NUS National University of Singapore | School of Computing

CS5340 Uncertainty Modeling in AI

Lecture 6: Parameter Learning with Complete Data

Asst. Prof. Lee Gim Hee

AY 2020/21

Semester 1

Course Schedule

Week	Date	Topic	Remarks
1	12 Aug	Introduction to probabilistic reasoning	**1830hrs: MS Teams (Live Introduction)**
2	19 Aug	Bayesian networks (Directed graphical models)	
3	26 Aug	Markov random Fields (Undirected graphical models)	**1830hrs: Zoom discussions**
4	02 Sep	Variable elimination and belief propagation	**Assignment 1:** Belief propagation and maximal probability (15%)
5	09 Sep	Factor graph and the junction tree algorithm	
6	16 Sep	Parameter learning with complete data	**Assignment 1:** Due **Assignment 2:** Junction tree and parameter learning (15%) **1830hrs: Zoom discussions**
-	23 Sep	Recess week	**No lecture**
7	30 Sep	Mixture models and the EM algorithm	**Assignment 2:** Due **Online quiz 1 (20%)**
8	07 Oct	Hidden Markov Models (HMM)	**Assignment 3:** Hidden Markov model (15%)
9	14 Oct	Monte Carlo inference (Sampling)	**1830hrs: Zoom discussions**
10	21 Oct	Variational inference	**Assignment 3:** Due **Assignment 4:** MCMC Sampling (15%)
11	28 Oct	Variational Auto-Encoder and Mixture Density Networks	
12	04 Nov	Graph-cut and alpha expansion	**Assignment 4:** Due **1830hrs: Zoom discussions**
-	11 Nov	--	**Online quiz 2 (20%)**

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 identical and independently 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 $p(x_1, \dots x_M | \theta)$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} [p(x_1, \dots x_M | \theta)]$$

$$= \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^{N} p\left(x_{1,i}, \dots, x_{M,i} \mid \theta\right) \right] \qquad \text{(i.i.d)}$$

Maximum a Posteriori (MAP)

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

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} [p(\theta | x_1, \dots x_M)]$$

$$= \underset{\theta}{\operatorname{argmax}} \left[\frac{p(x_1, \dots x_M | \theta) p(\theta)}{p(x_1, \dots x_M)} \right]$$

(Bayes' rule)

$$= \underset{\theta}{\operatorname{argmax}} \left[\frac{\prod_{i=1}^{N} p\left(x_{1,i}, \dots, x_{M,i} \middle| \theta\right) p(\theta)}{p(x_1, \dots x_M)} \right]$$

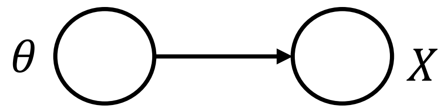
(i.i.d)

$$= \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^{N} p\left(x_{1,i}, \dots, x_{M,i} \middle| \theta\right) p(\theta) \right]$$

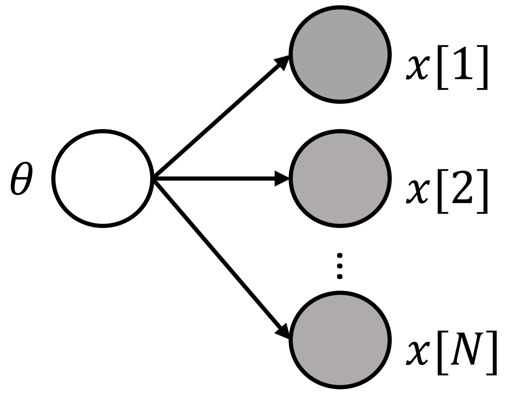
($p(x_1, \dots x_M)$ is removed since it is independent of θ)

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 an 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:

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

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)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}[p(x|\theta)]$$

$$= \underset{\theta}{\operatorname{argmax}}\left[\prod_{i=1}^{N} p(x[i]|\,\theta)\right] \qquad \text{(i.i.d)}$$

Likelihood given by pdf

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

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 1: Maximum Likelihood Estimation (MLE)

Algebraically:

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

where

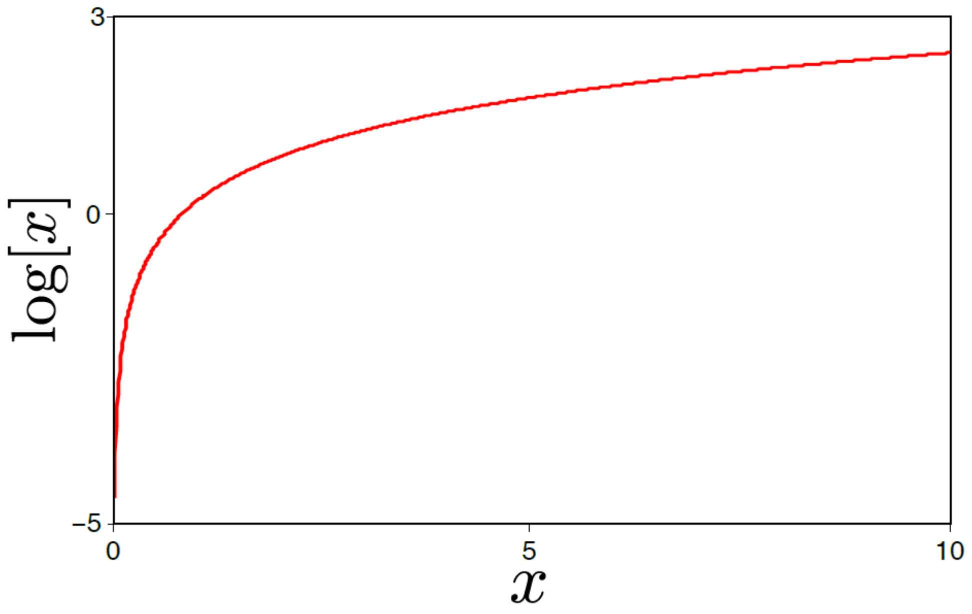
$$p(x|\mu,\sigma^2) = \prod_{i=1}^{N} \mathrm{Norm}_{x[i]}\,[\mu,\sigma^2],$$

or alternatively, we can maximize the logarithm:

$$\hat{\mu}, \hat{\sigma}^2 = \underset{\mu,\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log\left[\mathrm{Norm}_{x[i]}[\mu,\sigma^2]\right]$$

$$= \underset{\mu,\sigma^2}{\operatorname{argmax}} \left[-0.5N\log\,[2\pi] - 0.5N\log\sigma^2 - 0.5\sum_{i=1}^{N}\frac{(x[i]-\mu)^2}{\sigma^2}\right]$$

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.

