SLT 2018 Cheat Sheet

Lecture 1

Likelihood of dataset: $P(X|\Theta) = \prod_{i \leq n} p(x_i|\Theta)$ and we estimate Θ by maximizing the likelihood:
$\hat{\Theta} := argmax_{\Theta} P(X|\Theta)$

Markov Inequality: $P(X > \epsilon) \leq \frac{\mathbb{E}[X]}{\epsilon}$

Chebyshev inequality: $P(|\epsilon - \mu| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}$

Locally linear embedding: 1) Input: $\{X_i \in \mathbb{R}^D\}_{i \leq N}$, assumed to lie on a d-dimensional manifold with $d << D$.find embedding of this for X_i to corresponding $Y_i \in \mathbb{R}^d$, condition to preserve neighborhood relations. 2) X_i associated with set K nearest neighbors X_j such that X_i lies approximately in the affine subspace containing its neighbors. First identify how close each data point is to each of its neighbors by way of coefficients W_{ij}, which approximate X_i by the affine combination $\sum_j W_{ij} X_j$. This amounts to minimizing the least-squares cost function $\mathcal{E}(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2$, 3) under the constraints that $\sum_j W_{ij} = 1$ for each $1 \leq i \leq N$ and $W_{ij} = 0$ for each $1 \leq i, j \leq N$ such that X_j is not a neighbor of X_i. Note that each summand in $\mathcal{E}(W)$ concerns only one row of W, so the function may be minimized separately for each i. 4) The low-dimensional embedding is then obtained by preserving the neighborhood relations captured by W_{ij}, by minimizing a similar cost function, over Y with W fixed, $\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2$, under normalization constraints over Y_i fixing their mean to 0 and their covariance matrix to the identity, without loss of generality. 5) Minimizing first cost function amounts to finding, for each i, a normalized vector w_j, $1 \leq j \leq K$, which minimizes the quadratic form $w^\top C w$ with $C_{jk} = (X_i - X : n_i[j])^\top (X_i - X_{n_i[k]})$ for $1 \leq j, k \leq K$. Here n_i is the array of neighbors of X_i. Such a vector can be found by solving the linear system of equations $\sum_k C_{jk} w k = 1$ and then normalizing w. The weight matrix is then obtained by $W_{i,n_i[j]} = w_j$ and $W_{ij} = 0$ otherwise. 6) Second cost function is $\sum_{i,j} M_{ij} Y_i^\top Y_j$, where $M = (I - W)^\top (I - W)$. Minimizer given by d eigenvectors of M with the lowest positive eigenvalues, the all-ones eigenvector with eigenvalue 0 (resulting from the sum-to-one constraint) omitted. The ith component of the jth eigenvector gives the jth component of Y_i, and minimal value given by the sum of the corresponding eigenvalues.

Lecture 2 - Max Ent Inference

Self-Information/Surprise I(C): $I(C) = -log(p(C))$,
Properties: $I(C) = I(A \cap B) = I(A) + I(B)$ A,B independant.
Entropy H(X): Is the expected value of self-information of a random variable:
$H(X) = \sum_{x \in \omega} p(x) I(x) = -\sum_{x \in \omega} p(x) log(p(x))$
$H(X) \geq 0, H(X) \leq log(|\Omega|)$ equal if p uniform.
Entropy can be interpreted as choosing between $2^{H(X)}$ equal probable outcomes.

Joint Entropy X,Y: $H(X,Y) = \sum_{x,y \in \Omega} p(x,y) log(\frac{1}{p(x,y)})$,
and if X,Y indp. $H(X,Y) = H(X) + H(Y)$
Maximum Entropy: $max_{p(x)} - \sum_{\mathcal{X}} p(x) log(p(x))$
constraint by the expected values we know: $E[r_j(X)] = \mu_j$
Gibbs distribution maximizes Entropy:
$p(x) = \frac{exp(-\sum_j \lambda_j r_j(x))}{\int_{x' \in \mathcal{X}} exp(-\sum_j \lambda_j r_j(x'))dx}$ The random variable x can also denote some solution in a discrete optimization problem or so.

