

CS5340

Uncertainty Modeling in AI

Lecture 6: Parameter Learning with Complete Data

Assoc. Prof. Lee Gim Hee

AY 2022/23

Semester 1

Course Schedule

Week	Date	Topic	Remarks
1	10 Aug	Introduction to probabilistic reasoning	Assignment 0: Python Numpy Tutorial (Ungraded)
2	17 Aug	Bayesian networks (Directed graphical models)	
3	24 Aug	Markov random Fields (Undirected graphical models)	
4	31 Aug	Variable elimination and belief propagation	Assignment 1: Belief propagation and maximal probability (15%)
5	07 Sep	Factor graph and the junction tree algorithm	
6	14 Sep	Parameter learning with complete data	Assignment 1: Due Assignment 2: Junction tree and parameter learning (15%)
-	21 Sep	Recess week	No lecture
7	28 Sep	Mixture models and the EM algorithm	Assignment 2: Due
8	05 Oct	Hidden Markov Models (HMM)	Assignment 3: Hidden Markov model (15%)
9	12 Oct	Monte Carlo inference (Sampling)	
*	15 Oct	Variational inference	Makeup Lecture (LT15) Time: 9.30am – 12.30pm (Saturday)
10	19 Oct	Variational Auto-Encoder and Mixture Density Networks	Assignment 3: Due Assignment 4: MCMC Sampling (15%)
11	26 Oct	No Lecture	I will be traveling
12	02 Nov	Graph-cut and alpha expansion	Assignment 4: Due
13	09 Nov	-	

Acknowledgements

- A lot of slides and content of this lecture are adopted from:
 1. Michael I. Jordan “An introduction to probabilistic graphical models”, 2002, Chapter 9
 2. Kevin Murphy, “Machine learning: a probabilistic approach”, Chapters 10.4, 19.5, and 19.6.3
 3. “Computer Vision: Models, Learning, and Inference”, Simon Prince.
 4. Daphne Koller and Nir Friedman, “Probabilistic graphical models”, Chapter 17
 5. David Barber, “Bayesian reasoning and machine learning”, Chapters 9.1, 9.2, 9.3, 9.4, 9.6

Learning Outcomes

- Students should be able to:
 1. Compute the unknown parameters of discrete/continuous **DGMs** using **MLE** and **MAP**.
 2. Compute the unknown parameters of **MRFs** using **stochastic maximum likelihood**, and **iterative proportional fitting**.
 3. Compute the unknown parameters of **CRFs** using **stochastic gradient descent**.

Motivation

- In lectures 4 and 5, we learned how to do **exact inference** given a DGM/UGM $p(x_1, \dots, x_M | \theta)$.
- We will now look into the details of the unanswered question on:

How to get the **unknown parameter** θ of a DGM/UGM $p(x_1, \dots, x_M | \theta)$ from **fully observed data**?

Unknown Parameters Learning

- **Given:** a set of N **independent and identically distributed** (i.i.d) complete observation of each random variable $X: \{x_{1,1}, \dots, x_{1,N}, \dots, x_{M,1}, \dots, x_{M,N}\}$.
- Two commonly used approaches to **learn the unknown parameters** θ :
 1. Maximum likelihood estimate (MLE)
 2. Maximum a posteriori (MAP)

Maximum Likelihood Estimate (MLE)

- As the name suggests, we find the unknown parameters θ that **maximize the likelihood** $p(x_1, \dots, x_M | \theta)$:

$$\begin{aligned}\hat{\theta} &= \operatorname{argmax}_{\theta} [p(x_1, \dots, x_M | \theta)] \\ &= \operatorname{argmax}_{\theta} [\prod_{i=1}^N p(x_{1,i}, \dots, x_{M,i} | \theta)] \quad (\text{i.i.d})\end{aligned}$$

Maximum a Posteriori (MAP)

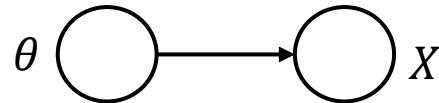
- As the name suggests, we find the unknown parameters θ that **maximize the a posteriori** probability $p(\theta|x_1, \dots x_M)$:

$$\begin{aligned}\hat{\theta} &= \operatorname{argmax}_{\theta} [p(\theta|x_1, \dots x_M)] \\ &= \operatorname{argmax}_{\theta} \left[\frac{p(x_1, \dots x_M|\theta)p(\theta)}{p(x_1, \dots x_M)} \right] \quad (\text{Bayes' rule}) \\ &= \operatorname{argmax}_{\theta} \left[\frac{\prod_{i=1}^N p(x_{1,i}, \dots, x_{M,i} | \theta) p(\theta)}{p(x_1, \dots x_M)} \right] \quad (\text{i.i.d}) \\ &= \operatorname{argmax}_{\theta} [\prod_{i=1}^N p(x_{1,i}, \dots, x_{M,i} | \theta) p(\theta)]\end{aligned}$$

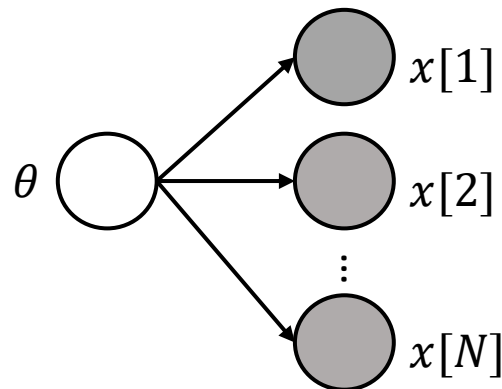
$(p(x_1, \dots x_M))$ is removed
since it is independent of θ

Special Case: Single Random Variable DGM

- We first look at learning the unknown parameter θ of a **single random variable** DGM $p(x|\theta)$.



- For N i.i.d. observations $X : \{x[1], \dots, x[N]\}$, the DGM becomes an augmentation of **N disconnected replicates of X** .



Special Case (Continuous DGM): Univariate Normal Distribution

Problem:

Fit a univariate normal distribution model to a set of scalar data $X : \{x[1], \dots, x[N]\}$.

Recall that the univariate normal distribution is given by:

$$p(x \mid \theta) = \text{Norm}_x[\mu, \sigma^2] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(x - \mu)^2}{2\sigma^2}$$

Our goal is to **find the two unknown parameters $\theta = (\mu, \sigma^2)$** .

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 1: **Maximum Likelihood Estimation (MLE)**

$$\begin{aligned}\hat{\theta} &= \operatorname{argmax}_{\theta} [p(x|\theta)] \\ &= \operatorname{argmax}_{\theta} \left[\prod_{i=1}^N p(x[i]|\theta) \right] \quad (\text{i.i.d})\end{aligned}$$

Likelihood given by pdf

$$p(x|\mu, \sigma^2) = \prod_{i=1}^N \text{Norm}_{x[i]} [\mu, \sigma^2],$$

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 1: **Maximum Likelihood Estimation (MLE)**

Algebraically:

$$\hat{\mu}, \hat{\sigma}^2 = \operatorname{argmax}_{\mu, \sigma^2} [p(x|\mu, \sigma^2)]$$

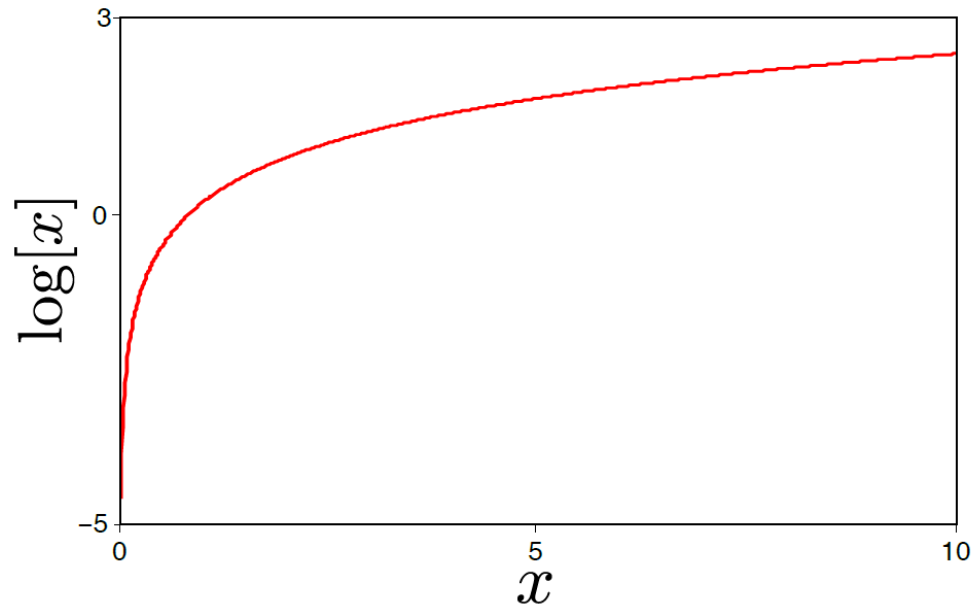
where

$$p(x|\mu, \sigma^2) = \prod_{i=1}^N \operatorname{Norm}_{x[i]} [\mu, \sigma^2],$$

or alternatively, we can **maximize the logarithm**:

$$\begin{aligned} \hat{\mu}, \hat{\sigma}^2 &= \operatorname{argmax}_{\mu, \sigma^2} \sum_{i=1}^N \log [\operatorname{Norm}_{x[i]} [\mu, \sigma^2]] \\ &= \operatorname{argmax}_{\mu, \sigma^2} \left[-0.5N \log [2\pi] - 0.5N \log \sigma^2 - 0.5 \sum_{i=1}^N \frac{(x[i] - \mu)^2}{\sigma^2} \right] \end{aligned}$$

Why the Logarithm?