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

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 1: Maximum Likelihood Estimation (MLE)

$$\hat{\mu}, \hat{\sigma}^2 = \underset{\mu,\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log\left[\text{Norm}_{x[i]}[\mu, \sigma^2]\right]$$

$$= \underset{\mu,\sigma^2}{\operatorname{argmax}} \underbrace{\left[-0.5N\log[2\pi] - 0.5N\log\sigma^2 - 0.5\sum_{i=1}^{N}\frac{(x[i]-\mu)^2}{\sigma^2}\right]}_{L}$$

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

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

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

Least Squares

Maximum likelihood for the normal distribution...

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

$$= \underset{\mu}{\operatorname{argmax}} \left[-\sum_{i=1}^{N} (x[i]-\mu)^2 \right]$$

$$= \underset{\mu}{\operatorname{argmin}} \left[\sum_{i=1}^{N} (x[i]-\mu)^2 \right]$$

...gives `least squares' fitting criterion.

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 2: Maximum a Posteriori (MAP)

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

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

$$p(\mu, \sigma^2) = \text{NormInvGam}_{\mu,\sigma^2}[\alpha, \beta, \gamma, \delta]$$

$$= \frac{\sqrt{\gamma}}{\sigma\sqrt{2\pi}} \frac{\beta^\alpha}{\Gamma[\alpha]} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left[-\frac{2\beta + \gamma(\delta - \mu)^2}{2\sigma^2}\right]$$

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 2: Maximum a Posteriori (MAP)

$$\hat{\mu}, \hat{\sigma}^2 = \underset{\mu,\sigma^2}{\operatorname{argmax}} \left[\prod_{i=1}^{N} p(x[i] \mid \mu, \sigma^2) \, p(\mu, \sigma^2) \right]$$

$$= \underset{\mu,\sigma^2}{\operatorname{argmax}} \left[\prod_{i=1}^{N} \mathrm{Norm}_{x[i]}[\mu, \sigma^2] \, \mathrm{NormInvGam}_{\mu,\sigma^2}[\alpha, \beta, \gamma, \delta] \right]$$

Maximize the logarithm:

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

Special Case (Continuous DGM): Univariate Normal Distribution

Approach 2: Maximum a Posteriori (MAP)

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

Taking derivatives and setting to zero:

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

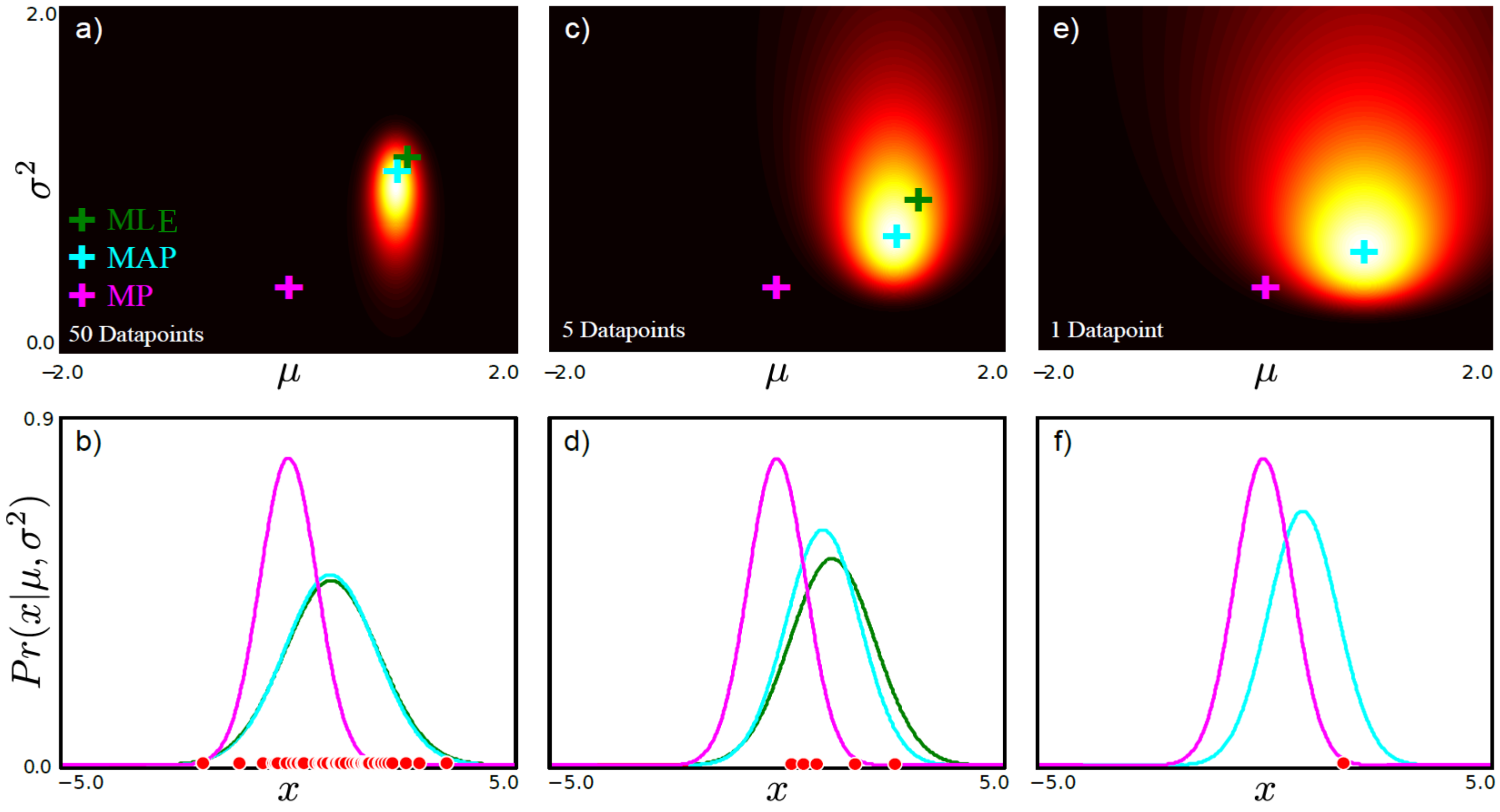
We get:

