STA414: Statistical Methods for Machine Learning II Lecture Notes

Yiliu Cao

May 3, 2024

Contents

1 Probabilistic models

1.1 An overview of probabilistic models

We have the random vector $X = (X_1, X_2, \cdots X_n)$, and we want to compute the relationship between each random variable. The joint distribution is $p(x) = p(x_1, \cdots, x_d)$. Denote the input data $\mathbf{x}$ (high-dimensional), and output y (discrete or continuous). In general, we have two models:

Regression:

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x,y)}{\int p(x,y)\, dy}$$

Classification/Clustering:

$$p(c|x) = \frac{p(x,c)}{\sum_c p(x,c)}$$

Observed vs Unobserved random variables

Supervised classification(learning):

- We **KNOW** what to predict
- **Supervised Dataset:** $\{x^{(i)}, c^{(i)}\}_{i=1}^N \sim p(x,c)$
- The class labels are observed.

Unsupervised classification(learning):

- We do **NOT KNOW** what to predict
- **Unsupervised Dataset:** $\{x^{(i)}\}_{i=1}^N \sim p(x) = \sum_c p(x,c)$
- We only observe the inputs $\mathbf{x}$

In order to estimate the unknown distribution $p(x)$, we have few assumptions:

1. **IID Data**: we assume the samples $x^{(i)}$ are independent and identically distributed.
2. **Parametrized distribution**: $p(x|\theta)$ comes from a parametrized family $\mathcal{P} = \{p(x|\theta) : \theta \in \Theta\}$

Maximum Likelihood Estimation(MLE)

MLE is the method to estimate the parameters of an assume probability distribution, given some observed data. Technically, we can use MLE to estimate any parameters we want. More specifically:

- Let $x^{(i)} \sim p_* = p(x|\theta_*)$ for $i = 1, \ldots, N$ be i.i.d. random variables.
- The joint of $\mathcal{D} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ is $p(\mathcal{D}|\theta_*) = \prod_i p(x^{(i)}|\theta_*)$.

- Assume we observe data $\mathcal{D}$ and θ_* is unknown. The likelihood function is:

$$\mathcal{L}(\theta; \mathcal{D}) = p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(x^{(i)}|\theta)$$

- The log-likelihood function:

$$\ell(\theta; \mathcal{D}) = \log \mathcal{L}(\theta; \mathcal{D}) = \sum_{i=1}^{N} \log p(x^{(i)}|\theta)$$

Here is an example of MLE

Sufficient Statistics and Exponential Families

A **sufficient statistics** is a function of the data that conveys exactly the same information about the parameter as the entire data.
In addition, we can writing any exponential family member in the form:

$$p(x|\eta) = h(x) \exp\{\eta^\top T(x) - A(\eta)\}$$

where

$$\begin{aligned} T(x) &: \text{sufficient statistics} \\ \eta &: \text{natural parameter} \\ A(\eta) &: \text{log-partition function} \\ h(x) &: \text{carrying measure} \end{aligned}$$

Moreover, let $X \sim p(x|\eta)$, then we have $E[T(X)] = A'(\eta)$
One example of exponential family

1.2 Statistical decision theory

Suppose we have an input vector x and the corresponding target, we want to predict the label given a new input. Notice that here we assume the output is the label/class which is discrete. However, the output can also be continuous (regression).

Intuitively, for a given new input x, we have:

$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)}$$

We then pick the $\mathcal{C}_k$ with the highest probability.
Decision Rule: Divide the input space to $\mathcal{R}_1$ & $\mathcal{R}_2$ such that all points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$. We want to make mistakes as less as possible; equivalently, we want to minimize the **misclassification rate**.
For $k \in \{1, 2\}$:

$$p(\text{mistake}) = p(x \in \mathcal{R}_2, \mathcal{C}_1) + p(x \in \mathcal{R}_2, \mathcal{C}_1) = \int_{\mathcal{R}_1} p(x, \mathcal{C}_2)\, dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1)\, dx \quad (1.1)$$