- The logarithm is a **monotonic** transformation.
- Hence, the position of the **peak stays in the same place**.
- But the log likelihood is **easier to work with**.

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 1: **Maximum Likelihood Estimation (MLE)**

$$\begin{aligned}\hat{\mu}, \hat{\sigma}^2 &= \operatorname{argmax}_{\mu, \sigma^2} \sum_{i=1}^N \log [\operatorname{Norm}_{x[i]}[\mu, \sigma^2]] \\ &= \operatorname{argmax}_{\mu, \sigma^2} \underbrace{\left[-0.5N \log [2\pi] - 0.5N \log \sigma^2 - 0.5 \sum_{i=1}^N \frac{(x[i] - \mu)^2}{\sigma^2} \right]}_L\end{aligned}$$

Maximization can be done in closed-form by taking **derivative w.r.t. the variable and equate to zero**:

$$\frac{\partial L}{\partial \mu} = \sum_{i=1}^N \frac{(x[i] - \mu)}{\sigma^2} = \frac{\sum_{i=1}^N x[i]}{\sigma^2} - \frac{N\mu}{\sigma^2} = 0, \quad \frac{\partial L}{\partial \sigma^2} = -\frac{N}{\sigma^2} + \sum_{i=1}^N \frac{(x[i] - \mu)^2}{\sigma^4} = 0$$

$$\Rightarrow \hat{\mu} = \frac{\sum_{i=1}^N x[i]}{N} = \bar{x}, \quad \Rightarrow \hat{\sigma}^2 = \frac{\sum_{i=1}^N (x[i] - \mu)^2}{N}$$

Least Squares

Maximum likelihood for the normal distribution...

$$\begin{aligned}\hat{\mu} &= \operatorname{argmax}_{\mu} \left[-0.5N \log [2\pi] - 0.5N \log \sigma^2 - 0.5 \sum_{i=1}^N \frac{(x[i] - \mu)^2}{\sigma^2} \right] \\ &= \operatorname{argmax}_{\mu} \left[- \sum_{i=1}^N (x[i] - \mu)^2 \right] \\ &= \operatorname{argmin}_{\mu} \left[\sum_{i=1}^N (x[i] - \mu)^2 \right]\end{aligned}$$

...gives 'least squares' fitting criterion.

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 2: **Maximum a Posteriori (MAP)**

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^N \underset{\substack{\uparrow \\ \text{Likelihood}}}{p(x[i] | \theta)} p(\theta) \right] \quad \underset{\substack{\nwarrow \\ \text{Prior}}}{p(\theta)}$$

Likelihood: univariate Normal distribution

$$p(x|\mu, \sigma^2) = \prod_{i=1}^N \text{Norm}_{x[i]} [\mu, \sigma^2],$$

Prior: conjugate prior – normal inverse gamma distribution

$$\begin{aligned} p(\mu, \sigma^2) &= \text{NormInvGam}_{\mu, \sigma^2} [\alpha, \beta, \gamma, \delta] \\ &= \frac{\sqrt{\gamma}}{\sigma \sqrt{2\pi}} \frac{\beta^\alpha}{\Gamma[\alpha]} \left(\frac{1}{\sigma^2} \right)^{\alpha+1} \exp \left[-\frac{2\beta + \gamma(\delta - \mu)^2}{2\sigma^2} \right] \end{aligned}$$

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 2: Maximum a Posteriori (MAP)

$$\begin{aligned}\hat{\mu}, \hat{\sigma}^2 &= \operatorname{argmax}_{\mu, \sigma^2} \left[\prod_{i=1}^N p(x[i] | \mu, \sigma^2) p(\mu, \sigma^2) \right] \\ &= \operatorname{argmax}_{\mu, \sigma^2} \left[\prod_{i=1}^N \operatorname{Norm}_{x[i]}[\mu, \sigma^2] \operatorname{NormInvGam}_{\mu, \sigma^2}[\alpha, \beta, \gamma, \delta] \right]\end{aligned}$$

Maximize the logarithm:

$$\hat{\mu}, \hat{\sigma}^2 = \operatorname{argmax}_{\mu, \sigma^2} \left[\sum_{i=1}^N \log [\operatorname{Norm}_{x[i]}[\mu, \sigma^2]] + \log [\operatorname{NormInvGam}_{\mu, \sigma^2}[\alpha, \beta, \gamma, \delta]] \right]$$

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 2: Maximum a Posteriori (MAP)

$$\hat{\mu}, \hat{\sigma}^2 = \underset{\mu, \sigma^2}{\operatorname{argmax}} \left[\underbrace{\sum_{i=1}^N \log [\operatorname{Norm}_{x[i]}[\mu, \sigma^2]]}_{L} + \log [\operatorname{NormInvGam}_{\mu, \sigma^2}[\alpha, \beta, \gamma, \delta]] \right]$$

Taking derivatives and setting to zero:

$$\frac{\partial L}{\partial \mu} = 0, \quad \frac{\partial L}{\partial \sigma^2} = 0$$

We get:

$$\begin{aligned} \hat{\mu} &= \frac{\sum_i x[i] + \gamma \delta}{N + \gamma}, & \hat{\sigma}^2 &= \frac{\sum_i (x[i] - \mu)^2 + 2\beta + \gamma(\delta - \mu)^2}{N + 3 + 2\alpha} \\ &= \frac{N\bar{x} + \gamma\delta}{N + \gamma} \end{aligned}$$

Comparing MLE and MAP

More data points \rightarrow MAP is closer to MLE
Fewer data points \rightarrow MAP is closer to MP