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

$$= \frac{N\bar{x} + \gamma\delta}{N + \gamma}$$

Comparing MLE and MAP

More data points → MAP is closer to MLE
Fewer data points → MAP is closer to MP



50 data points   5 data points   1 data points

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 $\boldsymbol{X} : \{\boldsymbol{x}[1], \dots, \boldsymbol{x}[N]\}$, where $\boldsymbol{x}[i] = \mathbf{e}_k$ is a vector with all zero elements expect *k*th e.g. $[0,0,0,1,0,0]$, where *K=6*.

Recall that the categorical distribution is given by:

$$p(\boldsymbol{X} = \mathbf{e}_k | \, \theta) = \mathrm{Cat}_{\boldsymbol{x}}[\lambda] = \prod_{k=1}^{K} \lambda_k^{\overset{k\text{th element of } \mathbf{e}_k}{x_k}} = \lambda_k$$

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)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}[p(\boldsymbol{x}|\theta)]$$

$$= \underset{\theta}{\operatorname{argmax}}\left[\prod_{i=1}^{N} p(\boldsymbol{x}[i]|\theta)\right] \qquad \text{(i.i.d)}$$

Likelihood given by pdf

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

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 1: Maximum Likelihood Estimation (MLE)

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

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

kth element of $\boldsymbol{x}_i$, $x_{ik} \in \{0,1\}$

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

N_k = # times we observed bin k

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

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 :

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

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

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

Normalized counts of # times we observed bin *k*

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 2: Maximum a Posteriori (MAP)

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

Likelihood: categorical distribution

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

Prior: conjugate prior – Dirichlet distribution

$$p(\lambda_1, \ldots, \lambda_K) = \text{Dir}_{\lambda_{1\ldots K}}[\alpha_1, \ldots \alpha_K]$$

$$= \frac{\Gamma[\sum_{k=1}^{K} \alpha_k]}{\prod_{k=1}^{K} \Gamma[\alpha_k]} \prod_{k=1}^{K} \lambda_k^{\alpha_k - 1}, \qquad \text{s.t. } \lambda_k \in [0,1], \; \sum_k \lambda_k = 1$$

Special Case (Discrete DGM): Univariate Categorical Distribution

Approach 2: Maximum a Posteriori (MAP)

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

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

Independent of $\lambda \Rightarrow$ can be ignored

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

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

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:

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

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

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

Same result as MLE with a uniform prior $\alpha_{1\ldots k} = 1$

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

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:

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

Complete Observation

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

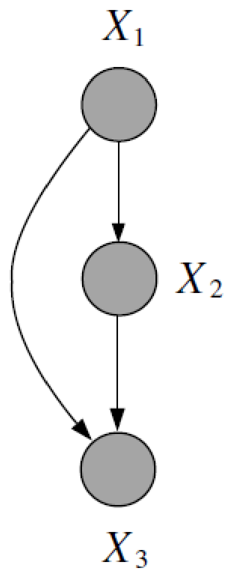


Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Complete Observation

- For N i.i.d. observations, the graphical model simply becomes an augmentation of N disconnected replicates of G.

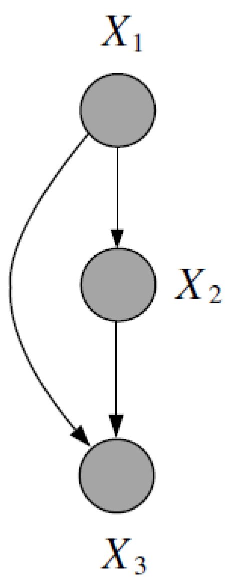
- We denote the augmented graphical model as:

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

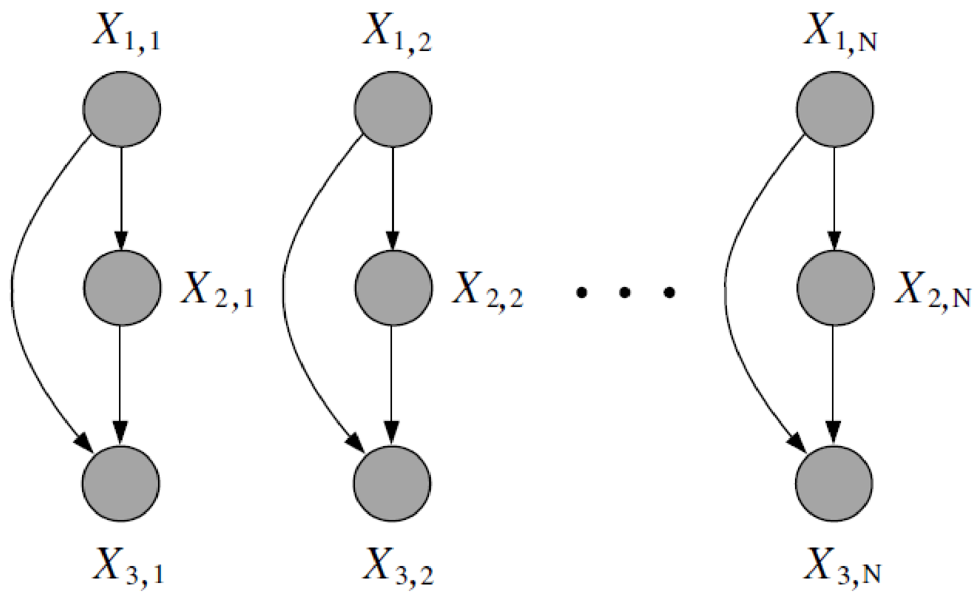
- Nodes $U^{(N)}$ are indexed using a pair of labels (u, n), where $u \in U$ designates a node in G and $n \in \{1, \dots, N\}$ designates the replication number.