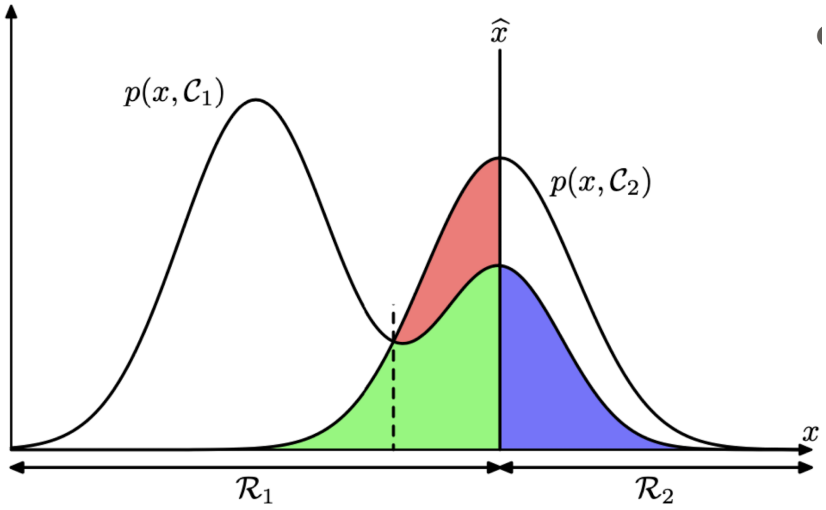


Figure 1.1: Misclassification Rate

1. **RedGreen regions**: inputs that belong to $\mathcal{C}_2$ but assigns to $\mathcal{R}_1$ as they are under $p\left(x, \mathcal{C}_2\right)$.
2. **Blue regions**: inputs that belong to $\mathcal{C}_1$ but assigns to $\mathcal{R}_2$ as they are under $p\left(x, \mathcal{C}_1\right)$
3. Therefore, for any data $\mathbf{x}$, if $p\left(x, \mathcal{C}_1\right) > p\left(x, \mathcal{C}_2\right)$, then we assign this point to $\mathcal{C}_1$, vice-versa. Therefore, $\mathcal{R} = \{x : \ p\left(x, \mathcal{C}_1\right) > p\left(x, \mathcal{C}_2\right)\}$

We want to minimize the misclassification error rate $\Rightarrow$ minimize the **loss**

Loss Function

Loss function measures the loss incurred by taking of any available decisions.

For discrete case

we denote L_{ij} as the $(k,\, j)$ element of the loss matrix. We want to minimize the expected loss
Therefore:

$$\begin{aligned}\mathbb{E}[L] &= \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p\left(x, \mathcal{C}_k\right) dx \\ &= \sum_j \int_{\mathcal{R}_j} \sum_k L_{kj} p\left(x, \mathcal{C}_k\right) dx\end{aligned}$$

Define $g_j(x) = \sum_k L_{kj} p\left(x, \mathcal{C}_k\right)$. Notice that $g_j(x) \geq 0$ and

$$\mathbb{E}[L] = \sum_j \int_{\mathcal{R}_j} g_j(x) dx$$

Thus, minimizing $\mathbb{E}[L]$ is equivalent to choosing

$$\mathcal{R}_j = \{x : g_j(x) < g_i(x) \text{ for all } i \neq j\} \tag{1.2}$$

$$\Rightarrow \mathcal{R}_j = \left\{ x : \sum_k L_{kj} p\left(\mathcal{C}_k \mid x\right) < \sum_k L_{ki} p\left(\mathcal{C}_k \mid x\right) \text{ for all } i \neq j \right\} \tag{1.3}$$

For regression

- Consider the input/target $(x,\ t)$, where t is continuous and the joint density is $p(x,\ t)$
- The regression function is $y(t)$
- The loss function is $L(y(x),\ t) = (y(x) - t)^2$

Therefore the expected loss will be:

$$\begin{aligned}
\mathbb{E}[L] &= \iint L(y(x), t) p(x, t) dx dt \\
&= \iint (y(x) - \mathbb{E}[t \mid x])^2 p(x, t) dx dt + \iint (\mathbb{E}[t \mid x] - t)^2 p(x, t) dx dt
\end{aligned}$$

Full derivations here
The second term is the conditional variance of $t|x$ and does not depend on $y(x)$ and hence the expected loss is minimized when $y(x) = \mathbb{E}[t|x]$. Therefore, we can see that the loss function will change the decision rule significantly; however, we can always reject the option or not making a decision.

