can be seen as data-driven (adaptive) variant to Nearest Neighbor (NN) method.

Density forests

- exchange supervised objective func. with unsupervised function to perform DE.
- unsupervised func can work only on structure/distribution of data.
- criminisi/shotton/konukoglu propose to prefer splits that lead to compact gaussian distributions.
  - compactness is measured using determinant of the gaussian covariance.
  - re-define entropy

$$Q_n(T) = \frac{1}{2} \log\left((2\pi e)^D \left|\Lambda(x_i \in R_n)\right|\right)$$

$|\cdot| \rightarrow$ det

$\Lambda \rightarrow$ covariance

  - geometrically, det. approximates rectangular volume around majority samples.

- single tree models the density as multiple truncated trees.
- average over the forests smooths out sharp boundaries.
- not necessary to sample from this density.
  1. Randomly select a tree
  2. " move from root to leaf with prob. $\frac{|S_L|}{|S_L| + |S_R|}$ where $S_{L/R} = \{x_i \in R_{L/R}\}$
  3. " sample from covariance in the leaf.
  4. Repeat 3 until the drawn sample is inside of the leaf region.

Training error: error a model makes on the same data it was trained on.
Increasing depth of tree does not increase training error but typically reduces it as the tree can fit better to the data.

recap

Non-parametric DE & kernel based : K-NN

Tree based method
- do not need to store all samples.
- subdivide sample space dynamically with objective function

- Only representation but no info reg. dist of sample.

1. Clustering
2. variational inference (VI): to draw conclusion on data
   approx. technique to inference tasks
3. Markov chain Monte carlo : (MCMC)
4. Manifold learning : reduces dimensionality while preserving structure of data.

Clustering : segments data into meaningful groups
- Assign identical labels to similar samples.
- Unsupervised technique
- used for exploring data
  1. K-means
  2. gaussian mixture model
  3. mean shift

Manifold learning

represent data manifold in a lower dim. space.
1. PCA
2. Muti-dimensional scaling
3. ISOMAP
4. Laplacian Eigenmap

K-means Clustering.

- represents observations as a few groups.
- hard clustering method: each sample gets a discrete cluster label.
- popular for: conceptual simplicity
  coding "
  reasonably fast execution time.
- Ex: vector quantization of colors.

K means will converge even on uniform data although the result would not make sense. Uniform data has least structure possible.

Ideal objective funct

- cluster is collection of points that are closest to a certain center point.
- assumes there are K centers [k-means clustering]
- we can iteratively update each center from the points close to the center.

objective func

within-cluster distance

$$W(C) = \frac{1}{2} \cdot \sum_{K=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} \| x_i - x_j \|^2$$

looking at summing euclidian dist of pairs of points within same cluster.

$$= \sum_{k=1}^{K} N_k \sum_{C(i)=k} \| x_i - \mu_k \|^2$$

μ_k → mean of all points in cluster k.

N_k → no of points

x_i → sample in cluster.

$C(i)$ → cluster id

K-Means always converges as with each iteration the W(C) decreases. However it may converge to local minima rather than global minima.

- locally optimal: K-means is locally optimal

different initializations could result in different results.

- The clusters partitions the space: this partitioning is called

Voronoi tesselation

- Convergence: change in mean vectors/points changes less than a threshold chosen [Epsilon] ε.

GMM

- covered in PR
- Models a PDF as sum of K normal distributions $\mathcal{N}$ with weights π_k

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \qquad 0 \le \pi_k \le 1, \quad \sum_{k=1}^{K} \pi_k = 1$$

- can be seen as parametric DE or as a clustering method.
- Clustering view: one mixture component $\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$ is one cluster.

$f(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ $= \infty$ if $\sigma \to 0$ (MLE singularity)

Model of observations & hidden variables

- direct fitting of GMM parameters via max likelihood fails due to singularities: likelihood $\to \infty$ when one component has only 1 observation. [1 gaussian to cover all data vs very narrow distribution centered around 1 point]
- instead use Expectation-Maximization (EM) algorithm. [similar to Max. Likelihood method]
- hidden variable $p(x) = \sum_z p(x, z)$ introduced by expressing the density as a marginal of a larger joint distribution.

step 1: define GMM parameters in terms of the marginal.

step 2: define numerical solver i.e. EM algo

$$p(x, z) = p(z)\, p(x \mid z) \quad [\text{Product rule}]$$

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

$$\| \qquad\qquad \| \qquad\qquad \| \qquad\qquad \| \qquad\qquad \|$$

$$\sum_z p(x, z) = \sum_z p(z)\, p(x \mid z)$$

z is an indicator vector that tells us in which component we are in.

if 1. z is one out of K indicator
2. $p(z) = \pi$ (marginalized)
3. $p(x \mid z) = \mathcal{N}$ (marginalized)

- set z as binary indicator vector
  it indicates to which component it belongs

$$z_k = \{0, 1\} \qquad \sum_{k=1}^{K} z_k = 1$$

- set prior so that its marginal is π_k

  i.e. $p(z_k = 1) = \pi_k$

$$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}$$

z_k selects/deactivates all components except one k or current component.

use same technique

$$p(x|z) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}$$

resulting in single gaussian at $z_k = 1$

$$p(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)$$

(*)

- Introduce z & then cancel it to make hidden param. explicit, to include them in numerical optimization.

- this method is soft-clustering variant of k-means.

- responsibility indicates degree of membership of a sample to a component. $p(z_k = 1|x) = \gamma(z_k)$ responsibility: prob. that sample x belongs to component k.

- Ancestral sampling is used for sampling from GMM.

  sampling from density forest was similar.

$$p(x) = \sum_{z} p(x, z) = \sum_{z} p(z) = \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}$$

EM algo:

1. initialize $\theta = (\pi_k, \mu_k, \Sigma_k)$
2. Expectation: determine membership of sample to GMM component, i.e posterior distr. of latent variable $p(z|x, \theta)$
3. Maximization: from those memberships find θ^{new} via maximizing expectation of complete-data log likelihood $\sum_{z} p(z|x, \theta) \ln p(x, z|\theta^{new})$
4. set $\theta = \theta^{new}$ & goto 2 until convergence.

Model selection algorithms for GMM such as MCMC (checkthis) & VI work towards obtaining the posterior of latent variable.

Mean shift algorithm

- iteratively ascends the gradient of a kernel density estimate
- converges to mode (local max) of density without estimating full density. we want to get to a densest point/cluster centre.
- middle way b/w k-means & GMMs.
- clusters result from a few local kernel DEs.

Kernel notation & constraints

- has to be radially symmetric [must depend only on euclidian dist to a sample]
- must accept squared diff. → sym.

$$K(x_0, x) = c \cdot k\left(\|x_0 - x\|_2^2\right)$$

gaussian kernel

$$k_{gauss}(x) = e^{-\frac{1}{2}x}; \text{ std dev. is set to } 1$$

Epanechnikov kernel

$$k_{Ep}(x) = \begin{cases} 1 - x & \text{for } |x| \leq 1 \\ 0 & \text{o.w} \end{cases}$$

- kernel sizes: assume $\|x_0 - x\|^2$ are already size normalised
- smaller kernel give more local clusters
- larger kernel clusters start to merge. Large kernels give less clusters with
- do not have to preset no. of clusters

KMeans	GMM	Mean shift
- result from min. dist. b/w samples	- result from full density estimation	- result from local kernel density estimates.

Using kernel ~~density~~ estimates to go to a local mode of density via gradient ascent.

gradient

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

$\nabla \rightarrow$ can be taken in due to linearity.

i.e, gradient can be calculated on the kernel function.

using kernel profile $k_\lambda(x_0, x) = c\left(\|x_0 - x\|_2^2\right)$ & substitute $s = \|x_0 - x\|_2^2$

$$\frac{\delta k(s)}{\delta(s)} = k'(s) \; ; \quad \frac{\delta(s)}{\delta(x)} = \frac{\delta (x_i - x)^T (x_i - x)}{\delta} = -2(x_i - x)$$

$\stackrel{!}{=}$ → define/set to operator

$$\therefore \nabla p(x) = \frac{1}{N} \sum_{i=1}^{N} c \cdot k'\left(\|x_i - x\|_2^2\right)\left(-2(x_i - x)\right) \stackrel{!}{=} 0$$

$$\sum_{i=1}^{N} k'\left(\|x_i - x\|_2^2\right)\left(-2(x_i - x)\right) = 0$$

$$\Rightarrow \sum_{i=1}^{N} k'\left(\|x_i - x\|_2^2\right) x_i - \sum_{i=1}^{N} k'\left(\|x_i - x\|_2^2\right) x = 0$$

take to other side

$$\Rightarrow \frac{\sum_{i=1}^{N} k'\left(\|x_i - x\|_2^2\right) x_i}{\sum_{i=1}^{N} k'\left(\|x_i - x\|_2^2\right)} - x = 0 = m^{(t)}(x)$$

$t \rightarrow$ iteration

$m^{(t)}(x) \rightarrow$ mean shift vec.

$$x^{(t+1)} = x^{(t)} + m^{(t)}(x) = \frac{\sum_{i=1}^{N} k'\left(\|x_i - x^{(t)}\|_2^2\right) \cdot x_i}{\sum_{i=1}^{N} k'\left(\|x_i - x^{(t)}\|_2^2\right)}$$

Algo:

1. calc mean shift vector [one gradient ascent step]
2. update pos $x^{(t)}$
3. repeat 1 & 2 until convergence. [until gradient = 0]
4. group samples converging to same location

Parameters req.

- kernel parameters [window size]
- cluster linking param for post processing [ex: dist threshold for grouping data points to clusters]

- with K_{EP}, the update $x(t+1)$ is just a calc of mean.
- Euclidean dist is sensitive to scaling differences → normalise the dimensions of the samples

Model selection for k-means

No objective func. as it is unsupervised.
No labels to compare.
What k to choose?

Gap statistics → statistical way to choose k.