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:

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

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

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

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

Maximum Log-Likelihood

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

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

- 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:

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

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

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

Maximum Log-Likelihood: Discrete Case

- Likelihood is given by Categorical distribution:

$$p(x_u | x_{\pi_u}, \lambda_u) = \mathrm{Cat}_{x_u|x_{\pi_u}}[\lambda_u]$$

$$= \prod_{c=1}^{C} \prod_{k=1}^{K} \lambda_{uck}^{x_{uck}} = \lambda_{uck}, \quad s.t. \quad \sum_k \lambda_{uck} = 1$$

- 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:

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

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

$$\Longrightarrow \underset{\lambda_{uc1},\ldots,\lambda_{ucK}}{\operatorname{argmax}} \sum_{n=1}^{N}\sum_{k=1}^{K}\log \lambda_{uck}^{x_{uck,n}}, \qquad s.t. \quad \sum_k \lambda_{uck} = 1$$

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

$$\Longrightarrow \underset{\lambda_{uc1},\ldots,\lambda_{ucK}}{\operatorname{argmax}} \sum_{k=1}^{K}\log \lambda_{uck}^{N_{uck}}, \qquad s.t. \quad \sum_k \lambda_{uck} = 1$$

Maximum Log-Likelihood: Discrete Case

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

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

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

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

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:

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

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

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

$$\underset{\lambda_u}{\operatorname{argmax}}\{\textstyle\sum_n \log p(x_{u,n}|x_{\pi_{u},n},\lambda_u) + \log p(\lambda_u)\},$$

- We get:

$$\underset{\lambda_{uc1},\ldots,\lambda_{ucK}}{\operatorname{argmax}} \left\{\textstyle\sum_n \log \mathrm{Cat}_{x_{u,n}|x_{\pi_{u},n}}[\lambda_u] + \log \mathrm{Dir}_{\lambda_{uc1\ldots ucK}}[\alpha_{uc1},\ldots\alpha_{ucK}]\right\}, \text{ s.t. } \textstyle\sum_k \lambda_{uck} = 1$$

Independent of λ_{uck}

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

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

$$= \underset{\lambda_{uc1},\ldots,\lambda_{ucK}}{\operatorname{argmax}} \left\{\textstyle\sum_k\left(\log \lambda_{uck}^{N_{uck}+\alpha_{uck}-1}\right)\right\}, \qquad \text{s.t. } \textstyle\sum_k \lambda_{uck} = 1$$

Maximum A Posteriori: Discrete Case

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

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

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

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

Maximum Log-Likelihood: Continuous Case

- Likelihood is given by linear-Gaussian model:

$$p\left(x_u \middle| x_{\pi_u}, \theta_{x_u|x_{\pi_u}}\right) = \mathrm{Norm}_{x_u|x_{\pi_u}}[w_{u0}, \dots, w_{uC}, \sigma_u^2]$$

$$= \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left\{-0.5 \frac{\left(x_u - \left(\sum_{c \in x_{\pi_u}} w_{uc} x_{uc} + w_{u0}\right)\right)^2}{\sigma_u^2}\right\}$$

- 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:

$$\underset{\theta_{x_u|x_{\pi_u}}}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(x_{u,n}|x_{\pi_u,n}, \theta_{x_u|x_{\pi_u}})$$

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

$$= \underset{\theta_{x_u|x_{\pi_u}}}{\operatorname{argmax}} \underbrace{\sum_{n=1}^{N}\left\{-\frac{1}{2}\log(2\pi\sigma_u^2) - \frac{1}{2\sigma_u^2}\left(x_{u,n} - \left(\sum_{c\in x_{\pi_u}} w_{uc}x_{uc,n} + w_{u0}\right)\right)^2\right\}}_{L}$$

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:

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

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

$$\vdots$$

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

- Which can be used to solve for the $C+1$ unknowns $w_{u0,}\, w_{u1}, \ldots\, w_{uC}$.

- 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:

$$p(w_{u0}, \dots w_{uC}, \sigma_u^2) = p(\sigma_u^2) p(w_{u0}, \dots w_{uC} | \sigma_u^2)$$

$$= p(\sigma_u^2) \prod_{c \in x_{\pi_u}} p(w_{uc} | \sigma_u^2) \qquad \text{(Naïve Bayes)}$$

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

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

- where μ_u is a hyperparameter.

Maximum A Posteriori: Continuous Case

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

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

- (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.

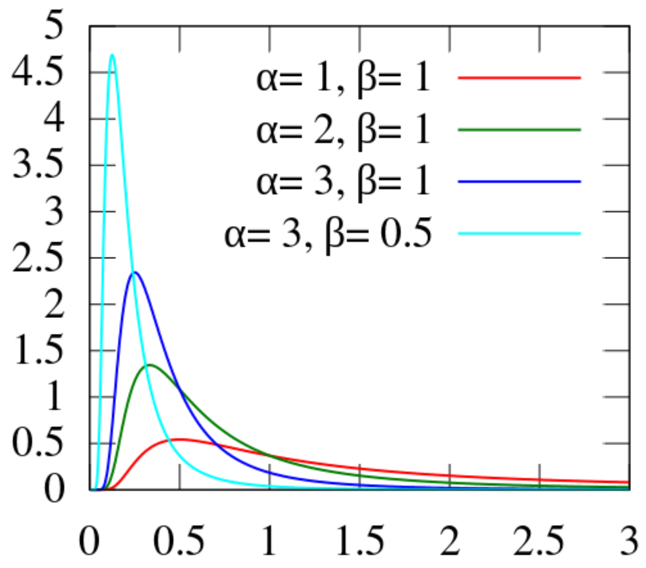


Image source: https://en.wikipedia.org/wiki/Inverse-gamma_distribution

Maximum A Posteriori: Continuous Case

- Putting the likelihood and conjugate prior into

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

- We get:

$$\underset{\theta_{x_u|x_{\pi_u}}}{\operatorname{argmax}} \{ \sum_n \log \mathrm{Norm}_{x_{u,n}|x_{\pi_u,n}}[w_{u0}, \ldots, w_{uC}, \sigma_u^2] + \underbrace{\log \mathrm{InvGam}_{\sigma_u^2}[\alpha_u, \beta_u] + \sum_c \log p(w_{uc}|\sigma_u^2)}_{L} \}$$