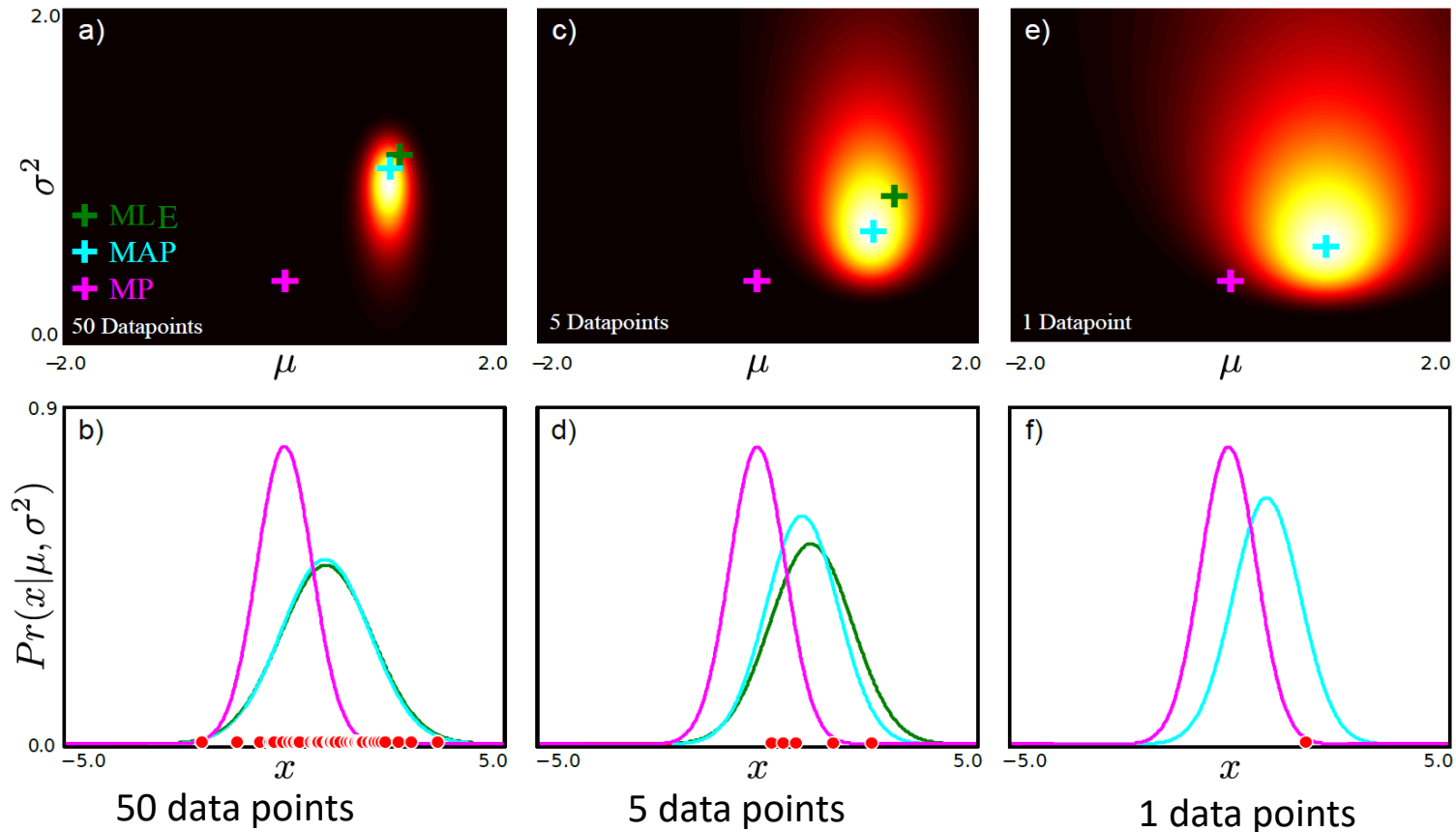


Image Source: "Computer Vision: Models, Learning, and Inference", Simon Prince

Special Case (Discrete DGM): Univariate Categorical Distribution

Problem:

Fit a categorical distribution model (**K categories**) to a set of data $\mathbf{X} : \{\mathbf{x}[1], \dots, \mathbf{x}[N]\}$, where $\mathbf{x}[i] = \mathbf{e}_k$ is a **vector with all zero elements except k^{th}** e.g. $[0,0,0,1,0,0]$, where $K=6$.

Recall that the categorical distribution is given by:

$$p(\mathbf{X} = \mathbf{e}_k | \theta) = \text{Cat}_x[\lambda] = \prod_{k=1}^K \lambda_k^{x_k} = \lambda_k$$

k^{th} element of \mathbf{e}_k

Our goal is to **find the K unknown parameters $\theta = \{\lambda_1, \dots, \lambda_K\}$** , where $\lambda_k \in [0,1]$ and $\sum_k \lambda_k = 1$.

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 1: **Maximum Likelihood Estimation (MLE)**

$$\begin{aligned}\hat{\theta} &= \operatorname{argmax}_{\theta} [p(\mathbf{x}|\theta)] \\ &= \operatorname{argmax}_{\theta} \left[\prod_{i=1}^N p(\mathbf{x}[i]|\theta) \right] \quad (\text{i.i.d})\end{aligned}$$

Likelihood given by pdf

$$p(\mathbf{x}|\lambda) = \prod_{i=1}^N \operatorname{Cat}_{\mathbf{x}[i]}[\lambda_{1\dots K}] = \prod_{i=1}^N \prod_{k=1}^K \lambda_k^{x_{ik}}$$

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 1: **Maximum Likelihood Estimation (MLE)**

$$\hat{\lambda}_{1...K} = \operatorname{argmax}_{\lambda_{1...K}} \prod_{i=1}^N p(\mathbf{x}[i] \mid \lambda_{1...K}), \quad s.t. \quad \sum_k \lambda_k = 1$$

$$= \operatorname{argmax}_{\lambda_{1...K}} \prod_{i=1}^N \operatorname{Cat}_{\mathbf{x}[i]}[\lambda_{1...K}], \quad s.t. \quad \sum_k \lambda_k = 1$$

$$= \operatorname{argmax}_{\lambda_{1...K}} \prod_{i=1}^N \prod_{k=1}^K \lambda_k^{x_{ik}}, \quad s.t. \quad \sum_k \lambda_k = 1$$

k^{th} element of \mathbf{x}_i , $x_{ik} \in \{0,1\}$

$$= \operatorname{argmax}_{\lambda_{1...K}} \prod_{k=1}^K \lambda_k^{N_k}, \quad s.t. \quad \sum_k \lambda_k = 1$$

$N_k = \#$ times we observed bin k

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 1: **Maximum Likelihood Estimation (MLE)**

Applying log probability and **Lagrange multiplier** ν on the constraint, we get the **auxiliary function** :

$$\mathcal{L} = \sum_{k=1}^K N_k \log[\lambda_k] + \nu \left(\sum_{k=1}^K \lambda_k - 1 \right)$$

Take derivative of \mathcal{L} w.r.t λ_k and ν , set to zero and solve for λ_k :

$$\hat{\lambda}_k = \frac{N_k}{\sum_{m=1}^K N_m}$$

Normalized counts of # times
we observed bin k

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 2: **Maximum a Posteriori (MAP)**

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\underbrace{\prod_{i=1}^N p(x[i]|\theta)}_{\text{Likelihood}} \underbrace{p(\theta)}_{\text{Prior}} \right]$$

Likelihood: categorical distribution

$$p(\mathbf{x}|\lambda) = \prod_{i=1}^N \operatorname{Cat}_{x[i]}[\lambda_{1...K}] = \prod_{i=1}^N \prod_{k=1}^K \lambda_k^{x_{ik}} = \prod_{k=1}^K \lambda_k^{N_k}$$

Prior: conjugate prior – Dirichlet distribution

$$\begin{aligned} p(\lambda_1, \dots, \lambda_K) &= \operatorname{Dir}_{\lambda_{1...K}}[\alpha_1, \dots, \alpha_K] \\ &= \frac{\Gamma[\sum_{k=1}^K \alpha_k]}{\prod_{k=1}^K \Gamma[\alpha_k]} \prod_{k=1}^K \lambda_k^{\alpha_k - 1}, \quad \text{s.t. } \lambda_k \in [0,1], \sum_k \lambda_k = 1 \end{aligned}$$

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 2: **Maximum a Posteriori (MAP)**

$$\hat{\lambda}_{1...K} = \operatorname{argmax}_{\lambda_{1...K}} \prod_{i=1}^N p(x[i]|\lambda_{1...K})p(\lambda_{1...K}), \quad s.t. \quad \sum_k \lambda_k = 1$$

$$= \operatorname{argmax}_{\lambda_{1...K}} \prod_{i=1}^N \operatorname{Cat}_{x[i]}[\lambda_{1...K}] \operatorname{Dir}_{\lambda_{1...K}}[\alpha_1, \dots, \alpha_K], \quad s.t. \quad \sum_k \lambda_k = 1$$

Independent of $\lambda \Rightarrow$ can be ignored

$$= \operatorname{argmax}_{\lambda_{1...K}} \frac{\Gamma[\sum_{k=1}^K \alpha_k]}{\prod_{k=1}^K \Gamma[\alpha_k]} \prod_{k=1}^K \lambda_k^{N_k} \prod_{k=1}^K \lambda_k^{\alpha_k - 1}, \quad s.t. \quad \sum_k \lambda_k = 1$$

$$= \operatorname{argmax}_{\lambda_{1...K}} \prod_{k=1}^K \lambda_k^{N_k + \alpha_k - 1}, \quad s.t. \quad \sum_k \lambda_k = 1$$

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 2: **Maximum a Posteriori (MAP)**

Applying log probability and **Lagrange multiplier** ν on the constraint, we get the **auxiliary function**:

$$\mathcal{L} = \sum_{k=1}^K (N_k + \alpha_k - 1) \log \lambda_k + \nu \left(\sum_{k=1}^K \lambda_k - 1 \right)$$

Take derivative of \mathcal{L} w.r.t λ_k and ν , set to zero and solve for λ_k :

Same result as MLE with a
uniform prior $\alpha_{1\dots K} = 1$

$$\hat{\lambda}_k = \frac{N_k + \alpha_k - 1}{\sum_{m=1}^K (N_m + \alpha_m - 1)}$$



$$\hat{\lambda}_k = \frac{N_k}{\sum_{m=1}^K N_m}$$

Parameter Learning: DGM

- Let $G = (U, E)$ be a **directed graph**, where U is the set of nodes and E the set of edges.
- X_u denotes the random variable associated with node $u \in U$, and x_u denotes a **realization** of X_u .
- To each node $u \in U$, we associate a **local conditional probability distribution** $p(x_u | x_{\pi_u}, \theta_u)$.