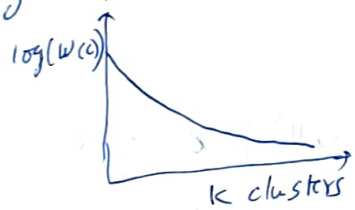
idea → examin WC distance for different k.
select smallest k for which W(c) is substantially better than W(c) of k+1 clusters.

natural choice for compact clusters

$$W(c) = \sum_{k=1}^{K} N_k \sum_{c(i)=k} ||x_i - \mu_k||^2$$

sum of euclidian dist. from samples in cluster k to the mean of cluster k multiplied by No. of samples in cluster k.

For increasing k, W(c) has to decrease



- we cannot search for min W(c) as it will be number of samples, i.e. group n samples in n clusters which is not sensible. [optimal k cannot be n.o of samples]
- elbow method is not good too: searches for bend on W(c) curve. it would be unsure which bend is better.

Gap statistics [captures structure in data] [tweaked usage of W(c) for finding optimal k]

- relate W(c) of actual samples to W(c) of an artificially created reference.
- reference samples drawn from uniform distribution. representing u/l with least poss. structure. [no natural clusters / structure]

Algo
1. draw B sets of uniform distributions [use 20]
2. On those distr. calc. log(W(c)) for different K values. denote result $\log(W_{unit}(c))$
3. For the K clusters calc. gap G(k) as diff u/l ref. $\log(W_{unit}(c))$ & $\log(W(c))$.
4. select K as $K^* = \arg\min_k \{ k \mid G(k) \geq G(k+1) - s'_{k+1} \}$

$s'_{k+1} = s_k \sqrt{1+1/B}$ → unbiased estimate of std. deviation s_k

the actual $\log(W(c))$ is lower than $\log(W_{unif}(c))$ because the ref. has least possible structure [unif. distr]

large (+ve) uniform

think of entropy ^ small (-ve) non-uniform

Relation with Huffman code?

For the graph choose k where the condition in point 4 is met the first time, because we have to choose smallest k.

Why choose smallest k?

- Ockham's razor: choose simplest model unless we have good reason otherwise.

Gap statistics formulates model selection as "finding smallest clustering that finds notable structure"

Sampling & GMM MCMC inference

choose k, number of components.

complicated than k-means, can be done in fully probabilistic way.

- GMM fitting with fixed k had no closed form sol. but iterative EM algo.

sol using additional approximation for selecting k.

1. MCMC: approximate intractable func. with finite samples
2. Variational Inference: approximates intractable func. with simpler tractable func.

Why we need sampling? **Motivation behind Sampling**

- To replace analytical sol.
- if we cannot simulate

Then can we approximate by drawing samples?

Ex: expectation $\mathbb{E}[f] = \int f(z)\, p(z)\, dz$ cannot be solved analytically.

but draw L samples from $p(z)$ & calc.

$$\hat{f} = \frac{1}{L} \sum_{l=1}^{L} f(z^{(l)})$$

This estimator is unbiased $\Rightarrow \mathbb{E}[\hat{f}] = \mathbb{E}[f]$ & variance

$$\text{var}[\hat{f}] = \frac{1}{L}\, \mathbb{E}\left[f - E(f)\right]^2$$

accuracy of estimator doesn't depend on dimension of z, i.e. few samples suffice

1st time this happens gap is not increasing anymore (which gives least K for which W(c) is max)

1st occurance before curve the gap decreases ie. here the occurance of difference of W(c) is large

Sampler from Lecture-1 can operate on general distr.
but its usefulness is ltd.
- req. Full density rep. at every location.
- whatever our density rep. is, we have to convert it to histogram.

Histograms are either
- quite coarse [few bins]
- quite inefficient (10^6 bins); storage is not prob. but each bin has to be filled with data points for sufficient statistics
  Ex: 1 billion bins req 10 billion data points.

Tradeoff we make here propagates into approximation error of sampler.

Alternatives:

Parametric alternative [Parametric std. distr.]
- Analytic mapping from uniform distr. to other distr. exist
  - gaussian
  - exponential
  - cauchy.

- Algo
  1. sample $p(z)$ from uniform distr.
  2. Transform sample into target distr. $p(y)$ with analytic mapping
     $y = f(z)$

note: when converting one distr. to other have to take the Jacobian into account for the derivative.

$$p(y) = p(z) \left| \frac{dz}{dy} \right|$$

$$y(y_1 \cdots y_m) = p(z_1 \cdots z_m) \left| \frac{\delta(z_1 \cdots z_m)}{\delta(y_1 \cdots y_m)} \right|$$

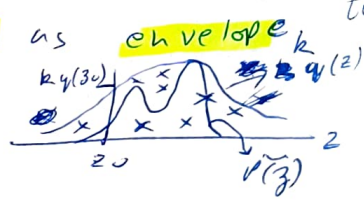
sampler more efficient as it is analytic mapping.
Similar to sampler in 1st lecture, instead of calc. CDF calc. mapping

$$y = f(z)$$

Rejection sampling

distribution are often more complicated ↗ ie. cannot apply parametric methods
to obtain $p(z)$ up to unknown norm. factor (z_p) $p(z) = \frac{1}{z_p} \tilde{p}(z)$ hence it may be possible

- define a simpler distr. $q(z)$ & const k as envelope to $\tilde{p}(z)$, s.t. $k q(z) \geq \tilde{p}(z)$ for all z.



$q \rightarrow$ analytic function.

1. draw sample z_0 from $q(z)$
2. draw u_0 from uniform distr. $[0, kq(z_0)]$
3. reject (z_0, u_0) if $u_0 > \tilde{p}(z_0)$ otherwise return z_0

$u_0 \rightarrow$ y axis value of sample in uniform distr.

certain prob. mass of the drawn sample belongs to $\tilde{p}(z)$ & some to $kq(z)$.

Why is this method correct?

- prior to rejection: the pair (z_0, u_0) is uniformly distributed across the area of the curve $kq(z)$
- after rejection, the pair (z_0, u_0) is uniformly distr. across the area of the curve $\tilde{p}(z)$

Adaptive Rejection sampling

Rejection sampling gets inefficient if q & p differ too much, ie. more samples have to be discarded.

So this method form a envelope using piece-wise linear functions in log space, which is better fitting.

But q might not have a simple analytic form.

- work well on log concave functions, ie. derivatives of $\log p(z)$ are non increasing function of z.
- fitting lines to log of func. = fitting piecewise exponential distr. to orig. func.

$$q(z) = k_i \lambda_i \exp\{-\lambda_i (z - z_i)\} \quad z_{i-1,i} < z \leq z_{i,i+1}$$

Sampling in Models with Many variables/attributes

cases:

1. Random variable has many attributes $x = (x_1, \dots x_N)$ or
2. model consists of many dependent RV. $x_1, \dots x_M$

Point 2. corresponds to modelling a sampling based sol. for GMM fitting with model selection: $K, \pi_1, \dots \pi_K, \mu_1, \dots \mu_K, \Sigma_1, \dots \Sigma_K$ & hyperprior are all R.V.

Such cases can be called high-dimensional spaces.

One sample is then full set of assignments to all unknowns

Impossible to use rejection sampling & ASR because gap b/w q & $\tilde{p}$ increases

Ex: Envelopes around all GMM var.

2. Draw a set of variables A one of them will be likely outside of $\tilde{p}$. [more variables more likely it is]

MCMC sampling

Mitigates issues in high-dimensional spaces.

Idea:
- sample from 1 var at a time
- repeat this sampling in an iterative manner, using recently sampled values.

- sample in iteration τ from a state space of variable $z^{(\tau)}$ using prev. iterations $z^{(1)} \dots z^{(\tau-1)}$
- Markov chain, models only 1st order statistical dependencies, ie we only look at prev. states,

$$p\left(z^{(\tau+1)} \mid z^{(1)} \dots z^{(\tau)}\right) = p\left(z^{(\tau+1)} \mid z^{(\tau)}\right)$$

skipped
Ex: Metropolis-Hastings method to consider = considering 1 prev state
special case Gibbs sampling all past

Gibbs sampling

sample distr. $p(z) = p(z_1, \dots z_M)$ of M R.V. [somehow initialised]

In each step update 1 variable by drawing from that variable conditioned on others.

sample $z_1^{(\tau+1)} \sim p\left(z_1 \mid z_2^{(\tau)} \dots z_M^{(\tau)}\right)$ → update for that var by a random draw from distr. of that var. conditioned on all other variables.

$z_2^{(\tau+1)} \sim p\left(z_2 \mid z_1^{(\tau)}, z_3^{(\tau)} \dots z_M^{(\tau)}\right)$

$\vdots$

$z_M^{(\tau+1)} \sim p\left(z_M \mid z_1^{(\tau)} \dots z_{M-1}^{(\tau)}\right)$

- subsequent sample are correlated due to markov chain.
- after T iterations sample of complex distributions is a single state.
  i.e. after $M \cdot T$ sub-draws
  (M ↓ No. of RV; T → iterations)

Bayesian GMM fitting : Model setup

- Put GMM parameter π, μ_k, Σ_k req a prior distribution
- Use conj. prior; param. distribution × prior = same distr. family as parameter distr.
- parameters to prior distr. are set to fixed values. [hyperpriors]
- value is not important
- By increasing n.o of observations overwrites the prior.
- prior's shape distr. in absence of observation.

Kullback Leibler Divergence

Describes how different 2 distributions p & q are

$$KL(p||q) = \sum_z p(z) \log\left(\frac{p(z)}{q(z)}\right) = -\sum_z p(z) \log\left(\frac{q(z)}{p(z)}\right)$$

It is the expected log difference b/w p & q WRT p.

derivation: let X be random discrete var with states x_i & $p(x=x_i)=p_i$

$H(p) = -\sum_i p_i \log(p_i)$ → lowerbound for encoding X.