Maximum A Posteriori: Continuous Case

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

$$\left.\begin{aligned}\frac{\partial L}{\partial w_{u0}} &= 0\\ \frac{\partial L}{\partial w_{u1}} &= 0\\ &\vdots\\ \frac{\partial L}{\partial w_{uC}} &= 0\end{aligned}\right\}$$

$C+1$ 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:

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

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:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N} \sum_{i}^{N} \left[\phi_c(\mathbf{y}_i) - \frac{\partial}{\partial \boldsymbol{\theta}_c} \log Z(\boldsymbol{\theta}) \right]$$

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:

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

Proof:

$$\frac{\partial \log Z(\theta)}{\partial \theta_c} = \frac{1}{Z(\theta)} \frac{\partial Z(\theta)}{\partial \theta_c},$$ where $$Z(\theta) = \sum_y \exp\left(\sum_c \theta_c^T \phi_c(y)\right)$$

$$\Rightarrow \frac{\partial Z(\theta)}{\partial \theta_c} = \sum_y \exp\left(\sum_c \theta_c^T \phi_c(y)\right)\phi_c(y)$$

$$\Rightarrow \frac{\partial \log Z(\theta)}{\partial \theta_c} = \frac{1}{Z(\theta)} \sum_y \phi_c(y) \exp\left(\sum_c \theta_c^T \phi_c(y)\right)$$

$$= \sum_y \phi_c(y) \underbrace{\frac{1}{Z(\theta)} \exp\left(\sum_c \theta_c^T \phi_c(y)\right)}_{p(y|\theta)} = \sum_y \phi_c(y)\ p(y|\theta)$$

Parameter Learning: UGM (MRF)

- Hence the gradient of the log-likelihood is:

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

- Clamped term: y is fixed to its observed values.
- Unclamped/contrastive term: 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:

Requires sum over all states of y, which is intractable

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

- 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.