Kullbeck-leibler: $KL(p||q) = \mathbb{E}[log(\frac{p(x)}{q(x)})]$

Lecture 3 - Max Ent Clustering

Overview: Probabilistic centroids y_α and probabilistic assignments $P_{i\alpha}$ of object i to cluster α. Assumption that data $\mathcal{X}$ and possible labels c are random variables.
Posterior: $P(c|\mathcal{X}, \mathcal{Y})$ for the labels c in space $\mathcal{C}$ given data $\mathcal{X}$ nad fixed centroids $\mathcal{Y}$. Constraint that
$\mathbb{E}_{P(c|\,\mathcal{X},\mathcal{Y})}\, \mathcal{R}(c, \mathcal{X}, \mathcal{Y}) = \mu$ is a constant fixxed value.
Kmeans: In the case of kmeans we have
$\mathcal{R}^{km}(c, \mathcal{Y}) = \sum_{i \leq n} \left\| x_i - y_{c(x_i)} \right\|^2$. We then use gibbs
$P(c|\,\mathcal{X}, \mathcal{Y}) = \frac{exp(-\mathcal{R}(c, \mathcal{X}, \mathcal{Y})/T}{\sum_{c' \in \mathcal{C}} exp(-\mathcal{R}(c', \mathcal{X}, \mathcal{Y})/T}$
Ising Model in Image: Image Truth: $x_i \in \{-1, +1\}$. But what we have is $y \in \{0, 1\}$. And we think about this like: "What image x could have generated best our noisy image y": $p(x, y) = p(x)p(y|x)$.
Prior: $p(x) \in \frac{1}{Z_0} exp(-E_0(x))$ with
$E_0(x) = -\sum_{i=1}^{D} \sum_{j \in nbr_i} W_{ij} x_i x_j$ most of the time we have $W_{ij} = 1$ meaning that if x_i its neighbors x_j are equal we get higher exponential value and therefore higher probability. So p(x) obviously sympathizes with equal valued pieces.
And therefore how likely is the image y if our base image was x?(likelihood): $p(y|x) = \prod_i p(y_i|x_i) = exp(\sum_i -L_i(x_i))$
Posterior of Ising: Most of the time we're interested in x. And therefore our posterior looks like this:
$p(x|y) = \frac{1}{Z} exp(-E(x))$ with $E(x) = E_0(x) - \sum_i L_i(x_i)$
Gibbs sampling Idea: We sample new variables conditioned on all other variables: $x_1^{s+1} \sim p(x_1|x_2^s, x_3^s) = p(x_i|x_{-i})$
Gibbs sampling in Ising Model:
$p(x_t|x_{-t}, \theta) = \frac{1}{Z} \prod_{s \in nbr(t)} \psi_{st}(x_s, x_t)$ and in the ising model we have edge potential of: $\psi_{st}(x_s, x_t) = exp(J x_s x_t)$ so with $x_i \in \{0, 1\}$
we get the full conditional as: $p(x_t = +1|x_{-t}, \theta) = \frac{\prod_{s \in nbr(t)} \psi_{st}(x_t = +1, x_s)}{\prod_{s \in nbr(t)} \psi_{st}(x_t = +1, x_s) + \prod_{s \in nbr(t)} \psi_{st}(x_t = -1, x_s)} = \frac{exp(J \sum_{s \in neigh(t)} x_s)}{exp(J \sum_{s \in neigh(t)} x_s) + exp(-J \sum_{s \in neigh(t)} x_s)} =$
$\frac{exp(J\eta_t)}{exp(J\eta_t) + exp(-J\eta_t)} = \frac{1}{1 + e^{-2J\eta_t}} = sigmoid(2J\eta_t)$
Metropolis Hastings: Idea: At each step we propose to move from state x to a new state x' with probability $q(x'|x)$ where q is called the proposal distribution. Having proposed to move to x' we then decide whether to accept this proposal or not according to some formula, (ensuring that the time we spend in x' is according to $p^*(x)$.
Accepting propability is for symmetric case($q(x'|x) = q(x|x')$): $r = min(1, \frac{p^*(x')}{p^*(x)})$, non-sym: $r = (1, \alpha), \alpha = \frac{p^*(x')q(x|x')}{p^*(x)q(x'|x)}$
Why MH is useful is because we do not have to know the normalization constant of $p^*(x') = \frac{1}{Z}\hat{p}(x')$ the Z cancel in the $\alpha-$equation.
Metropolis Sampler for Clusering:
Input: n objects, cost function $\mathcal{R}(c, Y, X)$, Output: partition c: {objects} → {clusters }. Algorithm:

1. Initialize $c(i) \in \{1, ..., k\} randomly$ and then repeat:
2. draw $c' \sim Q(c)$ where Q(c) is proposal distribution
3. $p \leftarrow min\{exp(-(\mathcal{R}(c', Y, X) - \mathcal{R}(c, Y, X))/T), 1\}$
4. draw $b \sim Bernouilli(p)$ if b=1 $c \leftarrow c'$
5. $t \leftarrow t + 1$ until convergence

Gibbs Sampling for Clustering:
Input: n objects, cost function $\mathcal{R}(c, Y, X)$, Output: partition c: {objects} → {clusters }. Algorithm:

1. Initialize $c(i) \in \{1, ..., k\} randomly$ and then repeat:
2. draw $i \in 1, ..., n$ randomly
3. $c(i) \sim P(c(i)|c(1), ..., c(i-1), c(i+1), ..., c(n))$
4. $t = t + 1$
5. until joint distribution $P(c(1), ..., c(n))$ converged.

Image denoisening / Sampling

Model: Noisy image $y = (y_1, \ldots, y_n), y_i \in \{\pm 1\}$. Find a denoised image $x = (x_1, \ldots, x_n)$ by minimizing the energy function: $E(x, y) = -h \sum_i x_i - \beta \sum_{i,j \in N_i} x_i x_j - \eta \sum_i x_i y_i$
Heatbath: Similarly to the Metropolis algorithm, at each iteration one cell is chosen randomly and flipped with a probability, which now takes the form of a sigmoid function $p = \frac{1}{2}[1 - \tanh(\frac{1}{2T}(E(x', y) - E(x^{(i)}, y)))] = \frac{\exp(-E(x',y)/T)}{\exp(-E(x',y)/T) + \exp(-E(x^{(i)},y)/T)}$.
Simulated Annealing: Finds the global mode of a probability distribution, in this case the global minimum of the energy function, by slowly decreasing the temperature in each transition. The other steps are the same as the Metropolis algorithm
Parallel Tempering: Runs N different simulations of the Metropolis algorithm at different temperatures, and decides either to continue each simulation separately or to swap the outputs of two simulations with probability
Deterministic Annealing: Introduces a possibly continuous distribution p_i on the possible space of codevectors y_i, called the mass of effective cluster i, that generalizes in the case of finite codevectors the fraction of clusters with the same codevector. The masses are treated as a prior probability over the space of codevectors and thus constrained to sum to 1.

The corresponding Gibbs distribution for the codevector y_i associated to the datum x is the tilted distribution

$$p(y_i|x) = \frac{p_i e^{-d(x,y_i)/T}}{Z_x}$$

where d is the chosen distortion measure, T is temperature and $Z_x = \sum_x p_i e^{-d(x,y_i)/T}$ is the partition function.

Algorithm splits the cluster in question by randomly perturbing its centroid. The critical temperature for cluster i is calculated in the paper is twice the highest eigenvalue of the covariance matrix of cluster i, given by $C_{x|y_i} = \sum_x p(x|y_i)(x-y_i)(x-y_i)^\top = \sum_x \frac{p(x)p(y_i|x)}{p(y_i)}(x-y_i)(x-y_i)^\top = \frac{1}{p(y_i)}\sum_x p(x)p(y_i|x)(x-y_i)(x-y_i)^\top$.