Without optimal p & having encoder q, the relative entropy is:

$$KL(p||q) = \int p(x) \log q(x)\,dx - \left(-\int p(x) \log p(x)\,dx\right)$$

$$= -\int p(x) \log\left(\frac{q(x)}{p(x)}\right) dx$$; this is the additional effort required for encoding p

Relative entropy ⟺ Kullback-Leibler divergence

Expected number of extra bits for encoding text with distribution p, with an optimal code for distribution q.

Properties: 1) $KL(p||p) = 0$

(non-ve) 2) $KL(p||q) \geq 0$; equal only if $p=q$

(non-symmetric) 3) $KL(p||q) \neq KL(q||p)$; it is not symmetric (hence it is not a distance)

Proof for Non-negativity

Jensen's inequality states: $f(tx_1 + (1-t)x_2) \leq t f(x_1) + (1-t) f(x_2)$

generalised form $f\left(\sum_i t_i x_i\right) \leq \sum_i t_i f(x_i)$

Mapping $\sum_z p(z) \log \frac{q(z)}{p(z)}$ to $\sum_i t_i f(x_i)$

(-ve in KL divergence is cancelled by additional -ve for flipping log's concavity to convexity)

hence

$$\log \sum_z p(z) \frac{q(z)}{p(z)} \leq \sum_z p(z) \log \frac{q(z)}{p(z)}$$

$\log \sum_z p(z) \frac{q(z)}{p(z)} = 0$ as summation of $q(z) = 1$ & $\log(1) = 0$

$\therefore$ KL divergence ≥ 0 always.

Non-Symmetry

- when fitting a distribution $q(x)$ to a fixed distribution $p(x)$ by minimision KL divergence, $KL(q||p)$: it focuses q to a mode of p, to avoid probability mass of q anywhere where p is close to 0.
- Minimizing reverse $KL(p||q)$ leads to a q that is broadly covering p, i.e. q distributes prob mass everywhere where p is non-zero.

General Expectation Maximisation

It is a solver for models consisting
- observations, X
- hidden variables, z
- parameters θ

when it is not possible to fit θ to X, i.e. difficult/infeasible to solve directly for $p(x|\theta)$.

In general we seek to maximise likelihood $p(x|\theta)$ as

$$p(x|\theta) = \sum_z p(x, z|\theta)$$

where z is hidden variables to simplify fit by splitting up the fitting task.

2 steps
1. E step : improves z via $p(z|x, \theta)$
2. M step : improves θ via ____.

Decomposition: log evidence decomposes to ELBO & KL divergence

$$\log p(x|\theta) = \log \sum_z p(x,z|\theta) = L(q,\theta) + KL(q\|p)$$

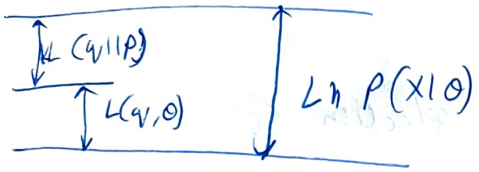
achieved after introducing a distribution over latent variables $q(z)$

where:

$$L(q,\theta) = \sum_z q(z) \log\left\{\frac{p(x,z|\theta)}{q(z)}\right\}$$

$$KL(q\|p) = -\sum_z q(z) \log\left\{\frac{p(z|x,\theta)}{q(z)}\right\}$$

to undo decomposition: put $\log p(x,z|\theta) = \log p(z|x,\theta) + \log p(x|\theta)$ into ELBO



goal is to iteratively improve $q(z)$ & θ in $L(q,\theta)$

steps
1. E step updates $q(z)$ with θ^{old}, effectively setting $KL(q\|p)=0$, by setting $q(z) = p(z|x,\theta^{old})$
2. M step update θ with $q(z)$ which recreates $KL(q\|p)>0$ at new location $p(x,z|\theta^{new})$

M-step provides a set of parameters
E-step " a $q(z)$ that closes gap $p(x|\theta)$

Variational Inference

Expresses posterior $p(z|x)$ with a suitably selected & parametrized distribution q.

Used when estimation of original posterior $P(z|x,\theta)$ is infeasible then fit a feasible distribution q.

2 choices for simplification
1. parametric distribution $q(z|w)$ where fitting parameter w is tractable
2. factorized distr. $q(z_1, z_2 \ldots z_M) = \prod_{i=1}^{M} q_i(z_i)$ where fitting independant factors is tractable.

Example for parametric distribution: Entropy

$$H[p] = -\sum_i p(x_i) \log p(x_i)$$

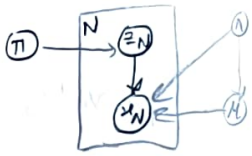
calculus of variations can be used to find $p(x)$ that maximises/minimises $H[p]$

A simpler distribution q could be a Gaussian, such that
- min. entropy is reached for $\sigma \to 0$
- max. " " " " $\sigma \to \infty$

Note: q only approximates these solutions ∧
∨ only models a tiny subspace for all possible distributions q.

EN For factorized distribution: GMM
- GMM model is complex for automatic selection of no. of components.
- estimating $p(z|x,\theta)$ [posterior] is hard, so std. EM doesn't work
- VI solution approximates original model p with a distr. q that can be factorised into multiple simpler terms.

Full GMM model



$$p(X, Z, \pi, \mu, \Lambda) = p(x|z,\mu\Lambda)\, p(z|\pi)\, p(\pi)\, p(\mu|\Lambda)\, p(\Lambda)$$

$Z \to$ latent variable, indicates which component creates which sample
Z depends on distribution π [the no. ∧ weight of the components]
component means μ depend on associated standard deviation (Λ)
observations X depend on Z, μ, Λ

Factorized approximation
unknown variables captured in variational distribution q.
tractable if q can be factorised b/w Z ∧ π, μ, Λ
i.e. $q(Z, \pi, \mu, \Lambda) = q(Z)\, q(\pi, \mu, \Lambda)$

$q(Z)\, q(\pi, \mu, \Lambda) = q(Z)\, q(\pi)\, q(\mu|\Lambda)$ [$\because \pi, (\mu, \Lambda)$ are conditionally independent in p]

Mean Field approximation

Generalizes EM framework to do variational inference on factorized distribution.

$$q(z) = \prod_{i=1}^{M} q_i(z_i)$$ [factoriser all hidden variables into M partitions]

if q_i is simple then E (expectation) can be calculated & reuse the EM framework.

→ IN EM FRAMEWORK

$$L(q,\theta) = -KL(q_j \| \tilde{p}) - \sum_{q_i \neq q_j} \int q_i \ln q_i \, dz$$

where $$\log \tilde{p}(x, z_i) = \sum_{q_i \neq q_j} \log p(x,z) = \int \log p(x,z) \prod_{q_i \neq q_j} q_i \, dz_i$$

Curse of Dimensionality

Looked at low dimensional feature vectors [also to visualise]
Real data could consist more dimensions.
Difficulty increases with data dimensionality.
↳ = curse of dimensionality

3 difficulties:

1) Visualization:
   - data often more larger than 3D
   - To understand data, it helps to visualise / simplify it.

   Ex: Remote sensing processes satellite recordings [photograph of surface of earth done in several narrow color bands]

   DNN learn featuremaps [representations] with many dimensions

   Recommender systems compress large & sparce matrices into few product proposals

2) Exponential n.o. of bins
   Assumption: similar features at similar locations in sample space.
   ∴ Classifier or regressor must make local predictions.
   For simplicity equally sized cells: n.o. grows exponentially with dimentions.

   which requires more model parameters & most importantly growing n.o. of data points for sufficient observations per cell.
   (Kernel estimators have difficulties in higher dimensions)

3) Distances become less discriminative
   - most samples lie at boundary
     [Dist. bw samples become similar & hence dist. are less meaningful]
   - rejection sampling is also affected due to this issue in high dimensions. [May not converge quickly/more rejections]

Example/illustration

D-dim sphere with radius 1, with uniformly distributed samples.

$V_D(r) = k_D \cdot r^D$ → general form of volume in D-dim

↳ const. vol. factor

consider another sphere with radius $1-\epsilon$.

fraction of data in 2 spheres $f_D(\epsilon) = \frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = \frac{1-(1-\epsilon)^D}{1} = 1-(1-\epsilon)^D$

looking at $f_D(\epsilon)$ it rapidly reaches 1

$D=10$ $\epsilon=0.1$ $f_{10}(0.1) = 65\%$

$D=100$ $\epsilon=0.01$ $f_{100}(0.01) = 63\%$

Dimensionality reduction

1. PCA
2. MDS
3. ISOMAP
4. LE

PCA

Also known as Karhunen-Loeve Transform (KLT)

[comes in IVC] Transform coding.

- provides compact representation in lower space.
- PCA is linear projection onto orthogonal basis U

$$U_i^T U_j = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{Otherwise} \end{cases}$$

- basis are eigen vectors of data covariance (Mean-free)
- Idea: normalize data & to perform eigen value decomposition.
- Magnitude of eigen values indicates contribution of a dimension to covariance of data.

Objective function

vectors $x \in R^d$ d-dimen. space

Find linear mapping $\phi: R^d \to R^{d'}$ $d' \ll d$ [maximises variance of data along each dimension]

J → obj func.

$$J = \sum_{i,j=1}^{N} (\phi_{x_i} - \phi_{x_j})^T (\phi_{x_i} - \phi_{x_j}) + \lambda(\phi^T\phi - 1) \quad ; x_i, x_j \to \text{data points.}$$

zeromean samples $\Rightarrow \sum_{i=1}^{N} x_i = 0$ [in practice subtract mean of all samples]

derivation:

- (*) seek projection of U onto 1D subspace that maximises the variance & show U is largest eigen vector of covariance matrix.
- U is 1st col in ϕ, further vectors iteratively obtained.
  - proj data onto d-1 dimensional subspace orthogonal to U
  - Repeat: (*)

To begin

$u \in \mathbb{R}^d$ be an arbitrary direction with unit length

i.e. $u^T u = 1$

Inner prod $u^T x$ projects x onto a 1-D space

Variance of proj data : $\frac{1}{N} \sum_{i=1}^{N} (u^T x_i - u^T \bar{x})^2 = u^T S u$; $\bar{x}$: component wise mean of all x_i

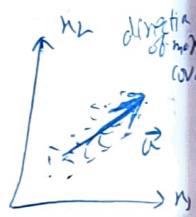
S : covariance matrix

$$S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

to maximise variance of proj data

$u^T S u + \lambda (1 - u^T u) \rightarrow$ maximise

$\lambda \rightarrow$ lagrangian multiplier to includ constraint $u^T u = 1$



$\frac{\delta}{\delta u} u^T S u + \lambda (1 - u^T u) \stackrel{!}{=} 0$; for maxima.