Parameter Learning: DGM

- π_u denotes the set of indices of the parents of u and where θ_u is a **parameter vector**.
- The **overall probability** associated with the graph G is a product of the local probabilities:

$$p(x_U|\theta) = \prod_{u \in U} p(x_u | x_{\pi_u}, \theta_u), \quad \theta = (\theta_1, \dots, \theta_{|U|})$$

Complete Observation

A complete observation is **an assignment of values to ALL of the random variables X_U in the model.**

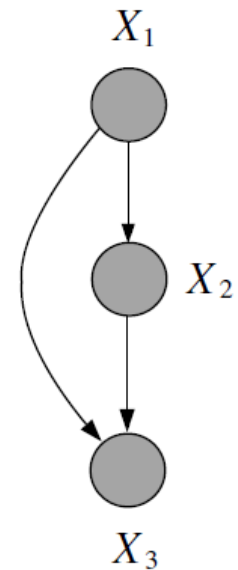


Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Complete Observation

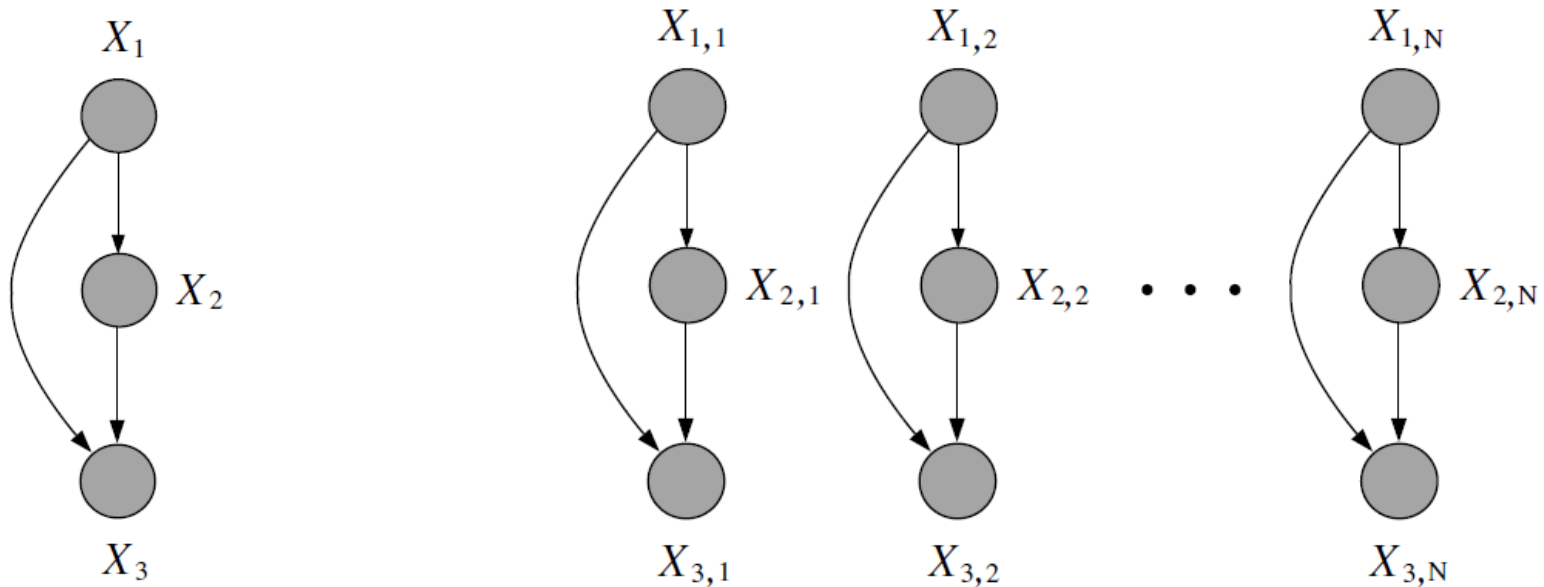
- For N i.i.d. observations, the graphical model simply becomes an augmentation of N disconnected replicates of G .
- We denote the augmented graphical model as:

$$G^{(N)} = (U^{(N)}, E^{(N)})$$

- Nodes $U^{(N)}$ are indexed using a pair of labels (u, n) , where $u \in U$ designates a node in G and $n \in \{1, \dots, N\}$ designates the replication number.

Complete Observation

Example:



A graphical model G ,
where $U = \{1,2,3\}$

$G^{(N)}$ obtained by making N replicates of G
 n^{th} complete observation is denoted $X_{U,n} = (x_{1,n}, x_{2,n}, x_{3,n})$

Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Complete Observation

- We can write the **entire set of observed data** in the completely observed setting as:

$$\mathcal{D} = (x_{U,1}, x_{U,2}, \dots, x_{U,N})$$

- The **probability model for $G^{(N)}$** is thus given by:

$$\begin{aligned} p(\mathcal{D} | \theta) &= \prod_n p(x_{U,n} | \theta) && \text{(i.i.d)} \\ &= \prod_n \prod_u p(x_{u,n} | x_{\pi_u,n}, \theta_u) \end{aligned}$$

$$\Rightarrow \log p(\mathcal{D} | \theta) = \sum_n \sum_u \log p(x_{u,n} | x_{\pi_u,n}, \theta_u) \quad \text{Log-likelihood}$$

Maximum Log-Likelihood

- Taking the **maximum log-likelihood** over parameter θ_u gives:

$$\operatorname{argmax}_{\theta_u} \log p(\mathcal{D}|\theta) = \operatorname{argmax}_{\theta_u} \sum_n \log p(x_{u,n}|x_{\pi_{u,n}}, \theta_u)$$

- We can ignore all terms that **do not involve** θ_u .
- Implies that it is sufficient to estimate θ_u with the **local subset of observations**, i.e. **sufficient statistics**:

$$\{x_{u,n}, x_{\pi_{u,n}}\}_{n=1}^N$$

Maximum A Posteriori (MAP)

- Taking the **maximum a posteriori (MAP)** over parameter θ_u gives:

$$\operatorname{argmax}_{\theta_u} \log p(\theta | \mathcal{D}) =$$

$$\operatorname{argmax}_{\theta_u} \left\{ \sum_n \log p(x_{u,n} | x_{\pi_{u,n}}, \theta_u) + \log p(\theta_u) \right\}$$

- $p(\theta_u)$ is the **conjugate prior** of the likelihood distribution.

Maximum Log-Likelihood: Discrete Case

- Likelihood is given by **Categorical distribution**:

$$\begin{aligned} p(x_u | x_{\pi_u}, \lambda_u) &= \text{Cat}_{x_u | x_{\pi_u}}[\lambda_u] \\ &= \prod_{c=1}^C \prod_{k=1}^K \lambda_{uck}^{x_{uck}} = \lambda_{uck}, \quad s.t. \quad \sum_k \lambda_{uck} = 1 \end{aligned}$$

- The **parameter** $\lambda_u = \{\lambda_{u11}, \dots, \lambda_{uck}, \dots, \lambda_{uCK}\}$.
- C is total number of states that X_{π_u} takes, and K is the total number of states that X_u takes.
- $x_{uck} = 1$ when $X_u = k$ and $X_{\pi_u} = c$, $x_{uck} = 0$ otherwise.

Maximum Log-Likelihood: Discrete Case

- Putting the likelihood into the **maximum log-likelihood**, we get:

$$\operatorname{argmax}_{\lambda_{uc1}, \dots, \lambda_{ucK}} \sum_{n=1}^N \sum_{c=1}^C \sum_{k=1}^K \log \lambda_{uck}^{x_{uck,n}}, \quad s.t. \quad \sum_k \lambda_{uck} = 1$$

- Sum over c is dropped** since we optimize over the parameters of each configuration of X_{π_u} :

$$\Rightarrow \operatorname{argmax}_{\lambda_{uc1}, \dots, \lambda_{ucK}} \sum_{n=1}^N \sum_{k=1}^K \log \lambda_{uck}^{x_{uck,n}}, \quad s.t. \quad \sum_k \lambda_{uck} = 1$$

times we observe $(x_u = k, x_{\pi_u} = c)$

$$\Rightarrow \operatorname{argmax}_{\lambda_{uc1}, \dots, \lambda_{ucK}} \sum_{k=1}^K \log \lambda_{uck}^{N_{uck}}, \quad s.t. \quad \sum_k \lambda_{uck} = 1$$

Maximum Log-Likelihood: Discrete Case

- Applying **Lagrange multiplier** ν on the constraint, we get the **auxiliary function** :

$$\mathcal{L} = \sum_{k=1}^K N_{uck} \log \lambda_{uck} + \nu \left(\sum_k \lambda_{uck} - 1 \right)$$

- Take derivative of \mathcal{L} w.r.t λ_{uck} and ν , set to zero and solve for λ_{uck} :

$$\hat{\lambda}_{uck} = \frac{N_{uck}}{\sum_{m=1}^K N_m}$$

Normalized counts of # times
we observed $x_u = k, x_{\pi_u} = c$

Maximum A Posteriori: Discrete Case

- We use the **Dirichlet distribution** as conjugate prior:

$$p(\lambda_{uc1}, \dots, \lambda_{ucK}) = \text{Dir}_{\lambda_{uc1}, \dots, \lambda_{ucK}}[\alpha_{uc1}, \dots, \alpha_{ucK}]$$

$$= \frac{\Gamma[\sum_{k=1}^K \alpha_{uck}]}{\prod_{k=1}^K \Gamma[\alpha_{uck}]} \prod_{k=1}^K \lambda_{uck}^{\alpha_{uck}-1},$$

$$\text{s.t. } \lambda_{uck} \in [0,1], \sum_k \lambda_{uck} = 1$$

- K **hyperparameters** $\alpha_{uck} > 1$ for each random variable X_u and **a state of its parents** $X_{\pi_u} = c$.