At each step, repeat for a fixed temperature T and for each effective cluster i the updates

$$p(y_i|x) = \frac{p(y_i)e^{-d(x,y_i)/T}}{\sum_j p(y_j)e^{-d(x,y_j)/T}}$$

$$p(y_i) = \sum_x p(x)p(y_i|x)$$

$$y_i = \frac{\sum_x xp(x)p(y_i|x)}{p(yi)}$$

until convergence, and then cools down the temperature and if the critical temperature of cluster i is reached and $K < Kmax$, it splits cluster i into two, creating a new cluster whose centroid is a random perturbation of y_i and whose conditional and marginal probabilities are all initially evenly partitioned with those of i. For a finite dataset, $p(x) = 1/N$

Clustering distributional Data

Non Parametric: Notation: We have n different objects $\{x_i\}_{i=1}^n$. And m possible feature values: $\{y_j\}_{j=1}^m$. An occurence (dyad) is a pair $(x_i, y_j) \in \mathcal{X} \times \mathcal{Y}$. Our dataset is then defined as such occurences: $\mathcal{Z} = \{(x_{i(r)}, y_{j(r)})\}_{r=1}^l$ which is for some specific x_i we have l possible feature values y_j. We assume that observations come from a specific cluster: $(x_i, y_j) \sim P((x_i, y_j)|c(x_i), Q(y_j|c(x_i)))$. $Q(y_j|\alpha)$ is the fature distribution in the specific cluster α.
Likelih.: $\mathcal{L}(\mathcal{Z}) = \prod_{i\le n}\prod_{j\le m} P((x_i,y_j)|c(x_i),Q)^{l\hat{P}(x_i,y_j)}$. Where $l\hat{P}(x_i,y_j)$ is the number of times object x_i exhibits feature value y_j. The empirical distribution $\hat{P}(x_i,y_j) = \frac{1}{l}\sum_{r\le l}\delta(x_i,x_{i(r)})\cdot\delta(y_i,y_{i(r)})$ So just counting how many times feature y_j occurs together with patch x_i.
Risc function: We then have $R^{hc}(c,\mathcal{Q},\mathcal{Z}) = -log(\mathcal{L}) = -\sum_{i\le n}\sum_{j\le m} l\hat{P}(x_i,y_j) log(P((x_i,y_j)|c(x_i),Q)$.
This then simplifies to the below in the table kullback-leibler divergence. (As objective function)

	k-means clustering	histogram clustering
data	$\{x_1,...,x_n\}$	$\{p(.\|1),...,p(.\|n)\}$
centroids	$\{y_1,...,y_k\}$	$\{q(.\|1),...,q(.\|k)\}$
distortion	$\|\|x_i - y_{c(i)}\|\|^2$	$D^{KL}(p(\cdot\|i)\|\|q(\cdot\|c(i)))$
costs	$\sum_{i\le n}\|x_i - y_{c(i)}\|^2$	$\frac{l}{n}\sum_{i\le n} D^{KL}(p(\cdot\|i)\|\|q(\cdot\|c(i)))$

lilekihood of 1 observation: $P(x,y|c,\theta) = P(x)P(y|c(x))$
Posterior: $P(c,\theta|n) \propto \prod_x\prod_y (P(x)P(y|c(x)))^{n(x,y)}$ given the loglikelihood: $L(c,\theta;n) = \sum_x n(x)\left[\sum_y \hat{P}(y|x)\log P(y|c(x)) + \log P(x)\right] + \log P(c)$ where θ consists of the model parameters $P(x)$ and $P(y|c)$, $n(x)$ the number of xs observed, $n(x,y)$ the number of (x,y)s observed and $\hat{P}(y|x)$ the empirical probability of observing feature y given object x. The resulting stationarity equations for MAP estimation for the parameters (c,θ) are $\hat{P}(x) = \frac{n(x)}{\sum_{x'} n(x')}$
$\hat{P}(y|c) = \sum_{x:\hat{c}(x)=c}\frac{n(x)}{\sum_{x':\hat{c}(x')=c} n(x')}\hat{P}(y|x) \equiv \sum_x \hat{P}(x|c)\hat{P}(y|x)$
$\hat{c}(x) = \arg\min_a\left[\sum_y \hat{P}(y|x)\log\hat{P}(y|c(x)) + \log P(c(x)=a)\right]$
In the case of DA, the hard cluster assignments $c: x \mapsto c(x)$ are replaced with a probabilitistic assignment $\hat{P}(c(x)=a|\theta) = \frac{\hat{P}(a)\exp\left(-n(x)D\mathrm{KL}(\hat{P}(\cdot|x)\|\hat{P}(\cdot|a))/T + \log P(c(x)=a)\right)}{\sum_{b=1^K}\hat{P}(b)\exp\left(-n(x)D_{\mathrm{KL}}(\hat{P}(\cdot|x)\|\hat{P}(\cdot|b))/T + \log P(c(x)=b)\right)}$,
$\hat{P}(a) = \sum_x \hat{P}(x)\hat{P}(c(x)=a)$ after which equation (5) becomes $\hat{P}(y|c) = \sum_x \hat{P}(x|c)\hat{P}(y|x) = \frac{1}{\hat{P}(c)}\sum_x \hat{P}(x)\hat{P}(c|x)\hat{P}(y|x) = \frac{\sum_x \hat{P}(x)\hat{P}(c|x)\hat{P}(y|x)}{\sum_x \hat{P}(x)\hat{P}(c|x)}$,

Parametric

To get a smooth histogram. We introduce a parametric distribution to solve this. $p(y|v) = \sum_{a\le s} g_a(y)p(a|v)$ where $g_a(y)$ is some distribution over the feature values. Most of the time we use gaussian.

Pairwise Clustering

Object relations encoded by proximity or similarity data.
Metric relational: Data corresponds to euclidian distances.
Non-Metric relations: Violate metric properties e.g. Triangle inequality, Symmetrie. **Clustering principle:** Group objects with high similarity together and split those into different clusters that have no similarity.