$2Su = 2\lambda u$

$Su = \lambda u \Rightarrow$ eigen vector decomposition of S.

i. largest eigen vector associated with largest eigen value gives max covariance. This vector is called principle component.

Approximation error / amount of info lost when operating in low dim. space $= \frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

if $= 0.98$, then it means subspace preserves 98% of variance of data.

MDS

- equivalent to PCA but operates on distances b/w samples
- useful if only relative info about samples is available. (Ex: from a learned similarity measure)

operates on dist. matrix $D = [d_{ij}^2]$ where $1 \le i, j \le N$

where $d_{ij}^2 = (x_i - x_j)^T (x_i - x_j)$; but x_i & x_j are unknown ; $x_i, x_j \in R^d$

Aim: construct $\tilde{X} = (\tilde{x}_1 \dots \tilde{x}_N) \in R^{N \times d'}$ where $d' << d$ is an orthogonal proj. onto d'-dim subspace.

the unique solution X must be mean free as we cannot distinguish spatial translations. (mean = 0).

Matrix Notation for d_{ij}^2:

$$d_{ij}^2 = (x_i - x_j)^T (x_i - x_j)$$
$$= x_i^T x_i - x_i^T x_j - x_j^T x_i + x_j^T x_j$$
$$= x_i^T x_i + x_j^T x_j - 2 x_i^T x_j$$
$$D^2 = \text{diag}(X^T X) \cdot \mathbb{1}\mathbb{1}^T + \mathbb{1}\mathbb{1}^T \cdot \text{diag}(X^T X)^T - 2X^T X \longrightarrow (*)$$

$\mathbb{1} = (1, 1, 1, \dots 1)^T \in R^N$

$\text{diag}(X^T X)$ is matrix with principle diag elements of X & 0 everywhere else.

(*) links distances to actual coordinates.

recover X by multiplying D^2 with centring matrix C

$$C = \left(I - \frac{1}{N} \mathbb{1}\mathbb{1}^T\right)$$ $I \to N \times N$ identity matrix

$$-\frac{1}{2} C D^2 C = -\frac{1}{2} \left(I - \frac{1}{N} \mathbb{1}\mathbb{1}^T\right) \left[\text{diag}(X^T X) \cdot \mathbb{1}\mathbb{1}^T + \mathbb{1}\mathbb{1}^T \cdot \text{diag}(X^T X)^T - 2X^T X\right] \left(I - \frac{1}{N}\mathbb{1}\mathbb{1}^T\right)$$

simplifies to $\Rightarrow \boxed{-\frac{1}{2} C D^2 C = X^T X}$ //

$\therefore X$ can be obtained from Singular Value Decomposition [SVD]

$$SVD\left(-\frac{1}{2} C D^2 C\right) = SVD(X^T X) = U^T \Sigma V$$
$$\Rightarrow \boxed{X = \Sigma^{1/2} V}$$

$\Sigma \rightarrow$ diag. matrix $\therefore \Sigma^{1/2}$ = square root of diag. entries.

- Like in PCA, now select no of dimension d' from magnitudes of the singular values.
- orthogonal projection to $d' << d$; components can be done by setting all singular values in Σ beyond the first d'-dimensions to 0.

ISOMAP

PCA & MDS perform Linear mapping.

- performs non-linear mapping.
- Think of swiss-roll [unrolling the swiss roll]
- PCA & MDS squashes the swiss roll.
- Non Linear Mapping approaches use the MDS concept of dist. b/w samples

ISOMAP uses "Non-linearity hack" for MDS.

- Preserves non-linear manifold by replacing euclidian dist. with geodesic distances on manifold.
- Dist. is calculated as shortest path from a graph where each sample is connected to its neighbours.

Algorithm:

1. define graph with edge weights as euclidian dist. (using K-NN or fixed dist. threshold)
2. Distance matrix = all pairs-shortest paths b/w samples (Floyd Warshall algorithm / Dijkstra's algo)
3. Perform MDS on distance matrix.

LE (Laplacian Eigen Map)

- uses local distances (advances one step than using local distances)
- preserves local neighbourhoods instead of preserving global variance (like PCA & MDS)

PCA & MDS	LE
Large dist remain large	Local neighbourhoods remain together

- organises local neighbourhoods in a graph Laplacian (general purpose matrix representing graphs)
- more robust to noise than ISOMAP.

Algorithm:

1. Build adjacency graph (via K-NN or fixed dist. threshold) } can be replaced by learned edge weights
2. Weight the edges using sample similarity. (ex: using kernel)
   Similarities are called affinities.
3. Perform eigen decomposition of graph laplacian.
4. Project sample to lower space.

Learn edge weights using density forest

1. train density forest
2. use sample affinity, ex: relative freq. that both samples end up in same leaf node of trees.

Learned weights are advantageous as similarity measure better adapts to data.

Objective function $\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i' - x_j')^2 w_{ij} \rightarrow$ minimise

$x_i', x_j' \in \mathbb{R}^{d'}$

$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2}\right); x_i, x_j \in \mathbb{R}^d$

Solution where $x_1' = \cdots = x_N'$ is prevented with additional cond.

$\tilde{x}^T D \tilde{x} = 1$

where $\tilde{x} = (x_1', x_2', \cdots x_N')^T$

$D = \text{diag}\left(\sum_{i=1}^{N} w_{1i}, \cdots, w_{Ni}\right)$

rewriting obj function

$\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i - x_j)^T w_{ij} = 2 \cdot \left(\tilde{x}^T (D - W) \tilde{x}\right)$

$L = D - W \rightarrow$ one variant of graph laplacian

$\min \tilde{x}^T L \tilde{x}$ subject to $\tilde{x}^T D \tilde{x}$

$$\frac{\delta}{\delta \tilde{x}} \left(\tilde{x}^T L \tilde{x} + \lambda \tilde{x}^T D \tilde{x} \right) \doteq 0$$

$$\Rightarrow \boxed{D^{-1} L \tilde{x} = \lambda \tilde{x}}$$

∴ goal is to minimise obj. func. ⇒ smallest eigen value indicates sol.

- discard $\lambda = 0$ as they only indicate no. of independent components.

Manifold forests + Laplacian Eigenmaps

- L can be composed of any set of pairwise affinities.

Manifold forests have affinities defined on a density forest.

For each tree, define an affinity matrix W_t using dist. $d_t(x_i, x_j)$:

$$W_t = [w_{ij}]_t = \exp(-d_t(x_i, x_j))$$

Gaussian Affinity

$$d_t(x_i, x_j) = \begin{cases} \frac{(x_i - x_j)^T (x_i - x_j)}{\sigma^2}, & \text{if } \text{leaf}(x_i) = \text{leaf}(x_j) \\ \infty \; ; & \text{O.W} \end{cases}$$

Binary Affinity

$$d_t(x_i, x_j) = \begin{cases} 0 \; ; & \text{if } \text{leaf}(x_i) = \text{leaf}(x_j) \\ \infty \; ; & \text{O.W} \end{cases}$$

with binary affinities, nodes are permuted s.t. W_t is a block matrix

$$W_t = \begin{pmatrix} 1 & 1 & 1 & & & & & & \\ 1 & 1 & 1 & & & & & & \\ 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & & & & \\ & & & 1 & 1 & & & & \\ & & & & & 1 & 1 & 1 & 1 \\ & & & & & 1 & 1 & 1 & 1 \\ & & & & & 1 & 1 & 1 & 1 \\ & & & & & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Averaging $W = \sum_{t=1}^{T} W_t$ over forest removes block structure.
- Affinities are large for sample pairs within same leaf. Creates sense of locality which is learned from data.
- W can be used to construct L.

$$\boxed{L = I - D^{-1/2} W D^{-1/2}}$$

Obtain embedding to d' dimensions by:

1. Sort eigen val & vectors in increasing order
2. Discard all eigen vectors associated with eigen values $\lambda = 0$
3. lower-dim. coordinates of x_i are ith entries of eigen-vectors associated with d' smallest eigenvalues.

Take aways

- LE more robust to outliers than ISOMAP
- ISOMAP: shortcut on manifold can be created by few outliers.
  - ↳ reduces shortest path.
- Graph laplacian connects graphs with linear algebra. [tool in spectral graph theory]
- Ex: K-eigen vectors of largest eigen values of L partition Re sample neighbourhood graph. Run K-means assigns clusters.

Intro to Probabilistic Graphical models

- Look at relationship b/w random variables here
- Probabilistic models represent data properties in probabilistic formulations

start with joint distribution $p(x_1 \cdots x_N)$

Factorize using product rule

$$p(x_1 \cdots x_N) = p(x_N | x_1 \cdots x_{N-1}) \cdots p(x_2|x_1)\, p(x_1)$$

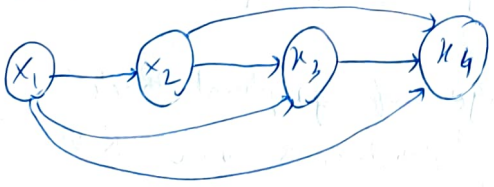
order can be arbitrary

hence even this is correct $\Rightarrow p(x_1 \cdots x_N) = p(x_3 | x_1, x_2, x_4 \cdots x_N) \cdots p(x_2|x_1)\, p(x_1)$

Directed edges ≡ conditional prob.