Maximum A Posteriori: Discrete Case

- Putting the **conjugate prior** into

$$\operatorname{argmax}_{\lambda_u} \{ \sum_n \log p(x_{u,n} | x_{\pi_u,n}, \lambda_u) + \log p(\lambda_u) \},$$

- We get:

$$\operatorname{argmax}_{\lambda_{uc1}, \dots, \lambda_{ucK}} \{ \sum_n \log \operatorname{Cat}_{x_{u,n} | x_{\pi_u,n}} [\lambda_u] + \log \operatorname{Dir}_{\lambda_{uc1} \dots \lambda_{ucK}} [\alpha_{uc1}, \dots, \alpha_{ucK}] \}, \text{ s.t. } \sum_k \lambda_{uck} = 1$$

$$= \operatorname{argmax}_{\lambda_{uc1}, \dots, \lambda_{ucK}} \left\{ \sum_k \left(\sum_n \log \lambda_{uck}^{x_{uck,n}} + \log \lambda_{uck}^{\alpha_{uck}-1} \right) + \log \frac{\Gamma[\sum_{k=1}^K \alpha_{uck}]}{\prod_{k=1}^K \Gamma[\alpha_{uck}]} \right\}, \text{ s.t. } \sum_k \lambda_{uck} = 1$$

Independent of λ_{uck}

times we observe $(x_u = k, x_{\pi_u} = c)$

$$= \operatorname{argmax}_{\lambda_{uc1}, \dots, \lambda_{ucK}} \{ \sum_k (\log \lambda_{uck}^{N_{uck} + \alpha_{uck} - 1}) \}, \text{ s.t. } \sum_k \lambda_{uck} = 1$$

Maximum A Posteriori: Discrete Case

- Applying **Lagrange multiplier** ν on the constraint, we get the **auxiliary function** :

$$\mathcal{L} = \sum_{k=1}^K (N_{uck} + \lambda_{uck} - 1) \log \lambda_{uck} + \nu \left(\sum_k \lambda_{uck} - 1 \right)$$

- Take derivative of \mathcal{L} w.r.t λ_{uck} and ν , set to zero and solve for λ_{uck} :

$$\hat{\lambda}_{uck} = \frac{N_{uck} + \alpha_{uck} - 1}{\sum_{m=1}^K (N_{ucm} + \alpha_{ucm} - 1)}$$

Maximum Log-Likelihood: Continuous Case

- Likelihood is given by **linear-Gaussian model**:

$$\begin{aligned} p(x_u | x_{\pi_u}, \theta_{x_u | x_{\pi_u}}) &= \text{Norm}_{x_u | x_{\pi_u}}[w_{u0}, \dots, w_{uC}, \sigma_u^2] \\ &= \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp \left\{ -0.5 \frac{(x_u - (\sum_{c \in x_{\pi_u}} w_{uc} x_{uc} + w_{u0}))^2}{\sigma_u^2} \right\} \end{aligned}$$

- The **parameter** $\theta_{x_u | x_{\pi_u}} = \{w_{u0}, \dots, w_{uC}, \sigma_u^2\}$.
- Mean $\mu_u = \sum_{c \in x_{\pi_u}} w_{uc} x_{uc} + w_{u0}$ is a **weighted sum** of the parent nodes x_{π_u} .
- C is the total number of parent nodes.

Maximum Log-Likelihood: Continuous Case

- Putting the likelihood into the **maximum log-likelihood**, we get:

$$\begin{aligned} & \operatorname{argmax}_{\theta_{x_u|x_{\pi_u}}} \sum_{n=1}^N \log p(x_{u,n} | x_{\pi_u,n}, \theta_{x_u|x_{\pi_u}}) \\ &= \operatorname{argmax}_{\theta_{x_u|x_{\pi_u}}} \sum_{n=1}^N \log \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp \left\{ -\frac{\left(x_{u,n} - \left(\sum_{c \in x_{\pi_u}} w_{uc} x_{uc,n} + w_{u0}\right)\right)^2}{2\sigma_u^2} \right\} \\ &= \operatorname{argmax}_{\theta_{x_u|x_{\pi_u}}} \underbrace{\sum_{n=1}^N \left\{ -\frac{1}{2} \log(2\pi\sigma_u^2) - \frac{1}{2\sigma_u^2} \left(x_{u,n} - \left(\sum_{c \in x_{\pi_u}} w_{uc} x_{uc,n} + w_{u0}\right)\right)^2 \right\}}_L \end{aligned}$$

Maximum Log-Likelihood: Continuous Case

- Take the **derivative of L w.r.t to w_{u0}, w_{uC}** and equating to zero, we get **$C + 1$ equations**:

$$\frac{\partial L}{\partial w_{u0}} = \sum_{n=1}^N \left(x_{u,n} - (w_{u1}x_{u1,n} + \dots + w_{uC}x_{uC,n} + w_{u0}) \right) = 0$$

$$\frac{\partial L}{\partial w_{u1}} = \sum_{n=1}^N \left(x_{u,n} - (w_{u1}x_{u1,n} + \dots + w_{uC}x_{uC,n} + w_{u0}) \right) x_{u1,n} = 0$$

\vdots

$$\frac{\partial L}{\partial w_{uC}} = \sum_{n=1}^N \left(x_{u,n} - (w_{u1}x_{u1,n} + \dots + w_{uC}x_{uC,n} + w_{u0}) \right) x_{uC,n} = 0$$

- Which can be used to solve for the **$C + 1$ unknowns** $w_{u0}, w_{u1}, \dots, w_{uC}$.
- Finally, take the **derivative of L w.r.t to σ_u^2** and equate to zero, to solve for σ_u^2 .

Maximum A Posteriori: Continuous Case

- We define the **prior** of the linear Gaussian parameters $\theta_{x_u|x_{\pi_u}} = \{w_{u0}, \dots, w_{uC}, \sigma_u^2\}$ as:

$$\begin{aligned} p(w_{u0}, \dots, w_{uC}, \sigma_u^2) &= p(\sigma_u^2) p(w_{u0}, \dots, w_{uC} | \sigma_u^2) \\ &= p(\sigma_u^2) \prod_{c \in x_{\pi_u}} p(w_{uc} | \sigma_u^2) \quad (\text{Naïve Bayes}) \end{aligned}$$

- $p(w_{uc} | \sigma_u^2)$ follows the **univariate normal distribution**:

$$p(w_{uc} | \mu_u, \sigma_u^2) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp -\frac{(w_{uc} - \mu_u)^2}{2\sigma_u^2} = \text{Norm}_{w_{uc}}[\mu_u, \sigma_u^2]$$

- where μ_u is a **hyperparameter**.

Maximum A Posteriori: Continuous Case

- $p(\sigma_u^2)$ follows the **inverse gamma distribution**:

$$p(\sigma_u^2 | \alpha_u, \beta_u) = \frac{\beta_u^{\alpha_u}}{\Gamma(\alpha_u)} (\sigma_u^2)^{-\alpha_u-1} \exp\left(-\frac{\beta_u}{\sigma_u^2}\right) = \text{InvGam}_{\sigma_u^2} [\alpha_u, \beta_u]$$

- (α_u, β_u) are the **hyperparameters** that describe the **shape** and **scale** of the distribution.
- $\alpha_u > 0$ and $\beta_u > 0$.
- $\Gamma(\alpha_u)$ denotes the **gamma function**.

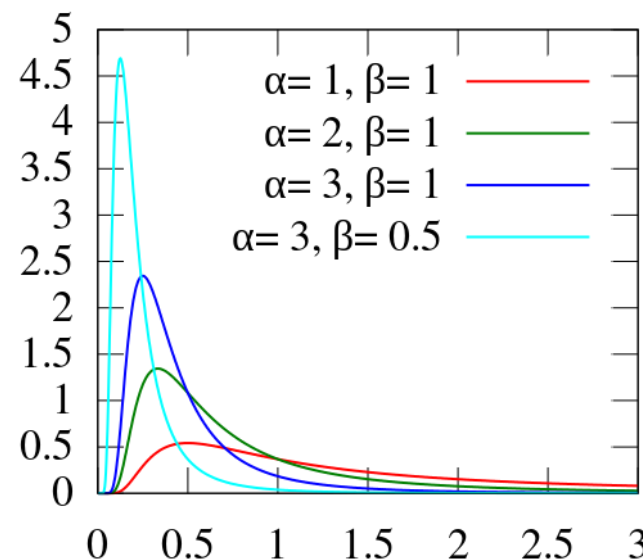


Image source: https://en.wikipedia.org/wiki/Inverse-gamma_distribution

Maximum A Posteriori: Continuous Case

- Putting the **likelihood** and **conjugate prior** into

$$\operatorname{argmax}_{\theta_{x_u|x_{\pi_u}}} \left\{ \sum_n \log p(x_{u,n}|x_{\pi_u,n}, \theta_{x_u|x_{\pi_u}}) + \underbrace{\log p(\theta_{x_u|x_{\pi_u}})}_{p(\sigma_u^2)p(w_{u0}, \dots, w_{uC}|\sigma_u^2)} \right\},$$