Dissimilarity to Similarity: $S_{ij} = \frac{exp(-D_{ij})}{C}$ C constant

Graphbased clustering

Given $o_i, o_j \in \mathcal{O}$, and weights of relations $\mathcal{D} = \{S_{ij}\}$ on edge i,j. A cluster α is defined as $\mathcal{G}_\alpha = \{o \in \mathcal{O} : c(o) = \alpha\}$.
Intercluster - edges:
$\mathcal{E}_{\alpha\beta} = \{(i,j) \in \mathcal{E} : o_i \in \mathcal{G} : o_i \in \mathcal{G}_\alpha \wedge o_j \in \mathcal{G}_\beta\}$

Shifted correlation clustering:

Counts only similarities relative to a threshold value u.
$|X| \pm X = max\{0, \pm X\}$

$\mathcal{R}^{cc}(c,\mathcal{D}) = -\frac{1}{2}\sum_{\nu\le k}\sum_{(i,j)\in\mathcal{E}_{\nu\nu}}(|S_{ij}+u| + S_{ij} + u) + \frac{1}{2}\sum_{\nu\le k}\sum_{\mu\neq\nu}\sum_{(i,j)\in\mathcal{E}_{\nu\mu}}(|S_{ij}+u| - S_{ij} + u)$.
Symmetrization: $\mathcal{R}(c; \frac{1}{2}(D_{ij}+D_{ji})) = \mathcal{R}(c;\mathcal{D})$**.** Is used to transform data without changing the solution. $X^c = QXQ^T$ with proj. matrix $Q = I_n - \frac{1}{n}e_n e_n^T$.
$X^c_{ij} = X_{ij} - \frac{1}{n}\sum_{k=1}^n X_{ik} - \frac{1}{n}\sum_{k=1}^n X_{kj} + \frac{1}{n^2}\sum_{k,l=1}^n X_{kl}$ where X either D or S.
Decomposition of dissimilarities: $D_{ij} = S_{ii} + S_{jj} - 2S_{ij}$.
$S^c_{ij} = -\frac{1}{2}\Big[(D_{ij} - S_{ii} - S_{jj}) - \frac{1}{n}\sum_{k=1}^n(D_{ik} - S_{ii} - S_{kk}) - \frac{1}{n}\sum_{k=1}^n(D_{kj} - S_{kk} - S_{jj}) + \frac{1}{n^2}\sum_{k,l=1}^n(D_{kl} - S_{kk} - S_{ll})\Big] = -\frac{1}{2}\Big[D_{ij} - \frac{1}{n}\sum_{k=1}^n D_{ik} - \frac{1}{n}\sum_{k=1}^n D_{kj} + \frac{1}{n^2}\sum_{k,l=1}^n D_{kl}\Big] = -\frac{1}{2}D^c_{ij}$. (smallest eigv. of centralized similaritiy matrix is half largest eigv of centralized dissimilarity matrix.)
Constant shift embedding: Define $\hat{D} = D + \lambda_0(1-I)$ as the shift with smallest eigv. of centralized matrix $S^c_{ij} = -\frac{1}{2}D^c_{ij}$ then: 1. $\hat{D}_{ij}$ are squared Euclidian distances between vectors $\{\phi_i\}_{i=1}^n \in \mathbb{R}^{n-1}$. 2. optimal assignments $P(c(i) = v|\mathcal{D})$ for k-means based on $\{\phi_i\}_{i=1}^n$ are identical to those of pariwise problem. 3. $\{\phi_i\}_{i=1}^n$ are explicitly found by eigenvalue decomposition. 4. optimal approximative vectors (in lsq) projecting on leading eigenvectors. (kernel PCA)
Algo: Add algo? 1. $D_{ij} = S_{ii} + S_{jj} - 2S_{ij}$

Mean-field approximation

Approximate gibbs distribution since normalization constant is really hard to calculate.
Model:$Q(c,\theta^0) = \prod_{i\le n} q_i(c(i)), q_i(v) \in [0,1]$. (No statistical dependencies between $c(i), c(j)$.
Mean field $h_{i,v}$**:** parametrize $q_i(c(i)) = \frac{exp(-\beta h_{i,c(i)})}{\sum_{v\le k} exp(-\beta h_{i,v})}$
Minimize: $\hat{\theta}^0 = argmin_{\theta_0}\mathcal{D}^{KL}(Q(c,\theta^0)||P^{Gibbs}(c,\theta)) = argmin_{\theta^0}\sum_{c\in\mathcal{C})} Q(c,\theta^0)\frac{log(Q(c,\theta^0)}{P^{Gibbs}(c,\theta)}$
Iterative update: Since $\hat{\theta}^0$ is function of expexted assignments $\mathbb{E}[c(i)]$ we use EM-like update: E-step: Estimate $\mathbb{E}[c(i)]$ for fixed $\hat{\theta}^0$. Minimize KL-divergence w.r.t. $\hat{\theta}^0$ fox fixed $\mathbb{E}[c(i)]$.
Mean Field upper bound:
$0 \le \sum_{c\in\mathcal{C}} Q(c,\theta^0) log(\frac{Q(c,\theta^0)}{exp(-\beta(\mathcal{R}(c)-\mathcal{F})}) = \sum_{c\in\mathcal{C}} Q(c,\theta^0)\Big[\sum_{i\le n} log(q_i(c(i)) + \beta(\mathcal{R}(c)-\mathcal{F})\Big] = \sum_{i\le n}\sum_{c\in\mathcal{C}} Q(c,\theta^0) log(q_i(c(i)) + \beta(\mathbb{E}_Q[\mathcal{R}] - \mathcal{F}) =^* \sum_{i\le n}\sum_v q_i(v) log(q_i(v)) + \beta\mathbb{E}_Q[\mathcal{R}] - \beta\mathcal{F}$. summation of c in C redues to c(i) in 1,..,k due to normailzation. summations of c(j) j!=i sum up to 1.

Surrogate optimization target: Since we have to minimizing the free energy we minimize the bound under constraint $\sum_{v\leq k} q_i(v) = 1, \forall i$.

$$\mathcal{F} \leq \frac{1}{\beta}\sum_{i\leq n}\sum_v q_i(v) log(q_i(v)) + \mathbb{E}_Q[\mathcal{R}] = \mathcal{B}(\{q_i(v)\})$$