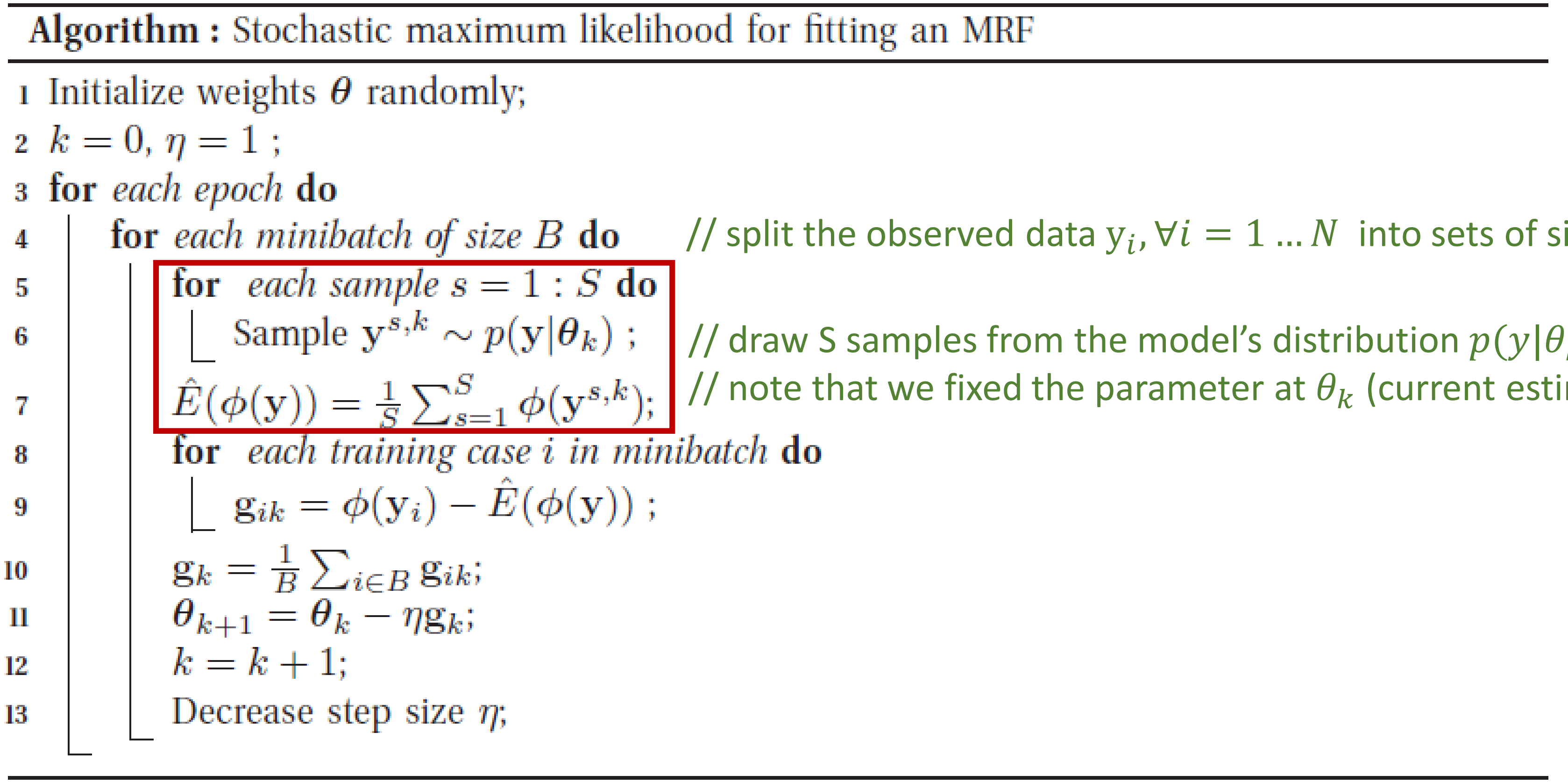
Method 1: Stochastic Maximum Likelihood

Algorithm : Stochastic maximum likelihood for fitting an MRF

```
Initialize weights $\boldsymbol{\theta}$ randomly;
$k = 0, \eta = 1$ ;
for each epoch do
    for each $\boxed{\textit{minibatch of size } B}$ do        // split the observed data $\mathrm{y}_i, \forall i = 1 \dots N$ into sets of size $B$
        for each sample $s = 1 : S$ do
            Sample $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\boldsymbol{\theta}_k)$ ;
        $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S}\sum_{s=1}^{S} \phi(\mathbf{y}^{s,k})$;
        for each training case $i$ in minibatch do
            $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$ ;
        $\mathbf{g}_k = \frac{1}{B}\sum_{i \in B} \mathbf{g}_{ik}$;
        $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \mathbf{g}_k$;
        $k = k + 1$;
        Decrease step size $\eta$;
```

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

Method 1: Stochastic Maximum Likelihood



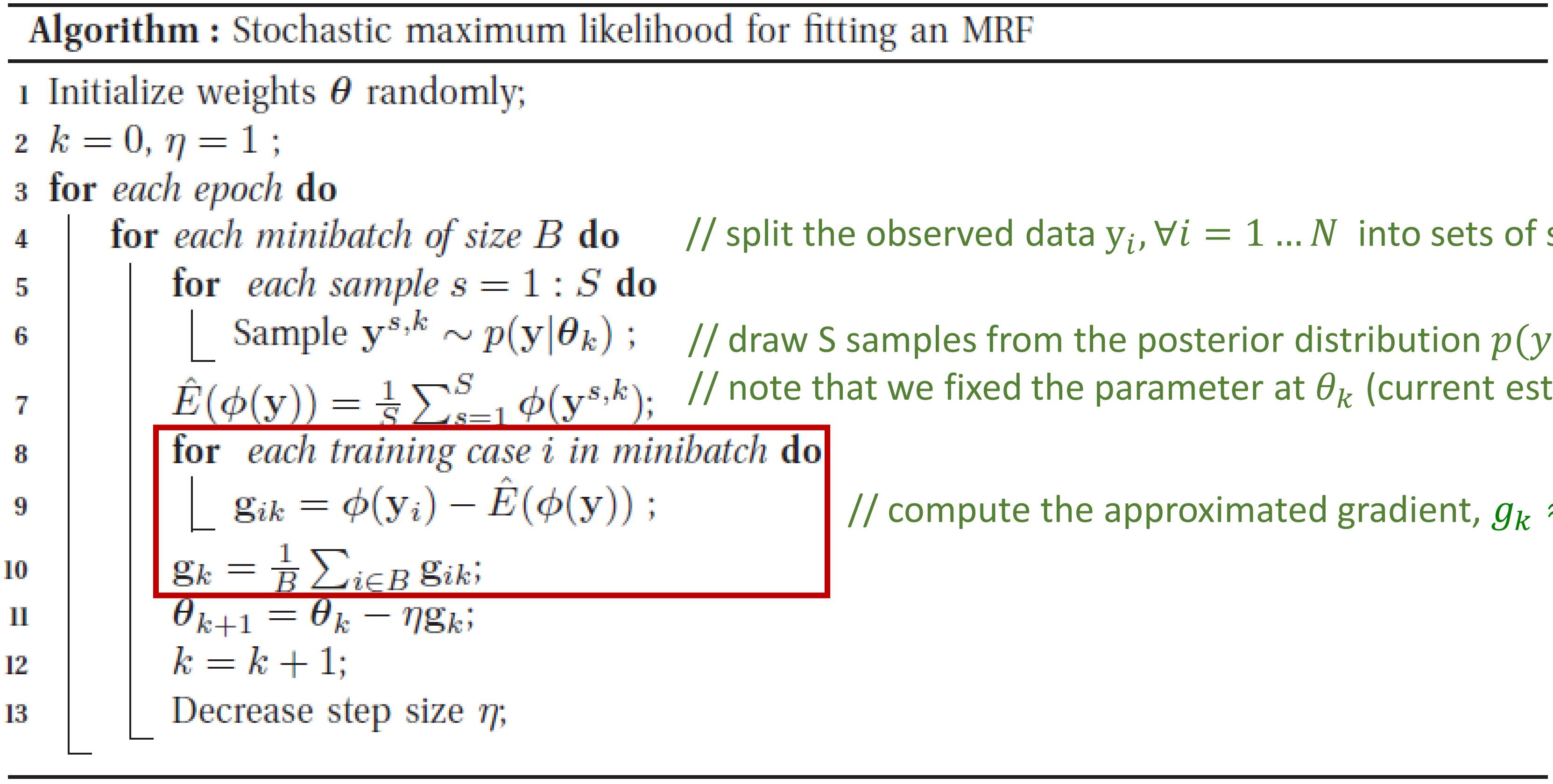
Algorithm : Stochastic maximum likelihood for fitting an MRF

```
Initialize weights $\theta$ randomly;
$k = 0, \eta = 1$ ;
for each epoch do
    for each minibatch of size $B$ do          // split the observed data $y_i, \forall i = 1 \dots N$ into sets of size $B$
        for each sample $s = 1 : S$ do
            Sample $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\theta_k)$ ;          // draw S samples from the model's distribution $p(y|\theta_k)$
        $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S}\sum_{s=1}^{S}\phi(\mathbf{y}^{s,k})$;          // note that we fixed the parameter at $\theta_k$ (current estimate)
        for each training case $i$ in minibatch do
            $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$ ;
        $\mathbf{g}_k = \frac{1}{B}\sum_{i \in B}\mathbf{g}_{ik}$;
        $\theta_{k+1} = \theta_k - \eta\mathbf{g}_k$;
        $k = k + 1$;
        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