Undirected " = joint prob.

Ex:



x_1 conditions all variables → 3 outgoing edges

x_4 depends on all other variables → 3 incoming edges

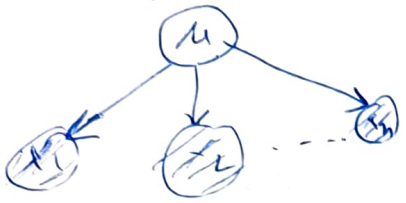
x_2 depends on x_1 & conditions x_3 & x_4 → 1 incoming edge & 2 outgoing Edge

Example 1 Parametric DE

Gaussian data with fixed σ^2 but unknown μ.

$\mu \rightarrow$ Random Variable

$x_1 \cdots x_N \rightarrow$ mutually independent observations



observed nodes are shaded.

Bayesian DE reverses the arrows to find $p(\mu | x_1 \cdots x_N)$ from likelihood $p(x_1 \cdots x_N | \mu)$ & prior $p(\mu)$.

plate notation simplifies above graph

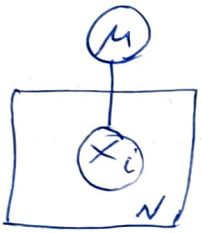


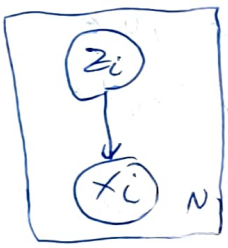
plate is short-hand notation for N independant, identically distributed (i.i.d) of same variables.

Ex2: Mixtures

- it is an overlay of multiple distributions.

$$p(x_i) = \sum_{z_i} p(x_i | z_i)\, p(z_i)$$

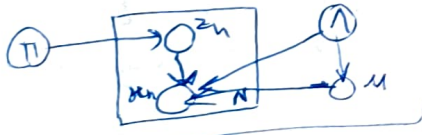
$z_i \rightarrow$ indicator variable



$p(x_i | z_i = k)$ generates x_i from k-th component.

Ex3: GMM with hyperpriors

$$p(x, z, \pi, \mu, \Lambda) = p(x | z, \mu, \Lambda)\, p(z | \pi)\, p(\pi)\, p(\mu|\Lambda) p(\Lambda)$$



Advantage of graphical model

- makes dependencies b/w R.V. more explicit through edges.
- variable independence tractability is better.

Conditional Independence

Independence enables factorization of models with many variables.

if $p(a|b,c) = p(a|c)$ then a is conditionally independant of b given c.

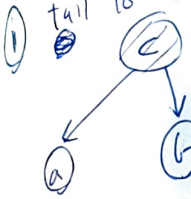
$a \perp\!\!\!\perp b|c$ ← mathematical notation.

Tricky on directed graphs / simpler on undirected graphs

Undirected graphs: Markov blanket enables inference.

Directed graphs

i) tail to tail



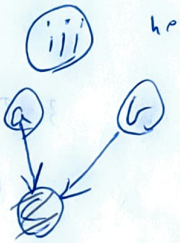
$= p(c)\ p(a,b|c)$

- if c is observed a, b become independent

ii) head to tail

$p(a)\ p(c|a)\ p(b|c)$

iii) head to head.



$p(a)\ p(b)\ p(c|a,b)$

a, b become conditionally dependant if c is observed

tail to tail

if c is unobserved $a \not\perp\!\!\!\perp b|c$ since $p(a,b) \neq p(a)\,p(b)$

$$p(a,b) = \sum_c p(c)\ p(a|c)\ p(b|c) \neq p(a)\ p(b)$$

if c is observed $a \perp\!\!\!\perp b|c$:

$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(c)\ p(a|c)\ p(b|c)}{p(c)} = p(a|c)\ p(b|c)$$

D-separation of var. in directed graph:

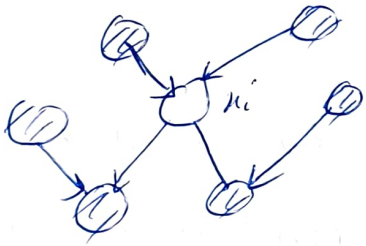
- D-sep. indicates weaker for sets of A, B, C; $A \perp\!\!\!\perp B|C$
- Var. in case observed.

Algo:

1. consider all path b/w A & B
2. A path is blocked if
   - node is in C & its arrows are tail to tail or head to tail.
   - node & its decendants are not in C & its arrows are head to head.
3. If all paths are blocked then A is d-sep. from B by C, i.e. $A \perp\!\!\!\perp B|C$.

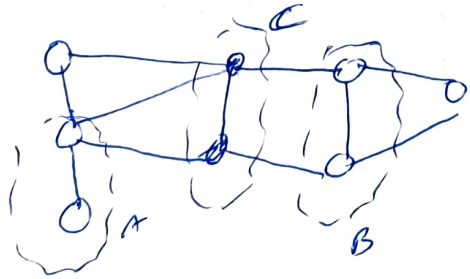
Markov Blanket: minimal set of nodes that isolate a node from rest of the graph. Include immediate parents, immediate children & co-parents of immediate children.

Directed graph

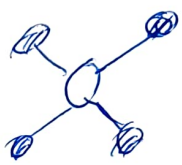


Undirected graph:

separate A & B by observed nodes for cond. independance



Markov blanket: includes set of neighbours of a node.



Hidden Markov Models (HMM)

Any distribution can be represented as a GMM. But not always practical.

Multiple linked GMM may be more efficient.

HMM → classical tool to process sequential data.

- can be seen as probabilistic state machine which maps observation sequences to states.
- states are hidden as user sees only i/p & o/p. but not states

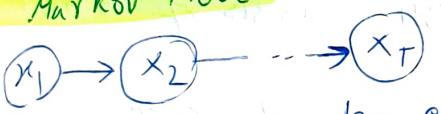
- HMMs are generative probabilistic models, hence possible to draw new samples from trained model.

Preliminary considerations on seq. data

1st order Markov Model

$$p(x_1, x_2 \dots x_T) = p(x_1) \cdot \prod_{t=2}^{T} p(x_t \mid x_1 \dots x_{t-1})$$

$$\stackrel{!}{=} p(x_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1})$$

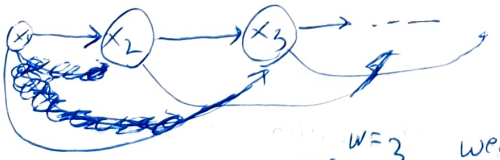


- inference can be done easily.
- ex: if $x_1 \dots x_T$ are measurements on diff. days then treating each x_t (meas.) separately doesn't make sense. Hence introduce dependency on the measurement of prev. day.

such model reaches performance saturation quickly.

Extend model to use prev. 2 days data.

$$p(x_1 \dots x_T) = p(x_1)\, p(x_2 \mid x_1) \prod_{t=3}^{T} p(x_T \mid x_{T-1}, x_{T-2})$$



Now if there are $w=3$ weather types then

1st order model should learn w^2 transitions from x_{t-1} to x_t

2nd " " " w^3 " " x_{t-2} & x_{t-1} to x_t

This approach cannot be scaled. [cannot be generalised]

HMM use a trick to keep no. of parameters tractable:

they use latent variables. (z_t)

↳ encode discrete states [April weather, Indian summer/seasons...]

Match seq. of days with particular weather to plausible seq. of states.

HMM joint distribution

$$p(x_1, \dots, x_T, z_1 \dots z_T) = p(\underbrace{x_1 \dots x_T}_{\text{obs. seq.}} \mid \underbrace{z_1 \dots z_T}_{\text{hidden state seq.}})\, p(z_1 \dots z_T) \quad (*)$$

Assumptions:

1. each obs. depends on single hidden state

$$p(x_1 \dots x_T \mid z_1 \dots z_T) = \prod_{t=1}^{T} p(x_t \mid z_t) \quad (1)$$

2. each state depends on predecessor

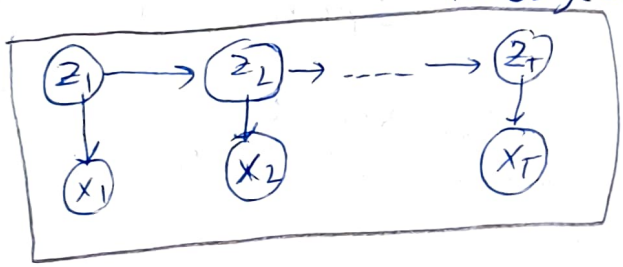
$$p(z_1 \dots z_t) = p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \quad (2)$$

From $(*)$ using (1) & (2)

$$\Rightarrow p(x_1 \dots x_T, z_1 \dots z_T) = p(z_1) \prod_{t=1}^{T} p(x_t \mid z_t) \prod_{t=2}^{T} p(z_t \mid z_{t-1})$$

($p(z_1)$: starting state prior; $p(x_t \mid z_t)$: prob. of seeing obs. x_t given state at time t; $p(z_t \mid z_{t-1})$: state transition prob.)

↳ factorized HMM consists of small terms.



can describe complex data in a compact representation via

1. distr. of starting state (z_1)
2. likelihood of state transitions $p(z_t \mid z_{t-1})$
3. likelihood of observation in each state $p(x_t \mid z_t)$.

Rabiner Notation

N states S_i

sequence of T observations; each observation is discrete symbol O_v from alphabet of size V.