Extermality condition (approx by factorial distribution): minimize $\mathcal{B}(\{q_i(v)\})$ that yields the stationary condition w.r.t. $\{q_i(v)\}$:

$$0 = \frac{\partial}{\partial q_u(\alpha)}\mathcal{B}(\{q_i(v)\}) + \sum_{i\leq n}\lambda_i(\sum_{v\leq k} q_i(v) - 1) =$$

$$\frac{\partial}{\partial q_u(\alpha)}\sum_{c\in\mathcal{C}}\prod_{i\leq n} q_i(c(i))\mathcal{R}(c) + \frac{1}{\beta}(log q_u(\alpha) + 1) + \lambda_u$$

$$= \underbrace{\sum_{c\in C}\prod_{i\leq n: i\neq u} q_i(c(i))\mathbb{I}_{\{c(u)=\alpha\}}\mathcal{R}(c)}_{mean\ fields\ h_{u,\alpha}} + \frac{1}{\beta}(log q_u(\alpha) + 1) + \lambda_u\ \lambda_u$$

determined by normalization condition $\sum_{v\leq k} q_i(v) = 1$ i.e. $exp(\beta\lambda_u + 1) = \sum_{v\leq k} exp(-\beta h_{u,v})$

minimal cost to assign object u to cluster α:

$$\begin{aligned}
h_{u,\alpha} &= \frac{\partial}{\partial q_u(\alpha)}\sum_{c\in\mathcal{C}}\prod_{i\leq n} q_i(c(i))\mathcal{R}(c) \\
&= \frac{\partial}{\partial q_u(\alpha)}\sum_{c\in\mathcal{C}\setminus\{c(u)\}}\prod_{\substack{i\leq n:\\ i\neq u}} q_i(c(i))\sum_{c(u)\in\{1,\dots,k\}} q_u(c(u))\,\mathcal{R}(c) \\
&= \sum_{c\in\mathcal{C}\setminus\{c(u)\}}\prod_{\substack{i\leq n:\\ i\neq u}} q_i(c(i))\frac{\partial}{\partial q_u(\alpha)}\sum_{\nu\leq k} q_u(\nu)\,\mathbb{I}_{\{c(u)=\nu\}}\,\mathcal{R}(c) \\
&= \sum_{c\in\mathcal{C}}\prod_{\substack{i\leq n:\\ i\neq u}} q_i(c(i))\mathbb{I}_{\{c(u)=\alpha\}}\,\mathcal{R}(c) \\
&=: \mathbb{E}_{\mathbf{Q}_{u\to\alpha}}[\mathcal{R}(c)]
\end{aligned}$$

where $E_{Q_u\to\alpha}$ denotes an expectation over all configuration under the constraint that object u is assigned to cluster α.

Meanfield Equations: $q_u(\alpha) = \frac{exp(-\beta h_{u,\alpha})}{\sum_{v\leq k} exp(-\beta h_{u,v})}$, $h_{u,\alpha} = E_{Q_{u\to\alpha}}[\mathcal{R}(c)]$. **Calculation of meanfields:** Decompose $R(c)$ in contributions which depend on object u and the costs of all the other objects. The u dependent part influences $q_u(\alpha)$ the rest is an irrelevant constant.

EM Algo for meanfield:

Single coordinate descent

1: initialize $\forall i, q_i(1),\dots,q_i(k)$ randomly s.t. $\sum_\nu q_i(\nu) = 1$;
2: **repeat**
3: draw $u \in \{1,\dots,n\}$ randomly; // site selection
4: calculate $h_{u,\alpha} = \mathbb{E}_{\mathbf{Q}_{u\to\alpha}}[\mathcal{R}(c)]$ for given $q_u(1),\dots,q_u(k)$;
5: estimate $\forall\alpha,\ q_u(\alpha) = \frac{\exp(-\beta h_{u,\alpha})}{\sum_{v\leq k}\exp(-\beta h_{u,\nu})}$ given $h_{u,1},\dots,h_{u,k}$;
6: $t \leftarrow t+1$;
7: **until** $\forall\alpha$ probabilities $q_u(\alpha)$ and meanfields $h_{u,\alpha}$ are converged
8: stop

MeanField practical part: Problem: We want to denoise an image.
Idea: Which denoised image produced the noisy image best. So basically we want to find is $argmax_{p(x)} p(x,y) = argmax_{p(x)} p(x)p(y|x)$: Which perfect image x explains the noisy image y best.
Prior $p(x)$**:** We use the ising model so we get the prior already for free:
$p(x) = \frac{1}{Z}exp(-E(x)) = \frac{1}{Z}exp(-E(\sum\sum_{j\in NN(i)} x_i w_{i,j} x_j))$ and just take the logarithm and Z is left as a constant. Favors big patches of constant values.
likelihood: The likelihood $p(y|x) = \prod_i p(y_i|x_i) = \sum_i exp(-L_i(x_i))$ for some loss function L_i. We will use some gaussian loss function.
Posterior: $p(x|y) = \frac{p(y|x)p(x)}{p(x,y)} = \frac{1}{Z}exp(\sum_i L_i(x_i) - E(x))$ but this is hard to optimize.
Solution to posterior problem: Assume posterior can be factorized: $q(x) = \prod_i q_i(x_i;\mu_i)$ where we baiscally replaced the dependance on the neighbors by some "summary" μ_i of the neighbors that we can calculate independantly.
new prior: As a result we get a new prior that we calculate via: $log\hat{p}(x_i) = x_i\sum_{j\in NN(i)} W_{i,j}\mu_i + Li(x)$. Note that in a practical implementation we have a constant $Wi,j = J$ for all i,j. And a result of that we have basically a convolution of a 3x3 kernel with i,i = 0.
mf update: Since we encapsulated the whole thing into μ we have to update this term , too. So what we calculate is $\mu = E[x_i] = 1P(x_i = 1) + (-1)P(xi = -1) =$ (see murphy (21.46)) $= tanh(\sum j \in NN(i) W_{i,j}\mu_j + 0.5 * (L_i(+1) - L_i(-1))$

Model selection for Clustering

Minimum description length: $-\underbrace{logp(X|\theta_k)}_{data} - \underbrace{logp(\theta_k)}_{model}$

with one possible approx:

$$\hat{k} \in argmin_{1\leq k\leq K_{max}}\Big(\underbrace{-log(p(X|\theta_k))}_{neg.loglikelihood} + \underbrace{\frac{1}{2}k' logn}_{complexitypenalty}\Big)$$

where k' is the number of indp. parameters in the model encoded by θ_k
Gap statistics: $gap_n(k) := \mathbb{E}^*_n[log(W_k)] - log(W_k)$ where $W_k := \sum_{1\leq v\leq k}\frac{1}{2n_v}\sum_(i,j)\mathcal{E}_{vv}D_{ij}$ with $D_{ij} = ||x_i - x_j||^2$, $\hat{k} := min\{k|gap_n(k) \geq gap_n(k+1) - \sigma_{k+1}\}$. In practice: Apprixmate $\mathbb{E}^*_n[log(W_k)]$ by bootstrapping. **summary:** Gap works for spherically distributed data. - it is not model-free as it contains a structural bias. - for k-means like criteria it is a fast heuristic. **stability:** Solutions on two data sets from the same source should be similar.