- We get:

$$\operatorname{argmax}_{\theta_{x_u|x_{\pi_u}}} \left\{ \sum_n \log \operatorname{Norm}_{x_{u,n}|x_{\pi_u,n}}[w_{u0}, \dots, w_{uC}, \sigma_u^2] + \right. \\ \left. \underbrace{\log \operatorname{InvGam}_{\sigma_u^2}[\alpha_u, \beta_u] + \sum_c \log \operatorname{Norm}_{w_{uc}}[\mu_u, \sigma_u^2]}_L \right\}$$

Maximum A Posteriori: Continuous Case

- Take the **derivative of L w.r.t to w_{u0}, w_{uC}** and equating to zero:

$$\frac{\partial L}{\partial w_{u0}} = 0$$

$$\frac{\partial L}{\partial w_{u1}} = 0$$

$$\vdots$$

$$\frac{\partial L}{\partial w_{uC}} = 0$$

$C + 1$ equations to solve for the
 $C + 1$ unknowns $\{w_{u0}, \dots, w_{uC}\}$.

- Finally, take the **derivative of L w.r.t to σ_u^2** and equate to zero, to solve for σ_u^2 .

Parameter Learning: UGM (MRF)

- Consider a Markov Random Field (MRF) in **log-linear form**, where c indexes the cliques:

$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp \left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y}) \right)$$

- The **scaled log-likelihood** is given by:

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_i \log p(\mathbf{y}_i|\boldsymbol{\theta}) = \frac{1}{N} \sum_i \left[\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y}_i) - \log Z(\boldsymbol{\theta}) \right]$$

Parameter Learning: UGM (MRF)

- Since MRFs are in the **exponential family**, we know that this function is **convex** in θ .
- So, it has a **unique global maximum**, which we can find using **gradient-based optimizers**.
- In particular, the **derivative for the weights** of a particular clique c is given by:

$$\frac{\partial \ell}{\partial \theta_c} = \frac{1}{N} \sum_i^N \left[\phi_c(y_i) - \frac{\partial}{\partial \theta_c} \log Z(\theta) \right]$$

Parameter Learning: UGM (MRF)

- The derivative of the **log partition function** w.r.t. θ_c is the expectation of the c^{th} feature under the model:

$$\frac{\partial \log Z(\boldsymbol{\theta})}{\partial \theta_c} = \mathbb{E}[\phi_c(\mathbf{y})|\boldsymbol{\theta}] = \sum_{\mathbf{y}} \phi_c(\mathbf{y}) p(\mathbf{y}|\boldsymbol{\theta})$$

Proof:

$$\begin{aligned} \frac{\partial \log Z(\theta)}{\partial \theta_c} &= \frac{1}{Z(\theta)} \frac{\partial Z(\theta)}{\partial \theta_c}, \quad \text{where} \quad Z(\theta) = \sum_{\mathbf{y}} \exp(\sum_c \theta_c^T \phi_c(\mathbf{y})) \\ &\Rightarrow \frac{\partial Z(\theta)}{\partial \theta_c} = \sum_{\mathbf{y}} \exp(\sum_c \theta_c^T \phi_c(\mathbf{y})) \phi_c(\mathbf{y}) \\ \Rightarrow \frac{\partial \log Z(\theta)}{\partial \theta_c} &= \frac{1}{Z(\theta)} \sum_{\mathbf{y}} \phi_c(\mathbf{y}) \exp(\sum_c \theta_c^T \phi_c(\mathbf{y})) \\ &= \sum_{\mathbf{y}} \phi_c(\mathbf{y}) \underbrace{\frac{1}{Z(\theta)} \exp(\sum_c \theta_c^T \phi_c(\mathbf{y}))}_{p(\mathbf{y}|\theta)} = \sum_{\mathbf{y}} \phi_c(\mathbf{y}) p(\mathbf{y}|\theta) \end{aligned}$$

□

Parameter Learning: UGM (MRF)

- Hence the **gradient of the log-likelihood** is:

$$\frac{\partial \ell}{\partial \theta_c} = \underbrace{\left[\frac{1}{N} \sum_i^N \phi_c(\mathbf{y}_i) \right]}_{\text{Clamped term}} - \underbrace{\mathbb{E}[\phi_c(\mathbf{y})]}_{\text{Unclamped/contrastive term}}$$

- Clamped term:** \mathbf{y} is fixed to its observed values.
- Unclamped/contrastive term:** \mathbf{y} is a free variable.
- Unclamped term **requires inference** in the model, once per gradient step, and this makes UGM learning **much slower** than DGM.

Parameter Learning: UGM (MRF)

- Gradient of the log-likelihood can be rewritten as:

$$\frac{\partial l}{\partial \theta_c} = E_{p_{emp}}[\phi_c(y)] - E_{p(y|\theta)}[\phi_c(y)]$$

- $E_{p_{emp}}[\phi_c(y)] = \frac{1}{N} \sum_{i=1}^N \phi_c(y_i)$: Expected feature vector according to the **empirical distribution**.
- $E_{p(y|\theta)}[\phi_c(y)]$: Expected feature vector according to the **model's distribution**.

Parameter Learning: UGM (MRF)

- At the optimum, the **gradient will be zero**:

$$E_{p_{emp}}[\phi_c(y)] - E_{p(y|\theta)}[\phi_c(y)] = 0$$

- **Problem:** $E_{p(y|\theta)}[\phi_c(y)] = \sum_y \phi_c(y) p(y|\theta)$ **cannot be evaluated in closed-form** in terms of the unknown parameters θ !

We **CANNOT** solve for the parameters θ in closed-form!!!

Parameter Learning: UGM (MRF)

Solution:

Use gradient-based optimizers!

- However, the **gradient requires inference**:

$$\frac{\partial l}{\partial \theta_c} = E_{p_{emp}}[\phi_c(y)] - E_{p(y|\theta)}[\phi_c(y)]$$

Requires sum over all states of y , which is intractable

- Gradient is **intractable**, hence learning also becomes intractable.
- We can combine **approximate inference** with gradient-based learning.

Method 1: Stochastic Maximum Likelihood

- This is a **stochastic gradient descent** method.
- We **iteratively updates** the parameter θ_{k+1} at the k step using the parameter and gradient from the previous step:

$$\theta_{k+1} \leftarrow \theta_k - \eta g_k$$

- η is the **step size**, or **learning rate**.
- $g_k \approx \frac{\partial l}{\partial \theta_c}$ is the gradient that can be approximated with **Markov Chain Monte Carlo (MCMC)**, i.e. sampling.

Method 1: Stochastic Maximum Likelihood

Algorithm : Stochastic maximum likelihood for fitting an MRF

```
1 Initialize weights  $\theta$  randomly;
2  $k = 0, \eta = 1$  ;
3 for each epoch do
4   for each minibatch of size  $B$  do    // split the observed data  $y_i, \forall i = 1 \dots N$  into sets of size  $B$ 
5     for each sample  $s = 1 : S$  do
6       Sample  $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\theta_k)$  ;
7        $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S} \sum_{s=1}^S \phi(\mathbf{y}^{s,k})$ ;
8       for each training case  $i$  in minibatch do
9          $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$  ;
10       $\mathbf{g}_k = \frac{1}{B} \sum_{i \in B} \mathbf{g}_{ik}$ ;
11       $\theta_{k+1} = \theta_k - \eta \mathbf{g}_k$ ;
12       $k = k + 1$ ;
13      Decrease step size  $\eta$ ;
```

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Method 1: Stochastic Maximum Likelihood

Algorithm : Stochastic maximum likelihood for fitting an MRF

```
1 Initialize weights  $\theta$  randomly;
2  $k = 0, \eta = 1$  ;
3 for each epoch do
4   for each minibatch of size B do // split the observed data  $y_i, \forall i = 1 \dots N$  into sets of size  $B$ 
5     for each sample  $s = 1 : S$  do
6       Sample  $y^{s,k} \sim p(y|\theta_k)$  ; // draw S samples from the model's distribution  $p(y|\theta_k)$ 
7        $\hat{E}(\phi(y)) = \frac{1}{S} \sum_{s=1}^S \phi(y^{s,k})$ ; // note that we fixed the parameter at  $\theta_k$  (current estimate)
8     for each training case  $i$  in minibatch do
9        $g_{ik} = \phi(y_i) - \hat{E}(\phi(y))$  ;
10     $g_k = \frac{1}{B} \sum_{i \in B} g_{ik}$ ;
11     $\theta_{k+1} = \theta_k - \eta g_k$ ;
12     $k = k + 1$ ;
13  Decrease step size  $\eta$ ;
```