HMM parameters $\rightarrow \lambda = (A, B, \pi)$

$A \rightarrow$ state transitions matrix from state S_i to S_j $\quad A = [a_{s_i s_j}] \in \mathbb{R}^{N \times N}$

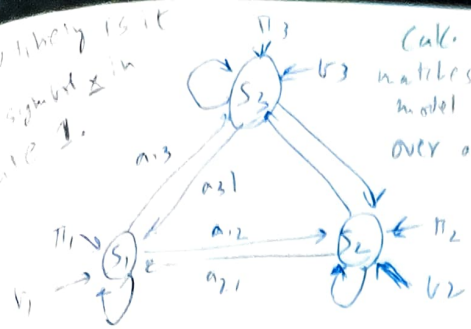
$$\sum_{j=1}^{N} a_{s_i s_j} = 1$$

$B \rightarrow$ matrix of symbol prob. per state S_i with $\sum_{v=1}^{V} b_{s_i}(O_v) = 1$

$B = [b_{s_i}(O_v)] \in \mathbb{R}^{N \times V}$ matrix of observation we can make

$\pi \rightarrow \pi = [\pi_{s_i}] \in \mathbb{R}^{N}$ vector of prob. that seq. starts in state i.

$$\sum_{i=1}^{N} \pi_{s_i} = 1$$



Calc. prob but input seq. matches the Training model by marginalising over all paths.

Evaluating HMM $\lambda = (A, B, \pi)$

requires 3 Algos:

1. Matching: prob. that obs belongs to λ. $P(x_1 \dots x_T | \lambda)$

FWD algorithm (bcwd algo)

2. Find Most likely state seq.

$\underset{z_i = s_i}{\operatorname{argmax}} P(x_1 \dots x_T, z_1 \dots z_T | \lambda)$

Viterbi algo

3. HMM training: determine λ from many training seq. $x_1 \dots x_T$

(training seq)

Baum-Welch formula [EM algo]

Examples/Applications

speech recognizers.

Any consist of large no of small HMMs. [nested HMM]

Matching: determines how well an observation matches to HMM.

conditional dependence of Algo 1 & 2 implies that they work on a trained model.

Algo 1: How well Matches an I/p seq. to specific HMM.

Calc $P(x_1 \dots x_T | \lambda)$ → model by marginalising over all state seq. $z_1 \dots z_T$

→ prob. of seq. $x_1 \dots x_T$ given λ

$$P(x_1 \dots x_T) = \sum_{z_1=s_1}^{s_N} \sum_{z_2=s_1}^{s_N} \dots \sum_{z_T=s_1}^{s_N} \pi_{z_1} b_{z_1}(x_1)\, a_{z_1 z_2} b_{z_2}(x_2) \dots a_{z_{T-1} z_T} b_{z_T}(x_T)$$

any state could be 1st, 2nd or last state respectively

$N \cdot N \cdot N \dots N$

⇒ computational cost of $O(N^T)$.

state seq. an input takes can have many possibilities.

$$P(x_1 \dots x_T) = \sum_{z_1=s_1}^{s_N} \sum_{z_2=s_1}^{s_N} \dots \sum_{z_T=s_1}^{s_N}$$

prob. of 1st in observ. → prob. of making 2nd obs. → $\pi_{z_1} b_{z_1}(x_1)\, a_{z_1 z_2} b_{z_2}(x_2) \dots a_{z_{T-1} z_T} b_{z_T}(x_T)$

π_{z_1}: prob to start in state z_1

$a_{z_1 z_2}$: next state could be any other state [prob of Trans. from z_1 to z_2]

$b_{z_2}(x_2)$

N possibilities

any state could be starting state

2nd state

last state

Factorized.

N^T possible paths

N^2 possibilities account for transition prob.

$$P(x_1 \dots x_T) = \sum_{z_1=s_1}^{s_N} \pi_{z_1} b_{z_1}(x_1) \left[\sum_{z_2=s_1} a_{z_1 z_2} b_{z_2}(x_2) \right] \dots \sum_{z_T=s_1}^{s_N} a_{z_{T-1} z_T} b_{z_T}(x_T)$$

output prob.

poss. to start in state 1

look for assignment to 2nd state

$O(N^2 T)$

Algorithm:

1. initialise

$t=1$ $\quad \alpha_1(z_1 = s_i) = \pi_{z_1} b_{z_1}(x_1)$

π_{z_1}: prob. we start in state z_1; $b_{z_1}(x_1)$: match symbol to state

$\alpha_1(z_1 = s_1) = 0.1$

$\alpha_1(z_1 = s_2) = 0.05$

$\alpha_1(z_1 = s_N) = 0.11$ ← read from list

2. Iteration: go from $T-1$ to t

$2 \le t \le T$; $\alpha_t(z_t = s_i) = \sum_{z_{t-1}=s_1}^{s_N} \alpha_{t-1}(z_{t-1}) \cdot a_{z_{t-1} z_t} \, b_{z_t}(x_t)$

- $\alpha_t(z_t = s_i)$: at time t we have prob. that we are in state s_i
- $\alpha_{t-1}(z_{t-1})$: prob. that in prev. time we were in prev. state
- $z_{t-1} = s_1$: we could have come from any other states.
- $a_{z_{t-1} z_t}$: transition prob. from (each) state to state t
- $b_{z_t}(x_t)$: match symbol to state t.

α → caching variable for path probability

3. summation over end points

$t = T$ $\quad p(x_1 \dots x_T) = \sum_{z_T = s_1}^{s_N} \alpha_T(z_t)$

sum over all possible paths.

Use this algo. to see how likely the seq. has been generated by the model.

Also for the training.

Backward algo: start at time T → go to 1.

1. init → caching var. β

$t = T$; $\beta_T(z_1 = s_i) = 1$ prob. at time T we are in state s_i.

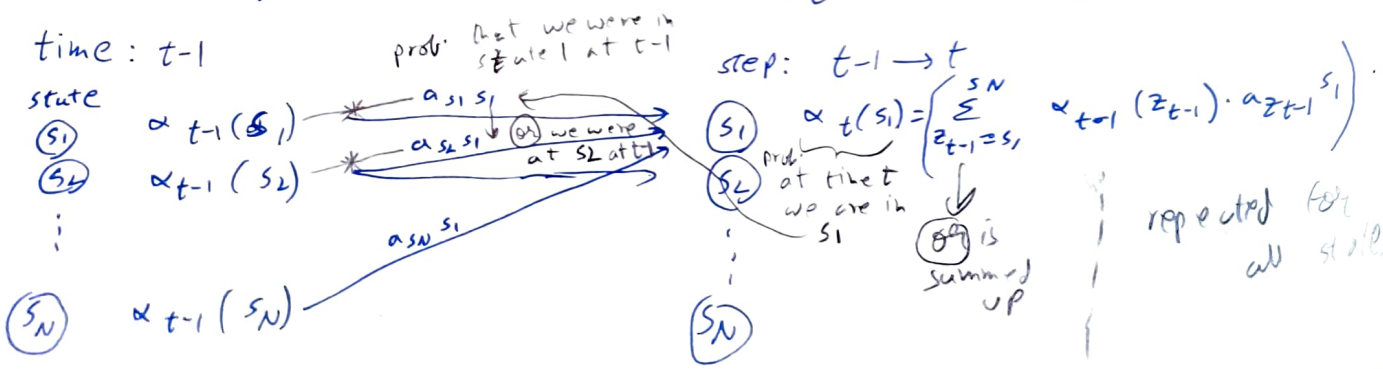
2. Iterate

$1 \le t \le t$ $\quad \beta_t(z_t = s_i) = \sum_{z_{t+1}=s_1}^{s_N} \beta_{t+1}(z_{t+1}) \, b_{z_{t+1}}(x_{t+1}) \, a_{z_t z_{t+1}}$

- $\beta_{t+1}(z_{t+1})$: prob. at time $t+1$ to be in that state
- $b_{z_{t+1}}(x_{t+1})$: o/p prob.
- $a_{z_t z_{t+1}}$: transition prob. from future state to current state.
- $z_{t+1} = s_1$: at $t+1$ we were in state i

BCKwd algo doesn't use ~~state~~ x_1 [ok for training use]

time: $t-1$ → step: $t-1 \to t$



Algo 2: searches for most likely state seq.

Find: $\arg\max_{z_t = s_j} P(x_1, \dots x_T, z_1, \dots z_T | \lambda)$ → from prob. of seq. of observation A seq. of states give trained model we seek the assignment of states to hidden variables that maximise prob. [most likely state seq]

$\delta_t(z_i)$ → caches prob. of most likely path to z_i within time 1.

$\psi_t(z_i)$ → caches predecessor state to reconstruct path.

Viterbi algo:

1. Init: $t = 1$: any state can be our starting state; starting prob; o/p prob.

$$\delta_1(z_1 = s_i) = \pi_{z_1} b_{z_1}(x_1)$$

$$\psi_1(z_1 = s_i) = 0 \rightarrow \text{no prev. state}$$

2. Iteration $1 \le t \le T$: prob. of most likely path that leads us in time t to state i

$$\delta_t(z_t = s_i) = \max_{s_1 \le z_{t-1} \le s_N} \left(\delta_{t-1}(z_{t-1}) \cdot a_{z_{t-1} z_t} \right) b_{z_t}(x_t)$$

$$\psi_t(z_t) = \arg\max_{s_1 \le z_i \le s_N} \delta_{t-1}(z_{t-1}) \cdot a_{z_{t-1} z_t}$$

3. Max over all end points.

$$p^* = \max_{s_1 \le z_T \le s_N} \delta_T(z_T)$$ → final most likely path is the end state that achieves max prob.

$$z_T^* = \arg\max_{s_1 \le z_T \le s_N} \delta_T(z_T)$$

↳ most likely end state.

4. Path reconstruction

$$z_t^* = \psi_{t+1}(z_{t+1}^*)$$

- $O(N^2 t)$: computational complexity
- Viterbi is also used in decoders

$\alpha_t(s_i)$ → sum of path prob. of all paths from time 1 to time t that are at time t in state s_i

$\beta_t(s_i)$ → sum of prob. of all paths from time T to time t that are at time t in state s_i

$\psi_t(z_i)$ → caches predecessor state to reconstruct path.