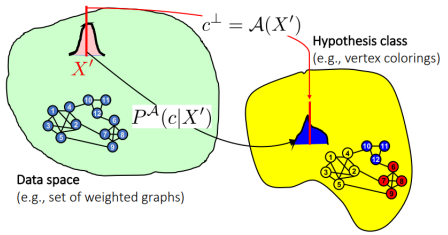
Lecture 8 - Model Validation by Information Theory

Distibution of algorithm output (minimizers): Replace "solution" $P(c^\perp) = \int_{\mathcal{X}} P(X)\delta(c^\perp - \mathcal{A}(X))dX$ where $P(c^\perp)$ denotes the distribution of minimzers given the stochastic data source $P(X)$. Data space: e.g. weighted graphs, Hypothesis class: e.g. vertex coloring

Sample algorithms from posteriori:

$$P(c) = \mathbb{E}_X[P^{\mathcal{A}}(c|X)] = \int_{\mathcal{X}} P^{\mathcal{A}}(c|X)P(X)dX \quad \text{with}$$

$$P^{\mathcal{A}}(c|X) = \frac{1}{Z}\exp(-\beta R(c,X)), \quad Z = \sum_{c'\in\mathcal{C}}\exp(-\beta R(c',X))$$



Construct algo posteriori:

Weighting of solutions: Given an instance X, we define a set of weights W

$$W : \mathcal{C}\times\mathcal{X}\times\mathbb{R}_+ \to [0,1]$$
$$(c,X,\theta) \mapsto W_\theta(c,X)$$

Choose the weights such that $\forall X\in\mathcal{X}, \forall\theta\in\mathbb{R}_+, \forall c',c''\in\mathcal{C}$,

$$R(c',X) \leq R(c'',X) \iff W_\theta(c',X) \geq W_\theta(c'',X)$$

Posterior distribution: $P(c|X) = \frac{W_\theta(c,X)}{\sum_{c'\in\mathcal{C}} W_\theta(c',X)}$

Boltzmann weights: $W_\beta(c,X) = \exp(-\beta R(c,X))$
Fermi weights: $W_{\beta,\gamma}(c,X) = (1+\exp(\beta(R(c,X)-\gamma)))^{-1}$
Approximation set: $W_\gamma(c,X) = \begin{cases}1 & \text{if } R(c,X) \leq R(c^\perp,X)+\gamma \\ 0 & \text{otherwise}\end{cases}$

Markov Chains

Let $A_ij = P(X_t = j|X_{t-1} = i)$ be **one step** transition matrix. $\pi_t(j) = P(X_t = j)$ being in state j at time t. π_0 initial distribution over states. Then $\pi_j(1) = \sum_i P(X_t = j|X_{t-1} = i) * \pi_0(i) = \sum_i A_{ij} * \pi_0(i) \implies \pi_1 = A\pi_0$ when iteratiting this equations and at some point $\pi = A\pi$ then π stationary distribution. Limiting distribution: $\pi_j = lim_{n\to\infty} A^n_{ij}$ $P(X_t = j) = \sum_i P(X_0 = i)A_{ij}(t) \to \pi_j ast \to \infty$
irreducable: There is a way from every node to every other node: $P(X_n|X_i) > 0$ for some n and every i.
Periodicity: $d(i) = gcd(t : A_{ii}(t) > 0)$ then check all $d(1), d(2),$.. and if all $d(i) = 1$ chain is aperiodic. Note that $d(1) = gcd\{3,5,7,9\} = 1$ And if all nodes have self loop anyway aperiodic.
Limitting distribution theorem: Every irreducible, aperiodic finite markov chain has a limiting distribution which is equal to π its stationary distribution.
Detailed balance equation: $p(x)p(x\to y) = p(y)p(y\to x)$ where $p(x\to y) = p(y|x)$. Detailed balance equation implies stationaryity: $\sum_x p(x)p(x\to y) = p(y)$

Exam properties on one line

Irreducibility: $\forall i,j \exists n s.t. P(X_n = j|X_0 = i) > 0$

Aperiocity: $d(i) = gcn\{t : P(X_t = i|X_0 = i) > 0\} = 1, \forall i$
Detailed balance:
$P(X_k = i)P(X_l = j|X_k = i) = P(X_k = j)P(X_l = i|X_k = j)$
Stationarity (Det. bal. implies stat.)
$P(X_k = j) = \sum_i P(X_q = i)P(X_k = j|X_q = i)$

EM Algorithm

Goal: Find max. likelihood solutions for models having latent variables.
Definitions: all observed data: $X \in \mathbb{R}^{n\times d}$ where x_i^T is the ith row. All latent variables by Z with a corresponding row z_i^T. The set of all parameters is θ.
(incomplete) Log-likelihood function:
$log(p(X|\theta) = ln(\sum_Z p(X, Z|\theta))$ **Complete data set:** if we assume that we know X and Z. Then complete log-likelihood: $log(p(X, Z|\theta))$ **Incomplete data set:** actual observed data X.
Algo:

1. Choose initial value for paramter θ^{old}
2. E-step evaluate $p(Z|X, \theta^{old})$
3. M step Evaluate $\theta^{new} = argmax_\theta Q(\theta, \theta^{old})$ where $Q(\theta, \theta^{old}) = \sum_Z p(Z|X, \theta^{old}) lnp(X, Z|\theta)$
4. check convergence, else $\theta^{old} = \theta$, goto 2