2 Graphical Models

2.1 Introduction to graphical models

Remember our goal is to specify the joint distribution N random variables $p(x_1, \cdots, x_N) = p(x)$. If we assume each x_i is binary such that $x_i \in \{0,\ 1\}$, then we need $\mathbf{2^N - 1}$ parameters to specify $p(x)$. For example, $p(x_1 = 0,\ x_2 = 0,\ \cdots, x_N = 0)$ or $p(x_1 = 1,\ x_2 = 0,\ \cdots, x_N = 0)$.
Equivalently, we can specify the joint distribution $p(x)$ as:

$$\begin{aligned}
p\left(x_1, x_2, \ldots, x_N\right) &= \prod_{j=1}^{N} p\left(x_j \mid x_1, x_2, \ldots, x_{j-1}\right) \\
&= p(x_1|x_0) p(x_2|x_1,\ x_0) \cdots
\end{aligned}$$

Thus total number of parameters is $1 + 2 + 4 + \cdots 2^{N-1} = 2^N - 1$
We can see that it requires a huge number of parameters to specify the joint distribution. We want to draw relationships between variables.

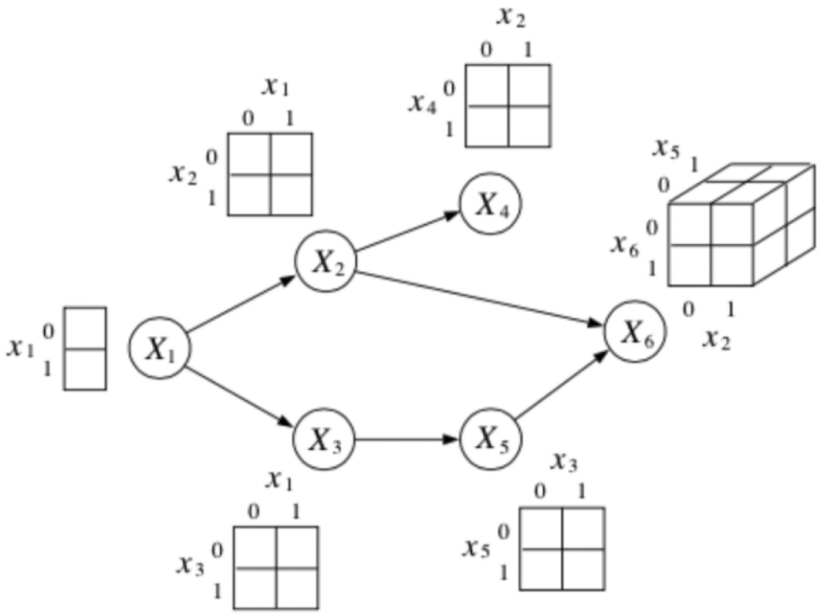


Figure 2.1: An example of conditional probability tables(CPT)

Condition independence

For three random variables x_A, x_B x_C, if x_A, x_B are conditionally independent given x_C, then we write $x_A \perp x_B \mid x_C$. The following conditions are equivalent:

- $x_A \perp x_B \mid x_C$
- $p(x_A,\, x_B|x_C) = p(x_A|x_C)p(x_B|x_C)$
- $p(x_A|x_B,\, x_C) = p(x_A|x_C)$
- $p(x_B|x_A,\, x_C) = p(x_B|x_C)$

2.2 Directed Acyclic Graphical Models

A directed cyclic graphical model encode a particular form of factorization of the joint distribution. The form of factorization is various.

Figure 2 shows an example of conditional probability. From the graph, we only need $2^1 * 4 + 2^0 + 2^2 = 13 < 2^6 - 1$ parameters.