* We will discuss more about MCMC sampling in Lecture 10

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Method 1: Stochastic Maximum Likelihood

Algorithm : Stochastic maximum likelihood for fitting an MRF

```
1 Initialize weights  $\theta$  randomly;
2  $k = 0, \eta = 1$  ;
3 for each epoch do
4   for each minibatch of size B do      // split the observed data  $y_i, \forall i = 1 \dots N$  into sets of size  $B$ 
5     for each sample  $s = 1 : S$  do
6       Sample  $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\theta_k)$  ;    // draw  $S$  samples from the posterior distribution  $p(\mathbf{y}|\theta_k)$ 
7        $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S} \sum_{s=1}^S \phi(\mathbf{y}^{s,k})$ ; // note that we fixed the parameter at  $\theta_k$  (current estimate)
8       for each training case  $i$  in minibatch do
9          $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$  ;    // compute the approximated gradient,  $\mathbf{g}_k \approx \frac{\partial l}{\partial \theta}$ 
10         $\mathbf{g}_k = \frac{1}{B} \sum_{i \in B} \mathbf{g}_{ik}$ ;
11         $\theta_{k+1} = \theta_k - \eta \mathbf{g}_k$ ;
12         $k = k + 1$ ;
13      Decrease step size  $\eta$ ;
```

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Method 1: Stochastic Maximum Likelihood

Algorithm : Stochastic maximum likelihood for fitting an MRF

```
1 Initialize weights  $\theta$  randomly;
2  $k = 0, \eta = 1$  ;
3 for each epoch do
4   for each minibatch of size  $B$  do    // split the observed data  $y_i, \forall i = 1 \dots N$  into sets of size  $B$ 
5     for each sample  $s = 1 : S$  do
6       Sample  $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\theta_k)$  ;    // draw  $S$  samples from the posterior distribution  $p(\mathbf{y}|\theta_k)$ 
7        $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S} \sum_{s=1}^S \phi(\mathbf{y}^{s,k})$ ;    // note that we fixed the parameter at  $\theta_k$  (current estimate)
8       for each training case  $i$  in minibatch do
9          $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$  ;    // compute the approximated gradient,  $\mathbf{g}_k \approx \frac{\partial l}{\partial \theta}$ 
10         $\mathbf{g}_k = \frac{1}{B} \sum_{i \in B} \mathbf{g}_{ik}$ ;
11         $\theta_{k+1} = \theta_k - \eta \mathbf{g}_k$ ;    // stochastic gradient descent update step
12         $k = k + 1$ ;
13      Decrease step size  $\eta$ ;
```

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Method 2: Iterative Proportional Fitting (IFP)

- An **alternative derivation** of the gradient in term of $\psi_c(y_c)$ is given by:

$$\begin{aligned}\ell &= \log \prod_i \frac{1}{Z} \prod_c \psi_c(y_{i,c}) \\ &= \sum_i \sum_c \log \psi_c(y_{i,c}) - N \log Z \\ &= \sum_c \sum_{y_c} \underbrace{N(y_c)}_{\text{# times clique } c \text{ is in configuration } y_c \text{ in the data}} \log \psi_c(y_c) - N \log Z\end{aligned}$$

$N(y_c) = \sum_i \delta(y_c, y_{i,c})$: # times clique c is in configuration y_c in the data

$$\Rightarrow \frac{\partial \ell}{\partial \psi_c(y_c)} = \frac{N(y_c)}{\psi_c(y_c)} - \underbrace{N \frac{\partial}{\partial \psi_c(y_c)} \log Z}_{\text{No closed-form!}}$$

No closed-form!

Method 2: Iterative Proportional Fitting (IFP)

- Derivative of the **log partition function**:

$$\begin{aligned}\frac{\partial}{\partial \psi_c(y_c)} \log Z &= \frac{1}{Z} \frac{\partial Z}{\partial \psi_c(y_c)} \\&= \frac{1}{Z} \frac{\partial}{\partial \psi_c(y_c)} \sum_y \prod_D \psi_D(y_D) \\&= \frac{1}{Z} \sum_y \delta(y, y_c) \frac{\partial}{\partial \psi_c(y_c)} \prod_D \psi_D(y_D) \\&= \frac{1}{Z} \sum_y \delta(y, y_c) \prod_{D \setminus c} \psi_D(y_D) \\&= \sum_y \delta(y, y_c) \frac{1}{\psi_c(y_c)} \boxed{\frac{1}{Z} \prod_D \psi_D(y_D)} \quad \leftarrow p(y) \\&= \frac{1}{\psi_c(y_c)} \boxed{\sum_y \delta(y, y_c) p(y)} \quad \leftarrow \sum_{y \setminus y_c} p(y) \\&= \frac{p(y_c | \psi)}{\psi_c(y_c)} \quad \leftarrow \text{Conditioned on all potentials since } p(y_c | \psi) \text{ is obtained from marginalization of the full distribution } p(y)\end{aligned}$$

Method 2: Iterative Proportional Fitting (IPF)

- Putting the derivative of the **log partition function** back into the **gradient**, and equating to zero we get:

$$\frac{N(y_c)}{\psi_c(y_c)} - N \frac{p(y_c|\psi)}{\psi_c(y_c)} = 0$$

- From this we infer:

$$p_{emp}(y_c) \leftarrow \boxed{\frac{N(y_c)}{N}} \frac{1}{\psi_c(y_c)} = \frac{p(y_c|\psi)}{\psi_c(y_c)}$$

- We can solve for $\psi_c(y_c)$ iteratively using the **fixed point equation**:

$$\psi_c^{t+1}(\mathbf{y}_c) = \psi_c^t(\mathbf{y}_c) \times \frac{p_{emp}(\mathbf{y}_c)}{p(\mathbf{y}_c|\psi^t)}$$

Element-wise multiplication

Method 2: Iterative Proportional Fitting (IPF)

Algorithm : Iterative Proportional Fitting algorithm for tabular MRFs

```
1 Initialize  $\psi_c = 1$  for  $c = 1 : C$ ;  
2 repeat  
3   for  $c = 1 : C$  do  
4      $p_c = p(\mathbf{y}_c | \psi)$ ; // do marginalization with current  $\psi$ :  $\sum_{\mathbf{y} \setminus \mathbf{y}_c} p(\mathbf{y})$   
5      $\hat{p}_c = p_{\text{emp}}(\mathbf{y}_c)$ ;  
6      $\psi_c = \psi_c * \frac{\hat{p}_c}{p_c}$  ;  
7 until converged;
```

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Method 2: Iterative Proportional Fitting (IPF)

Algorithm : Iterative Proportional Fitting algorithm for tabular MRFs

```
1 Initialize  $\psi_c = 1$  for  $c = 1 : C$ ;  
2 repeat  
3   for  $c = 1 : C$  do  
4      $p_c = p(\mathbf{y}_c | \psi)$ ;           // do marginalization with current  $\psi$ :  $\sum_{\mathbf{y} \setminus \mathbf{y}_c} p(\mathbf{y})$   
5      $\hat{p}_c = p_{\text{emp}}(\mathbf{y}_c)$ ;       // compute empirical probability of current clique  $c$   
6      $\psi_c = \psi_c * \frac{p_c}{\hat{p}_c}$  ;  
7 until converged;
```

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Method 2: Iterative Proportional Fitting (IPF)

Algorithm : Iterative Proportional Fitting algorithm for tabular MRFs

```
1 Initialize  $\psi_c = 1$  for  $c = 1 : C$ ;  
2 repeat  
3   for  $c = 1 : C$  do  
4      $p_c = p(\mathbf{y}_c | \psi)$ ;           // do marginalization with current  $\psi$ :  $\sum_{\mathbf{y} \setminus \mathbf{y}_c} p(\mathbf{y})$   
5      $\hat{p}_c = p_{\text{emp}}(\mathbf{y}_c)$ ;       // compute empirical probability of current clique  $c$   
6      $\psi_c = \psi_c * \frac{\hat{p}_c}{p_c}$ ;    // iterative proportional fitting  
7 until converged;
```

Source: Kevin Murphy, "Machine Learning: a probabilistic perspective"

Maximum A Posteriori (MAP): MRF (UGM)

- We can also do MAP to learn the unknown parameters in UGM, where we add a **prior term**:

$$\operatorname{argmax}_{\theta} \left\{ \sum_i \log p(y_i | \theta) + \underbrace{\log p(\theta)}_{\text{Prior term}} \right\}$$

- A **Gaussian prior** is often use:

$$p(\theta) = \mathcal{N}(\theta | \mu, \Sigma) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\theta - \mu)^T \Sigma^{-1}(\theta - \mu)\right]$$

- Where (μ, Σ) are the **hyperparameters**.