Constant Shift embedding

Given D pairwise clustering cost function is

$$H^{pc}(M) = \frac{1}{2}\sum \nu = 1^k \frac{\sum i, j = 1^n Mi\nu Mj\nu Dij}{\sum i = 1^n Mi\nu}$$

where $Mi\nu$ is binary assignment i belongs to ν. to transform the problem into one of k-means clustering on n-dimensional embeddings $\{xi\}i = 1^n$. The steps of the embedding algorithm are: Symmetrize the matrix D, and decompose it by introducing a similarity matrix S such that $Dij = Sii + Sjj - 2Sij$. For any such matrix, its centralization $S^c = QSQ$ where $Q = I_n - \frac{1}{n}e_n e_n^\top$ also induces a decomposition of D and satisfies $S^c = -\frac{1}{2}D^c$. Compute the smallest eigenvalue $\lambda_n(S^c)$ of S^c, and if negative shift the diagonal entries of S^c to obtain a positive semidefinite matrix $\tilde{S} = S^c - \lambda_n(S^c)I_n$ which can be expressed as a Gram matrix of vectors $\{xi\}i = 1^n$. Compute $\tilde{D}_{ij} = \tilde{S}_{ii} + \tilde{S}_{jj} - 2\tilde{S}_{ij}$, which is the dissimilarity matrix obtained by shifting the off-diagonal entries of D by $\lambda_n(S^c)$ and hence induces the same cost function, but also satisfies $Dij = \|x_i - x_j\|^2$. Use the resulting matrix of squared distances to apply k-means clustering to the vectors $\{x_i\}_{i=1}^n$
The embeddings $\{x_i\}_{i=1}^n$ can be obtained by an eigendecomposition of $S^c = XX^\top = U\Lambda U^\top$, from which $X = (xi, j)ij$ may be calculated as $X = U_p\sqrt{\Lambda_p}$, where p is the number of nonzero eigenvalues of S^c, and Λ_p and U_p store the corresponding eigenvalues and eigenvectors. This also allows us to denoise the embedding vectors using PCA, by only considering the first $p^* < p$ eigenvalues of S^c.

Pairwise clustering

Given a dataset of size N and an $N \times N$ dissimilarity matrix $\mathbf{D} = (\mathcal{D}_{ik})$, the problem of pairwise clustering tries to cluster the data into K clusters, represented by an $N \times K$ assignment matrix $\mathbf{M} = (M_{i\nu}) \in \{0, 1\}^{N\times K}$ that minimizes the cost function $\mathcal{H}^{pc}(\mathbf{M}) = \frac{1}{2}\sum_{i,k=1}^N \frac{\mathcal{D}_{ik}}{N}\left(\sum_{\nu=1}^K \frac{M_{i\nu}M_{k\nu}}{p_\nu} - 1\right)$ where $p_\nu = \frac{1}{N}\sum_{i=1}^N M_{i\nu}$ is the weight of cluster ν. Algorithm II approximates the Gibbs distribution given by $\mathcal{H}^{pc}$ with the Gibbs distribution given by the factorial cost function $\mathcal{H}^0(\mathbf{M}, \mathcal{E}) = \sum_{i=1}^N \sum_{\nu=1}^K M_{i\nu}\mathcal{E}_{i\nu}$, parameterized by external mean fields $\mathcal{E} = (\mathcal{E}_{i\nu})$. Minimizing the Kullback-Leibler divergence $D^{KL}(\mathbf{P}^{Gb}(\mathcal{H}^0)\|\mathbf{P}^{Gb}(\mathcal{H}^{pc})$, where $\mathbf{P}^{Gb}(\mathcal{H})$ is the Gibbs distribution for the cost function $\mathcal{H}(\mathbf{M})$ gives the optimal $\mathcal{E}$ as

$$\mathcal{E}^*_{i\nu} = \left\langle \frac{1}{1 + \sum_{j\neq i} M_{j\nu}} \left[\frac{1}{2}\mathcal{D}_{ii} + \sum_{k\neq i} M_{k\nu}\left(\mathcal{D}_{ik} - \frac{1}{2}\frac{\sum_{j\neq i} M_{j\nu}\mathcal{D}_{jk}}{\sum_{j\neq i} M_{j\nu}}\right)\right]\right\rangle$$

where $\langle\cdot\rangle$ corresponds to taking expectations with respect to $\mathbf{P}^{Gb}(\mathcal{H}^0(\mathbf{M}))$. In the limit of large N, this optimum can be approximated as

$$\mathcal{E}^*_{i\nu} = \frac{1}{1 + \sum_{j\neq i}\langle M_{j\nu}\rangle}\left[\frac{1}{2}\mathcal{D}_{ii} + \sum_{k\neq i}\langle M_{k\nu}\rangle\left(\mathcal{D}_{ik} - \frac{1}{2}\frac{\sum_{j\neq i}\langle M_{j\nu}\rangle\mathcal{D}_{jk}}{\sum_{j\neq i}\langle M_{j\nu}\rangle}\right)\right]$$

The expectations $\langle M_{j\alpha}\rangle$ under the Gibbs distribution $\mathbf{P}^{Gb}(\mathcal{H}^0(\mathbf{M}))$ with temperature T are then given by

$$\langle M_{j\alpha}\rangle = \frac{\exp(-\mathcal{E}^*_{i\alpha}/T)}{\sum_{\nu=1}^K \exp(-\mathcal{E}^*_{i\nu}/T)}$$

Algorithm II then uses deterministic annealing to approximate the Gibbs distribution for varying temperatures, by starting at high temperature, iteratively applying the two above equations for $\langle M_{j\alpha}\rangle$ and $\mathcal{E}^*_{i\nu}$ in an EM scheme for fixed T, and decreasing T exponentially until the final temperature is reached.
Algorithm III further constrains the variational distribution to be in the form of a Gibbs distribution for the k-means cost function

$$\mathcal{H}^{cc}(\mathbf{M}) = \sum_{i=1}^N\sum_{\nu=1}^K M_{i\nu}\|\mathbf{x}_i - \mathbf{y}_\nu\|^2,$$

where $(\mathbf{x}_i)_{i=1}^N$ are embeddings of the data to be determined, and $(\mathbf{y}_\nu)_{\nu=1}^K$ are centroids calculated from $(\mathbf{x}_i)_{i=1}^N$; in this case the mean fields are constrained to take the form $\mathcal{E}_{i\nu} = \|\mathbf{x}_i - \mathbf{y}_\nu\|^2$. Minimizing the KL-divergence $D^{KL}(\mathbf{P}^{Gb}(\mathcal{H}^{cc})\|\mathbf{P}^{Gb}(\mathcal{H}^{pc}))$ with respect to $(\mathbf{x}_i)_{i=1}^N$ and $(\mathbf{y}_\nu)_{\nu=1}^K$ now gives