```
Initialize weights $\theta$ randomly;
$k = 0, \eta = 1$ ;
for each epoch do
    for each minibatch of size $B$ do
        for each sample $s = 1 : S$ do
            Sample $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\theta_k)$ ;
        $\hat{E}(\phi(\mathbf{y})) = \frac{1}{S}\sum_{s=1}^{S} \phi(\mathbf{y}^{s,k})$;
        for each training case $i$ in minibatch do
            $\mathbf{g}_{ik} = \phi(\mathbf{y}_i) - \hat{E}(\phi(\mathbf{y}))$ ;
        $\mathbf{g}_k = \frac{1}{B}\sum_{i \in B} \mathbf{g}_{ik}$;
        $\theta_{k+1} = \theta_k - \eta \mathbf{g}_k$;
        $k = k + 1$;
        Decrease step size $\eta$;
```

// split the observed data $y_i, \forall i = 1 \dots N$ into sets of size B

// draw S samples from the posterior distribution $p(y|\theta_k)$

// note that we fixed the parameter at θ_k (current estimate)

// compute the approximated gradient, $g_k \approx \frac{\partial l}{\partial \theta}$

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

Method 1: Stochastic Maximum Likelihood

Algorithm : Stochastic maximum likelihood for fitting an MRF

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

$$\ell = \log \prod_i \frac{1}{Z} \prod_c \psi_c(y_{i,c})$$

$$= \sum_i \sum_c \log \psi_c(y_{i,c}) - N \log Z$$

$$= \sum_c \sum_{y_c} \underbrace{N(y_c)}_{} \log \psi_c(y_c) - N \log Z$$

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

$$\Rightarrow \frac{\partial \ell}{\partial \psi_c(y_c)} = \frac{N(y_c)}{\psi_c(y_c)} - \underbrace{N \frac{\partial}{\partial \psi_c(y_c)} \log Z}_{}$$

No closed-form!

Method 2: Iterative Proportional Fitting (IFP)

- Derivative of the log partition function:

$$\frac{\partial}{\partial\psi_c(y_c)}\log Z = \frac{1}{Z}\frac{\partial Z}{\partial\psi_c(y_c)}$$

$$= \frac{1}{Z}\frac{\partial}{\partial\psi_c(y_c)}\sum_y \prod_D \psi_D(y_D)$$

$$= \frac{1}{Z}\sum_y \delta(y, y_c)\frac{\partial}{\partial\psi_c(y_c)}\prod_D \psi_D(y_D)$$

$$= \frac{1}{Z}\sum_y \delta(y, y_c)\prod_{D\backslash c} \psi_D(y_D)$$

$$= \sum_y \delta(y, y_c)\frac{1}{\psi_c(y_c)}\boxed{\frac{1}{Z}\prod_D \psi_D(y_D)} \leftarrow p(y)$$

$$= \frac{1}{\psi_c(y_c)}\boxed{\sum_y \delta(y, y_c)p(y)} \leftarrow \sum_{y\backslash y_c} p(y)$$

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

Conditioned on all potentials since $p(y_c|\psi)$ is obtained from marginalization of the full distribution $p(y)$

Method 2: Iterative Proportional Fitting (IFP)

- 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:

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

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

Element-wise multiplication

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

Method 2: Iterative Proportional Fitting (IFP)

Algorithm : Iterative Proportional Fitting algorithm for tabular MRFs

```
Initialize $\psi_c = 1$ for $c = 1 : C$;
repeat
    for $c = 1 : C$ do
        $\boxed{p_c = p(\mathbf{y}_c | \psi);}$
        $\hat{p}_c = p_{\text{emp}}(\mathbf{y}_c)$;
        $\psi_c = \psi_c * \frac{\hat{p}_c}{p_c}$ ;
until converged;
```

// do marginalization with current ψ: $\sum_{y \backslash y_c} p(y)$

Source: Kevin Murphy, “Machine Learning: a probabilistic perspective”

Method 2: Iterative Proportional Fitting (IFP)

Algorithm : Iterative Proportional Fitting algorithm for tabular MRFs

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

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

Method 2: Iterative Proportional Fitting (IFP)

```
Algorithm : Iterative Proportional Fitting algorithm for tabular MRFs
Initialize $\psi_c = 1$ for $c = 1 : C$;
repeat
    for $c = 1 : C$ do
        $p_c = p(\mathbf{y}_c|\psi)$;                      // do marginalization with current $\psi$: $\sum_{y\backslash \mathrm{y}_c} p(y)$
        $\hat{p}_c = p_{\text{emp}}(\mathbf{y}_c)$;        // compute empirical probability of current clique $c$
        $\boxed{\psi_c = \psi_c * \frac{\hat{p}_c}{p_c}\ ;}$   // iterative proportional fitting
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:

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

- A Gaussian prior is often use:

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

- 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:

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

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

Parameter Learning: UGM (CRF)

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

$$\ell(\mathbf{w}) \triangleq \frac{1}{N}\sum_{i}^{N} \log p(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w})$$

$$= \frac{1}{N}\sum_{i}^{N}\left[\sum_{c} \mathbf{w}_c^T \phi_c(\mathbf{y}_i,\mathbf{x}_i) - \log Z(\mathbf{w},\mathbf{x}_i)\right]$$

- The gradient now becomes:

$$\frac{\partial \ell}{\partial \mathbf{w}_c} = \frac{1}{N}\sum_{i}^{N}\left[\phi_c(\mathbf{y}_i,\mathbf{x}_i) - \frac{\partial}{\partial \mathbf{w}_c}\log Z(\mathbf{w},\mathbf{x}_i)\right]$$

$$= \frac{1}{N}\sum_{i}^{N}\left[\phi_c\underbrace{(\mathbf{y}_i,\mathbf{x}_i)}_{} - \mathbb{E}\left[\phi_c(\mathbf{y},\mathbf{x}_i)\right]\right]$$

We need labeled pairs of data $\{y_i, x_i\}_{i=1}^{N}$ for learning!