Parameter Learning: UGM (CRF)

- Consider a Conditional Random Field (CRF) in **log-linear form**, where c indexes the cliques:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \prod_c \exp(\mathbf{w}_c^T \phi_c(\mathbf{x}, \mathbf{y}_c))$$

- $\phi_c(\mathbf{x}, \mathbf{y}_c)$ is a **feature vector** derived from **the global inputs** \mathbf{x} and the **local set of labels** \mathbf{y}_c .

Parameter Learning: UGM (CRF)

- We can modify the **gradient based optimization** of MRFs to the CRF, the **scaled log-likelihood** becomes:

$$\begin{aligned}\ell(\mathbf{w}) &\triangleq \frac{1}{N} \sum_i^N \log p(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}) \\ &= \frac{1}{N} \sum_i^N \left[\sum_c \mathbf{w}_c^T \phi_c(\mathbf{y}_i, \mathbf{x}_i) - \log Z(\mathbf{w}, \mathbf{x}_i) \right]\end{aligned}$$

- The **gradient** now becomes:

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{w}_c} &= \frac{1}{N} \sum_i^N \left[\phi_c(\mathbf{y}_i, \mathbf{x}_i) - \frac{\partial}{\partial \mathbf{w}_c} \log Z(\mathbf{w}, \mathbf{x}_i) \right] \\ &= \frac{1}{N} \sum_i^N [\phi_c(\mathbf{y}_i, \mathbf{x}_i) - \mathbb{E}[\phi_c(\mathbf{y}, \mathbf{x}_i)]]\end{aligned}$$

Parameter Learning: UGM (CRF)

$$\frac{\partial \ell}{\partial \mathbf{w}_c} = \frac{1}{N} \sum_i^N [\phi_c(\mathbf{y}_i, \mathbf{x}_i) - \underbrace{\mathbb{E}[\phi_c(\mathbf{y}, \mathbf{x}_i)]}]$$

$$\mathbb{E}[\phi_c(\mathbf{y}, \mathbf{x}_i)] = \sum_{\mathbf{y}, \mathbf{x}_i} p(\mathbf{y}|\mathbf{x}_i, \mathbf{w}) \phi_c(\mathbf{y}, \mathbf{x}_i)$$

- The partition function **depends on** the observed data \mathbf{x}_i !
- This means that we **cannot** bring $\mathbb{E}[\phi_c(\mathbf{y}, \mathbf{x}_i)]$ out of the summation.
- We now must perform inference for every single training case inside each gradient step, which is **$O(N)$ times slower** than the MRF case.

Stochastic Gradient Descent

Algorithm: Stochastic Gradient Descent

```
1:  $w^* = \text{STOCHASTICGRADIENTDESCENT}(T, \eta)$ 
2: Input:
3:    $T$  // number of iterations
4:    $\eta_1, \dots, \eta_T$  // sequence of learning rates (can be the same)
5: Output:
6:    $w^* \in \mathbb{R}^D$  // learned weight vector
7: Algorithm:
8:    $w_{cur} \leftarrow 0$  // initialize parameters to 0
9:   for  $t=1, \dots, T$  do
10:     $(x^n, y^n) \in \mathcal{D}'$ 
11:     $d \leftarrow -\tilde{\nabla}_w^{(x^n, y^n)} \mathcal{L}(w_{cur})$ 
12:     $w_{cur} \leftarrow w_{cur} + \eta_t d$ 
13:   end for
14:  $w^* \leftarrow w_{cur}$ 
```

Source: “Structured Learning and Prediction in Computer Vision”, Sebastian Nowozin and Christoph H. Lampert, 2013

Stochastic Gradient Descent

Algorithm: Stochastic Gradient Descent

```
1:  $w^* = \text{STOCHASTICGRADIENTDESCENT}(T, \eta)$ 
2: Input:
3:    $T$  // number of iterations
4:    $\eta_1, \dots, \eta_T$  // sequence of learning rates (can be the same)
5: Output:
6:    $w^* \in \mathbb{R}^D$  // learned weight vector
7: Algorithm:
8:    $w_{cur} \leftarrow 0$  // initialize parameters to 0
9:   for  $t=1, \dots, T$  do
10:     $(x^n, y^n) \in \mathcal{D}'$  // pick random subset of data (often 1–3 elements), i.e.  $\mathcal{D}' \subset \mathcal{D}$ 
11:     $d \leftarrow -\tilde{\nabla}_w^{(x^n, y^n)} \mathcal{L}(w_{cur})$ 
12:     $w_{cur} \leftarrow w_{cur} + \eta_t d$ 
13:  end for
14:  $w^* \leftarrow w_{cur}$ 
```

This reduces the amount of computation to perform inference for every single training case!

Source: “Structured Learning and Prediction in Computer Vision”, Sebastian Nowozin and Christoph H. Lampert, 2013

Stochastic Gradient Descent

Algorithm: Stochastic Gradient Descent

```
1:  $w^* = \text{STOCHASTICGRADIENTDESCENT}(T, \eta)$ 
2: Input:
3:    $T$  // number of iterations
4:    $\eta_1, \dots, \eta_T$  // sequence of learning rates (can be the same)
5: Output:
6:    $w^* \in \mathbb{R}^D$  // learned weight vector
7: Algorithm:
8:    $w_{cur} \leftarrow 0$  // initialize parameters to 0
9:   for  $t=1, \dots, T$  do
10:     $(x^n, y^n) \in \mathcal{D}'$  // pick random subset of data (often 1–3 elements), i.e.  $\mathcal{D}' \subset \mathcal{D}$ 
11:     $d \leftarrow -\tilde{\nabla}_w^{(x^n, y^n)} \mathcal{L}(w_{cur})$  // gradient approximation
12:     $w_{cur} \leftarrow w_{cur} + \eta_t d$ 
13:   end for
14:  $w^* \leftarrow w_{cur}$ 
```

Source: “Structured Learning and Prediction in Computer Vision”, Sebastian Nowozin and Christoph H. Lampert, 2013

Gradient Approximation

- We will **approximate the gradient** with \mathcal{D}' and MCMC samples from the likelihood $p(y|\mathbf{x}_i, \mathbf{w})$.

$$\nabla \mathcal{L}(\mathbf{w}_c) = \frac{\partial \ell}{\partial \mathbf{w}_c} = \frac{1}{N} \sum_i [\phi_c(\mathbf{y}_i, \mathbf{x}_i) - \underbrace{\mathbb{E}[\phi_c(\mathbf{y}, \mathbf{x}_i)]}]$$

$$\mathbb{E}[\phi_c(\mathbf{y}, \mathbf{x}_i)] = \sum_{\mathbf{y}, \mathbf{x}_i} p(\mathbf{y}|\mathbf{x}_i, \mathbf{w}) \phi_c(\mathbf{y}, \mathbf{x}_i)$$



$$\tilde{\nabla} \mathcal{L}(\mathbf{w}_c) = \frac{1}{|\mathcal{D}'|} \sum_{(\mathbf{x}^n, \mathbf{y}^n) \in \mathcal{D}'} \left[\phi_c(\mathbf{x}^n, \mathbf{y}^n) - \underbrace{\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{x}_i, \mathbf{w})} \phi_c(\mathbf{x}^n, \mathbf{y})} \right]$$

Expectation is computed from samples drawn from the likelihood (MCMC sampling)

Stochastic Gradient Descent

Algorithm: Stochastic Gradient Descent

```
1:  $w^* = \text{STOCHASTICGRADIENTDESCENT}(T, \eta)$ 
2: Input:
3:    $T$  // number of iterations
4:    $\eta_1, \dots, \eta_T$  // sequence of learning rates (can be the same)
5: Output:
6:    $w^* \in \mathbb{R}^D$  // learned weight vector
7: Algorithm:
8:    $w_{cur} \leftarrow 0$  // initialize parameters to 0
9:   for  $t=1, \dots, T$  do
10:     $(x^n, y^n) \in \mathcal{D}'$  // pick random subset of data (often 1–3 elements), i.e.  $\mathcal{D}' \subset \mathcal{D}$ 
11:     $d \leftarrow -\tilde{\nabla}_w^{(x^n, y^n)} \mathcal{L}(w_{cur})$  // gradient approximation
12:     $w_{cur} \leftarrow w_{cur} + \eta_t d$  // weight update
13:   end for
14:  $w^* \leftarrow w_{cur}$ 
```

Source: “Structured Learning and Prediction in Computer Vision”, Sebastian Nowozin and Christoph H. Lampert, 2013

Maximum A Posteriori (MAP): CRF (UGM)

- We can also do a **maximum a posteriori** estimation of the unknown parameter in CRF:

$$\operatorname{argmax}_w \left\{ \sum_i \log p(y_i | x_i, w) + \log p(w) \right\}$$

- Where a **Gaussian prior** is often used for $p(w)$.

Summary

- We have looked at how to:
 1. Compute the unknown parameters of discrete/continuous **DGMs** using **MLE** and **MAP**.
 2. Compute the unknown parameters of **MRFs** using **stochastic maximum likelihood**, and **iterative proportional fitting**.
 3. Compute the unknown parameters of **CRFs** using **stochastic gradient descent**.