$$\mathbf{K}_i\mathbf{x}_i \approx \frac{1}{2}\sum_{\nu=1}^K \langle M_{i\nu}\rangle\left(\|\mathbf{y}_\nu\|^2 - \mathcal{E}^*_{i\nu}\right)\left(\mathbf{y}_\nu - \langle\mathbf{y}\rangle_i\right)$$

where $\langle\mathbf{y}\rangle_i = \sum_{\nu=1}^K \langle M_{i\nu}\rangle\mathbf{y}_\nu$ is the mean of $\mathbf{y}$ under the conditional distribution $p(\nu|i) = \langle M_{i\nu}\rangle$ and $\mathbf{K}_i = \left\langle\mathbf{y}\mathbf{y}^\top\right\rangle_i - \langle\mathbf{y}\rangle_i\langle\mathbf{y}\rangle_i^\top$ the corresponding covariance matrix, and $\mathcal{E}^*_{i\nu}$ is as calculated in Algorithm II. Algorithm III again uses deterministic annealing, calculating $\langle M_{i\nu}\rangle^{(t+1)}$ in the E-step from the mean fields $\mathcal{E}^{(t)}_{i\nu} = \|\mathbf{x}_i^{(t)} - \mathbf{y}_\nu^{(t)}\|^2$, and in the M-step updating $\mathbf{x}_i^{(t+1)}$ and $\mathbf{y}_\nu^{(t+1)}$ iteratively from fixed $\langle M_{i\nu}\rangle^{(t+1)}$ until the variables converge.

Model Validation

In order to compare the stability of cost functions $R(c, \mathbf{X}) \in \mathcal{R}$ under noise in the data $\mathbf{X} \sim \mathbb{P}(\mathbf{X})$, approximation set coding considers a communication scenario including a sender, a problem generator and a receiver. Each entity is initially given access to a sample $\mathbf{X^{(1)}} \sim \mathbb{P}(\mathbf{X})$ and an optimal solution $c^\perp(\mathbf{X^{(1)}}) \in \arg\max_{c\in\mathcal{C}} R(c, \mathbf{X^{(1)}})$. The sender then chooses a transformation $\tau_s \in \mathbb{T}$, $|\mathbb{T}| = 2^{n\rho}$ which acts on the solution space (by permuting elements of the dataset in the case of clustering) and sends it to the problem generator, which generates another sample $\mathbf{X^{(2)}} \sim \mathbb{P}(\mathbf{X})$ and sends $\tau_s \circ \mathbf{X}^{(2)}$ to the receiver. The receiver then estimates τ_s by maximizing the posterior agreement between $\mathbb{P}(c|\beta, \mathbf{X}^{(1)}) \propto \exp(-\beta R(c, \mathbf{X}^{(1)}))$ and $\mathbb{P}(c|\beta, \tau \circ \mathbf{X}^{(2)}) \propto \exp(-\beta R(c, \tau \circ \mathbf{X}^{(2)}))$, returning $\hat{\tau} \in \arg\max_{\tau\sum\mathbb{T}}\sum_{c\in\mathcal{C}}\exp(-\beta(R(c, \mathbf{X}^{(1)} + R(c, \tau \circ \mathbf{X}^{(2)}))$. Here β represents the degree to which the data is trusted; the larger the noise in the data, the lower β has to be chosen to prevent decoding errors. The decoding error $\mathbb{P}(\hat{\tau} \neq \tau_s|\tau_s)$ can be shown to vanish asymptotically if the rate of transmission ρ falls below the mutual information

$$\mathcal{I}_\beta(\tau_s, \hat{\tau}) = \frac{1}{n}\log\frac{|\{\tau_s\}|\mathcal{Z}_{12}}{\mathcal{Z}_1\mathcal{Z}_2},$$

where $\mathcal{Z}_1 = \sum_{c\in\mathcal{C}}\exp(-\beta R(c, \mathbf{X^{(1)}}))$,
$\mathcal{Z}_2 = \sum_{c\in\mathcal{C}}\exp(-\beta R(c, \mathbf{X^{(2)}}))$,
$\mathcal{Z}_{12} = \sum_{c\in\mathcal{C}}\exp(-\beta(R(c, \mathbf{X^{(1)}}) + R(c, \mathbf{X^{(2)}})))$ and $|\{\tau_s\}|$ is the number of possible realizations of $c^\perp(\tau_s \circ \mathbf{X}^{(2)})$.
The approximation capacity is then defined as the maximum of $\mathcal{I}_\beta$ over all inverse temperatures β. Then to apply model selection on $\mathcal{R}$, we split a given dataset $\mathbf{X}$ randomly into two datasets $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, compute the mutual information $\mathcal{I}_\beta$ for each $R \in \mathcal{R}$, maximize it with respect to β, and choose the cost function with the largest approximation capacity.
For k-means clustering, we have $R(c, \mathbf{X}; \mathbf{Y}) = \sum_{i=1}^n \epsilon_{i,c(i)}$ where $\epsilon_{ik} = \epsilon_{ik}(\mathbf{X}, \mathbf{Y}) = \|x_i - y_k\|^2$ and $\mathbf{Y}$ are the cluster centroids inferred by maximizing the entropy of $\mathbb{P}(c|\beta, \mathbf{X})$. For $\epsilon_{ik}^{(i)} = \epsilon_{ik}(\mathbf{X}^{(i)}, \mathbf{Y})$ the mutual information is calculated as

$$\mathcal{I}_\beta = \frac{1}{n}\log|\{\tau_s\}| + \frac{1}{n}\sum_{i=1}^n \log\frac{\sum_{k=1}^K e^{-\beta(\epsilon_{ik}^{(1)} + \epsilon_{ik}^{(2)})}}{\sum_{k=1}^K e^{-\beta\epsilon_{ik}^{(1)}}\sum_{k=1}^K e^{-\beta\epsilon_{ik}^{(2)}}}$$

where $|\{\tau_s\}|$ is the number of distinct clusterings on $\mathbf{X}^{(1)}$. Then to evaluate the approximation capacity of the k-means cost function, we use deterministic annealing to compute the optimal centroids and costs $R(c, \mathbf{X}^{(i)})$ at different temperatures $T = \beta^{-1}$, allowing for $2K$ possible clusters to enable overfitting, and choose as stopping temperature the one with the highest mutual information, balancing informativeness and robustness.