Parameter Learning: UGM (CRF)

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

- The partition function depends on the inputs x_i!
- This means that we cannot bring $\mathbb{E}[\phi_c(\mathrm{y}, \mathrm{x}_i)]$ out of the summation.
- We now have to 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:  $w^* = \textsc{StochasticGradientDescent}(T, \eta)$
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,...,T do
10:     $(x^n, y^n) \in \mathcal{D}'$
11:     $d \leftarrow -\widetilde{\nabla}_w^{(x^n,y^n)} \mathcal{L}(w_{cur})$
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: $w^* = \textsc{StochasticGradientDescent}(T, \eta)$
2: **Input:**
3:     $T$                          // number of iterations
4:     $\eta_1, \ldots, \eta_T$     // 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,...,T **do**
10:    $\boxed{(x^n, y^n) \in \mathcal{D}'}$   // pick random subset of data (often 1–3 elements), i.e. $\mathcal{D}' \subset \mathcal{D}$
11:    $d \leftarrow -\widetilde{\nabla}_w^{(x^n,y^n)} \mathcal{L}(w_{cur})$
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:      $\eta_1, \ldots, \eta_T$     // 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,...,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:     $\boxed{d \leftarrow -\tilde{\nabla}_w^{(x^n, y^n)} \mathcal{L}(w_{cur})}$    // 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 $p(\mathrm{y}|\mathrm{x}_i, \mathrm{w})$.

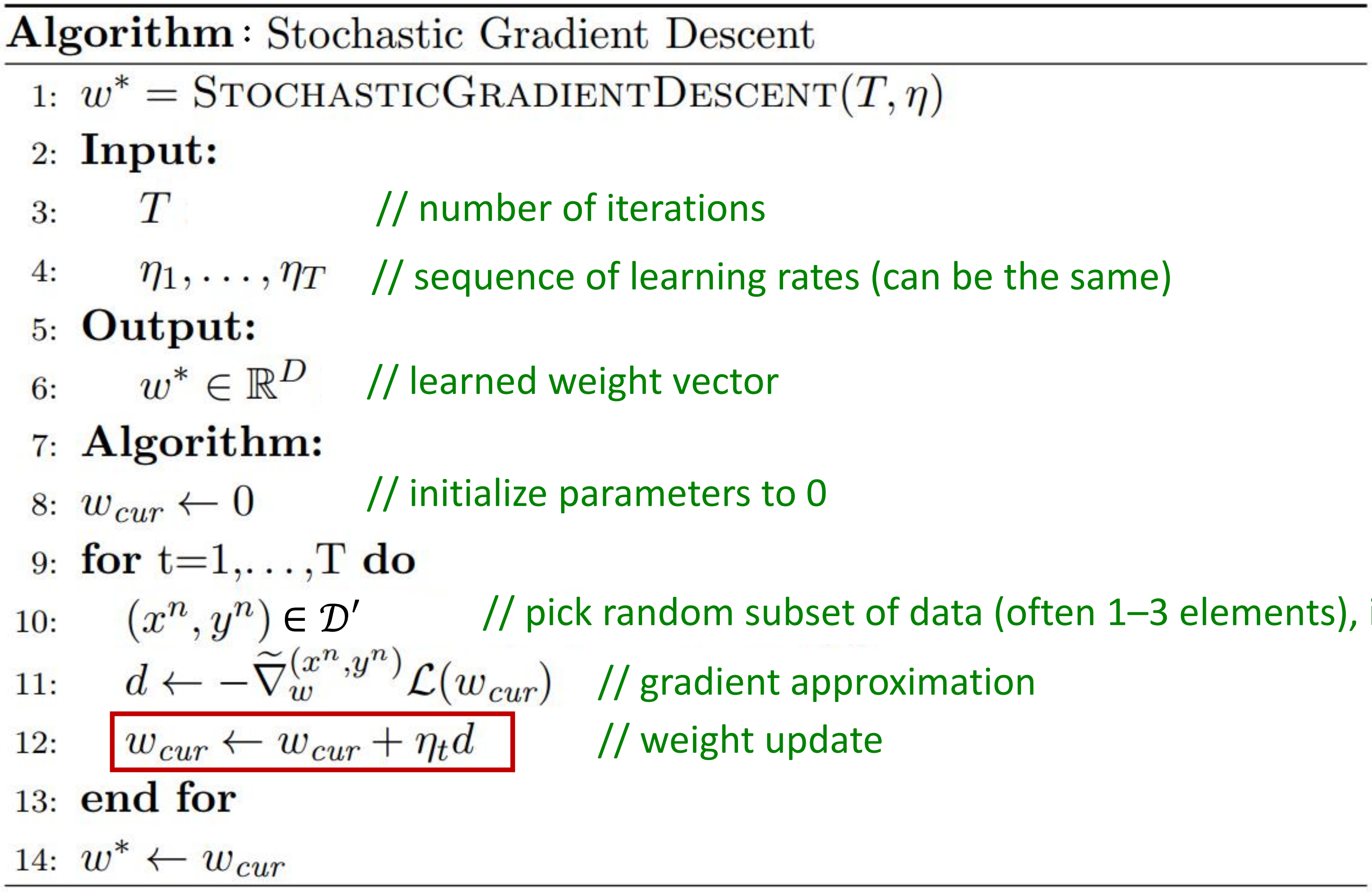
$$\nabla \mathcal{L}(w_c) = \frac{\partial \ell}{\partial \mathbf{w}_c} = \frac{1}{N} \sum_i \left[\phi_c(\mathbf{y}_i, \mathbf{x}_i) - \underbrace{\mathbb{E}\left[\phi_c(\mathbf{y}, \mathbf{x}_i)\right]}_{\mathbb{E}[\phi_c(\mathrm{y}, \mathrm{x}_i)] = \sum_{\mathrm{y}, \mathrm{x}_i} p(\mathrm{y}|\mathrm{x}_i, \mathrm{w}) \phi_c(\mathrm{y}, \mathrm{x}_i)} \right]$$

$$\Downarrow$$

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

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

Stochastic Gradient Descent



Algorithm: Stochastic Gradient Descent

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

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:

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

- 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.