$\delta_t(s_i)$ → prob. of single most likely path from time 1 to time t that is at time t in state s_i

Algo 3: Training

Baum-Welch formulae = EM algorithm

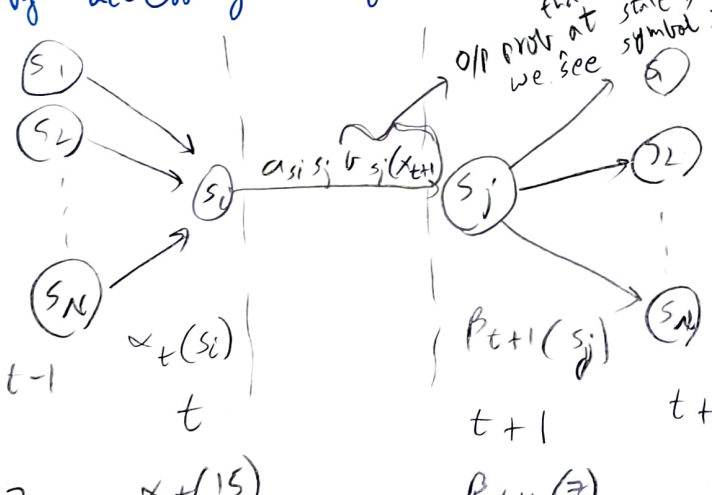
Expectations
- responsibilities are estimations for current state/state transition & associating expectations
- idea: FWD & BWD algo marginalise most parameters to isolate param. of interest.

Maximization:
updates model param.
idea: count relative freq. converges to local opt.

x_t is a position in an i/p seq. $x_1, \dots x_T$

However training is done over a dataset / many i/p sequences

- state seq. is hidden
- for training we have to update ex: update individual state transitions $a_{s_i s_j}$
- done by accessing $\alpha_t(z_t = s_i)$ & $\beta_{t+1}(z_{t+1} = s_j)$



ex: $a_{15,7}$ $\alpha_t(15)$ $\beta_{t+1}(7)$

STEP E likelihood of state transition $s_i \to s_j$ at time t

$$\xi_t(i,j) = p(z_t = s_i, z_{t+1} = s_j)$$

$$= \frac{\alpha_t(s_i)\, a_{s_i s_j}\, b_{s_j}(x_{t+1})\, \beta_{t+1}(s_j)}{\sum_{z_t = s_i}^{s_N} \sum_{z_{t+1} = s}^{s_N} \alpha_t(z_t) \cdot a_{z_t, z_{t+1}}\, b_{z_{t+1}}(x_{t+1}) \cdot \beta_{t+1}(z_{t+1})}$$

likelihood for being in S_i at time t: marginalize S_j

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

expected n.o. of transitions $S_i \to S_j$ at all times: $\sum_{t=1}^{T} \xi_t(i,j)$

" " " times S_i is visited: $\sum_{t=1}^{T} \gamma_t(i)$

Up M

$\bar{\pi}_{Si} = \gamma_1(i) \to$ expected # times in S_i in time 1 over all training data

Update $\bar{a}_{S_i S_j}$ (state transition prob.)

$$\bar{a}_{S_i S_j} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\text{exp. \# of trans. } S_i \to S_j}{\text{exp \# of times in state } S_i}$$

update O/p prob.

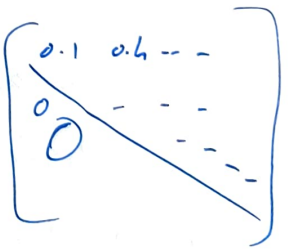
$$\bar{b}_{Si}(O_v) = \frac{\sum_{t=1}^{T} \gamma_t(i)\, \mathbb{I}(X_t, O_v)}{\sum_{t=1}^{T} \gamma_t(S_i)} = \frac{\text{exp. \# times in } S_i \text{ \& obs. } O_v}{\text{expected \# times in } S_i}$$

HMM = ergodic if every state can be reached from every other state

[speech is not ergodic]

Left → Right HMMs are mostly used. [$a_{S_i S_j} = 0$ if $i > j$]

(only Fwd & self loops allowed)

$A =$ 

- GMM & HMM training work well together
  - both probabilistic models
  - " trained using EM.

[can nest these eqv. together]

- good init. can improve results.

Markov Random Fields

- Undirected graphical model
- used for labeling tasks: segmentation, stereo matching, multiview stitching.
- idea: consider unknown labels as a field of RV with conditional independance,

Joind distr. & objective

$$p(\underbrace{x_1 \dots x_N}_{\text{observations}}, \underbrace{z_1 \dots z_N}_{\text{hidden variables}}) = p(x_1 \dots x_N \mid z_1 \dots z_N)\; p(z_1 \dots z_N)$$

MRF inference seeks posterior of hidden var.

$$p(z_1 \dots z_N \mid x_1 \dots x_N) = \frac{p(x_1 \dots x_N \mid z_1 \dots z_N)\; p(z_1 \dots z_N)}{p(x_1 \dots x_N)}$$

$$= \frac{p(x_1 \dots x_N \mid z_1 \dots z_N)\; p(z_1 \dots z_N)}{\sum_{z_1} \dots \sum_{z_N} p(x_1 \dots x_N \mid z_1 \dots z_N)\; p(z_1 \dots z_N)} \rightarrow \text{marginalisation of numerator}$$

here evidence needs exponential calculations as we have to sum over all possible combinations of variables, hence we will seek Max. likelihood sol.

then - we will show the denominator is const. WRT z.

- Likelihood & prior must also be simplified.

- Ex: in images $x_i \rightarrow$ i/p pixel (observations (x_i) if x_i is modelled as variable)
  $z_i \rightarrow$ label we want to assign (only z's $= 2^8 \cdot 2^8$)
  for low res. images 256×256 pixels $\rightarrow 2 \cdot 2^{16}$ variables.

Joint distr. is too complex [look at graph in slide]

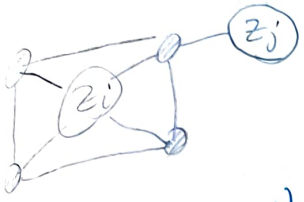
HMM & MRF make model tractable with simple dependency struct.

Factorization of joint dist

1. Each obs. depends only on 1 hidden var.
2. (a) Each z depends only on its neighbours i.e. $z_i \perp\!\!\!\perp z_j \mid \mathcal{N}(z_i)$
   for all $z_j \notin \mathcal{N}(z_i)$. [$\mathcal{N}(z_i) \rightarrow$ set of neighbours of z_i]
   z dependent on neighbours but independent of non-neighbours

(17) neighbourhood ~~Set~~ of z_i should be split in pairs for tractability such that prior $p(z_1 \dots z_N)$ decomposes to

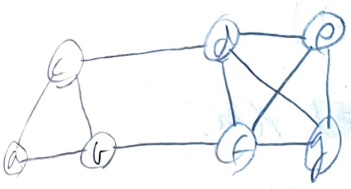
$$p(z_1 \dots z_N) = \prod_{z_k \in N(z_i)} p(z_i, z_k) \prod_{z_k \in N(z_j)} p(z_i, z_k)$$



point 2a → highlights neighbourhoods determine factorization..

Undirected graphs factorize into maximal cliques.

Clique → fully connected subgraph.

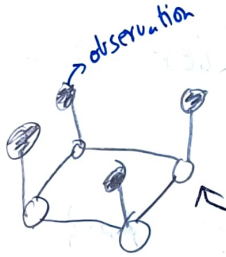


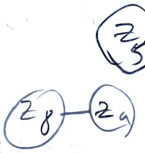
dependency structure.

$$p(a,b,c,d,e,f,g) = p(a,b,c)\, p(c,d)\, p(d,e,f,g)$$

maximal fully connected subgraphs

(2b) → implies we model dependencies with cliques ≤ 2.





← clique of size 1

← clique of size 2

} count of hidden var.

These assumptions result in posterior of ~~given~~ labels given observations

$$p(z_1 \dots z_N \mid x_1 \dots x_N) = \frac{\prod_{i=1}^{N} p(x_i \mid z_i) \prod_{z_k \in N(z_i)} p(z_i, z_k)}{\sum_{z_1} \dots \sum_{z_N} \prod_{i=1}^{N} p(x_i \mid z_i) \prod_{z_k \in N(z_i)} p(z_i, z_k)}$$

inference for MRF is to find [Max Likelihood]

$$z^* = \arg\max_{z} p(z_1 \dots z_N \mid x_1 \dots x_N)$$

- inference : tractable
- learning : intractable
  ↳ to find $p(x_i \mid z_i)$ & $p(z_i, z_j)$ these are difficult to handcraft

with learning being intractable & difficult to hand-craft inference is impossible.

Trick : use potentials instead of distributions.

Hammersley-Clifford Theorem

- define $p(x_i|z_i)$ & $p(z_i, z_j)$ optimally by using energy terms
- Hammersley-Clifford theorem allows to move b/w MRF & Gibbs Random Field (GRF) which wraps energy terms in exponentials.

$\psi(c) \rightarrow$ potential functions over cliques of var. C.

$\psi(c) \geq 0$ [non-ve]

MRF = GRF [consisting of $\psi(c)$]

Instead of specifying $p(x_i|z_j)$ & $p(z_i, z_j)$ set $\psi(c)$ where $C = \{z_i\}$ are maximal cliques of hidden variables.

Potential functions are often more easier to choose, in many applications.

potential functions $\psi(c)$ over set of nodes $C = \{z_i\}$ are set of exponential functions

$$\psi(c) = \exp(-E(c)) \; ; \; E(c) \rightarrow \text{energy function}$$

lower energy = closer to goal

∴ factorized MRF is

$$p(z_1 \cdots z_N | x_1 \cdots x_N) = \frac{1}{Z} \prod_c \psi(c)$$

where $Z = \sum_z \prod_c \psi(c) \rightarrow$ called partition function

$Z \rightarrow$ enables use of any non-negative func. for potentials

$Z \rightarrow$ intractable, but is const. when maximizing $p(z)$

∴ it can be ignored.

Inference with potentials

$$z^* = \arg\max_z p(z_1 \cdots z_N | x \cdots x_N) = \arg\max_z \frac{1}{Z} \prod_c \psi(c)$$

Inference via Gibbs Sampling (MCMC based)

1. select node z_i (randomly / follow a pattern)
2. get current labels from all neighbours (Markov Blanket)
3. Assign a new label to z_i with minimal energy.
4. Goto 1 for set iterations / until convergence.

↳ go for min. energy / aim for most likely sol. instead of sampling any sol.
This makes sol. locally optimal.
Graph cuts find globally optimal sol.

Unary potentials: potentials with energy terms $E(x_i, z_i)$

Pairwise " : " " " " $E(z_i, z_j)$

Optimization analogy: unary potentials the data term & the pairwise potentials are regularizer.

Example — denoising binary img

energy func. for obs-hidden connections

$$E(x_i, z_i) = -\eta x_i z_i$$

$\eta \rightarrow$ weight constant
$x_i \rightarrow$ ith i/p pixel
$z_i \rightarrow$ ith hidden variable

energy func. for hidden-hidden connections

$$E(z_i, z_j) = -\beta z_i z_j \qquad \beta \rightarrow \text{weight const.}$$

- value range of labels & i/ps constraints the potential functions.

if it is binary image then there are 2 possibilities

identical labels $\rightarrow E(x_i = -1, z_i = -1) = -\eta$ ← so we want to encourage this is lowest energy hence this is favoured more

diff. labels $\rightarrow E(x_i = -1, z_i = 1) = +\eta$

same logic holds for hidden-hidden connections

returned pixel label is the denoised image

unit energy term

$E(z_i) = \eta \sum_{z_i} z_i + \beta \sum_{z_i, z_j \in N(z_i)} z_i z_j - \eta \sum_{(x_i, z_i)} x_i z_i$

z_i: unit term to decide a label

Non-binary labeling

Unary term could be $\eta \|x_i - y_i\|_2^2$. low energy sol. $\vec{z}$ that are close to i/p $\vec{x}$.

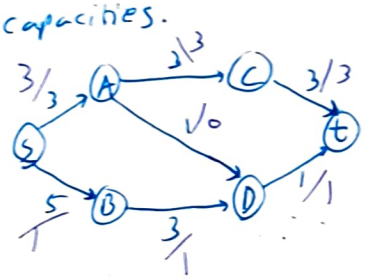
Pairwise energy term could be $\beta \|z_i - z_j\|_2^2$. low energy sol $\vec{z}$ where neighbouring pixels y_i, y_j are more similar.

Inference via Min-cut & Max flow.

What is max-flow?
- combinatorial std. problem
- solved in polynomial time.

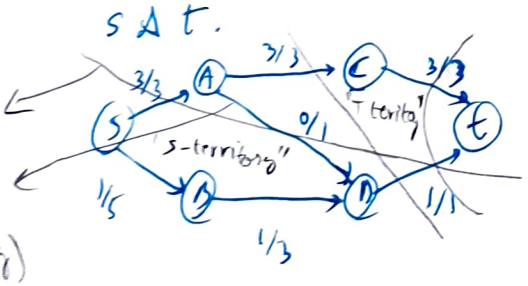
graph
source node
sink node
+ve edge weights (tube capacities) (t) if edge weights ↓ per unit time.

Determine max throughput of water from source(s) to sink(t) where edge weights are tube capacities.



→ pumped units
Max flow = 4 //

What is Min-cut?
- seeks smallest sum of edges to disconnect s & t.
- select minimal set of maxed out edges that disconnects s & t.

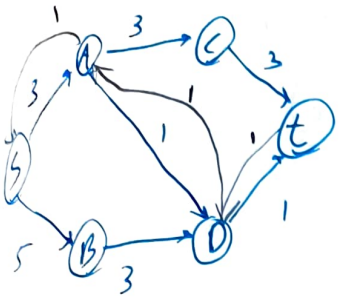


do not count edges from t→S (territory)

edge weight of min cut = 3+1=4

Ford fulkerson Algo:

Finds out Max flow



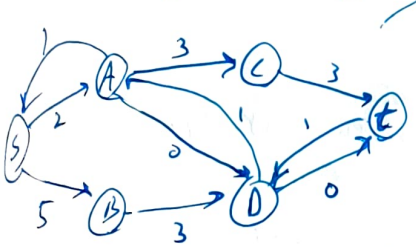
1. Greedily search shortest path
2. Max-out flow capacity on that path, reduce edge weights
3. Introduce backward edges to undo greedy dead ends, goto (1) if s-t path left.

Undo bad selection from greedy selection

shortest path: $s \to A - D - t$ [5]
path flow = 1

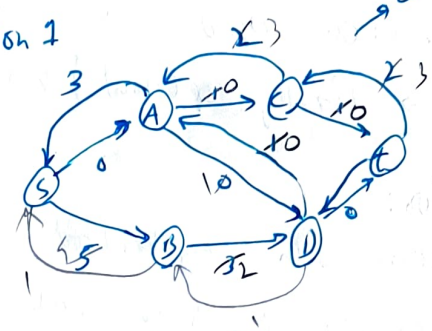
capacity of 3 becomes 2
1 " 0

→ after iteration 1



shortest path $s \to A \to C \to t$
path flow = 2

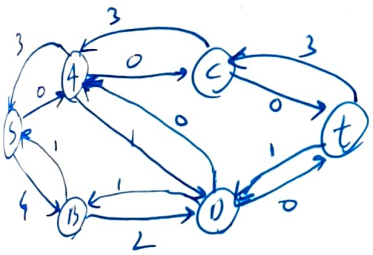
→ after iteration 2



iteration 3

shortest path: $s \to B - D - A - C - t$ (11)

path flow: 1



total flow through iterations = 1 + 2 + 1 = 4

min cut = $D \to t$

either $(s-A, A-C, C-t)$

MRF inference via Graph cuts

MRF → GRF → energy minimization

Objective of MRF inference task: Find optimal label assignments

idea of MRF inference via Graph cuts: transform maximum likelihood task to equivalent min cut-task.

Procedure: construct directed graph with edge weights such that graph's min cut provides sol. to max. likelihood task.

Requirements for graph construction / constraints for algo

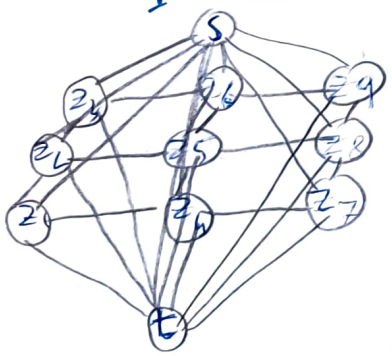
1. Binary labels
2. Max. clique size of 2
3. Pairwise energy terms satisfy submodularity condition

$$E(0,0) + E(1,1) \leq E(0,1) + E(1,0)$$

Under these 3 constraints algo. finds
- globally optimal sol.
- in polynomial time.

α expansion used to extend this method to non-binary labeling
↳ locally optimal [but within guaranteed margin around global optimum]

Construction:
1. start with neighborhood relationship of hidden var z_i.
2. Encode optimization prob.
   a. add source node S, identified with label 0
   b. " sink " t, " " " 1
   c. connect nodes with appropriately chosen weights.
3. Calc. min. cut b/w S & t
4. Assign "0" to nodes connected to S
   "1" to nodes connected to t



Choosing edge weights: Additivity of graphs

Equivalance b/w MRF inference & graph cut ensured by construction of edge weights.

Every energy term can be transformed to graph shippet because energy minimization via min. cut is homomorphic.

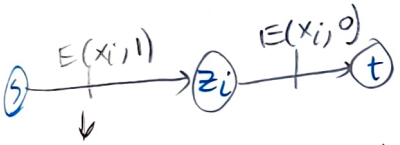
i.e.

if G_1 encodes $\min(E_1)$ & G_2 encodes $\min(E_2)$

then $(G_1 \cup G_2)$ encodes $\min(E_1 + E_2)$

∴ if we encode each single energy term to a graph such that graph's min cut encodes the min. energy, then also min cut of sum / union of all individual graphs will provide sol^n to min energy of sum of terms.

Encoding unary term



cut b/w s & z_i assigns label $z_i = t = 1$

∴ cost is $E(x_i, 1)$ → relates observation x_i to 1

smaller graph

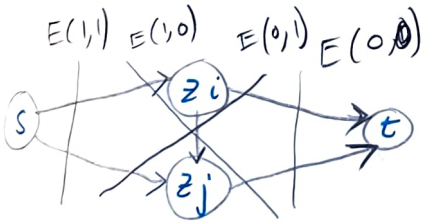
$s = 0$ & $t = 1$

if $E(x_i,1) > E(x_i,0)$

if $E(x_i,0) > E(x_i,1)$

Smaller graphs computationally cheaper but shift minimum's offset.

Encoding pair wise term

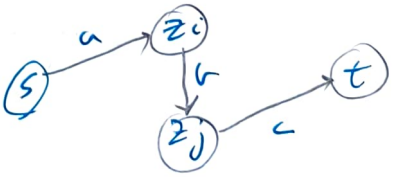


$s = 0$ & $t = 1$

smaller graph

if $E(1,0) > E(0,0)$

$E(1,0) > E(1,1)$



a, b, c obtained from

$a + k = E(1,1)$

$c + k = E(0,0)$

$b + k = E(0,1)$

$a + c + k = E(1,0)$

$k \rightarrow$ const offset.

α - expansion

1. select 1 specific label α from the label set
2. Fix hidden variables that already have label α
3. seek lowest energy labeling that switch atleast 1 other hidden var. to α
4. keep expanded α labeling if its energy is lower than current energy.