D-separation: If C d-separates A and B, then $x_A \perp x_B \mid x_C \, \forall a \in A,\, b \in B$

Bayes ball algorithm

Bayes ball determines the conditional independence/dependence in a DAG (I personally found this part most ambiguous). There are three fundamental Bayes ball algorithms which are causal chain, common cause and explaining away. For each one, we will under it intuitively by drawing a story.

1. **Causal chain**

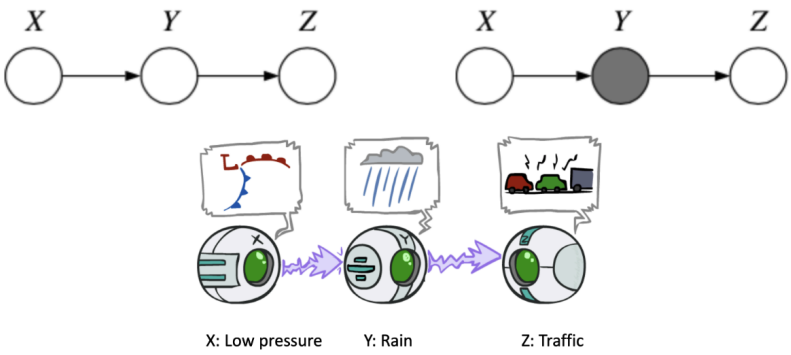


Figure 2.2: An illustration of causal chain

$$
\begin{aligned}
p(z \mid x, y) &= \frac{p(x, y, z)}{p(x, y)} \\
&= \frac{p(x)p(y \mid x)p(z \mid y)}{p(x)p(y \mid x)} \\
&= p(z \mid y) \\
\Rightarrow X \text{ and } Z &\text{ d-separated given } Y
\end{aligned}
$$

2. **Common cause**

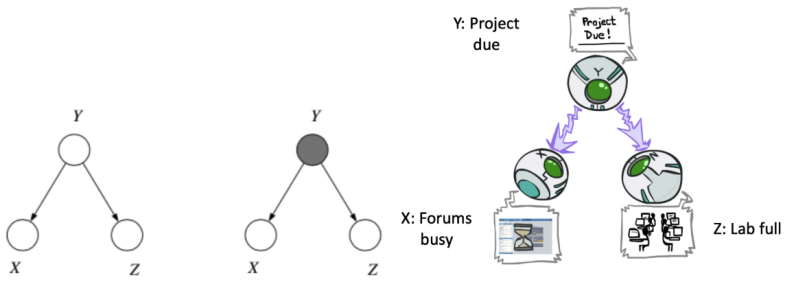


Figure 2.3: An illustration of common cause

$$
\begin{aligned}
p(x, z \mid y) &= \frac{p(x, y, z)}{p(y)} \\
&= \frac{p(y)p(x \mid y)p(z \mid y)}{p(y)} \\
&= p(x \mid y)p(z \mid y) \\
\Rightarrow X \text{ and } Z &\text{ d-separated given } Y
\end{aligned}
$$

3. **Explaining away**

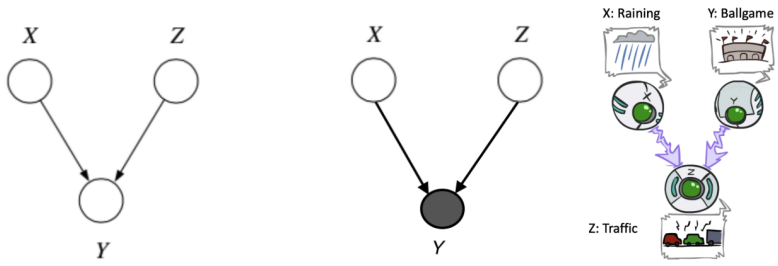


Figure 2.4: An illustration of explaining away

$$
\begin{aligned}
p(z \mid x, y) &= \frac{p(x)p(z)p(y \mid x, z)}{p(x)p(y \mid x)} \\
&= \frac{p(z)p(y \mid x, z)}{p(y \mid x)} \neq p(z \mid y) \\
&\Rightarrow X \text{ and } Z \text{ are NOT d-separated given } Y
\end{aligned}
$$

In general, the Bayes ball works as follows:

1. Shade all nodes x_C (these are observed)
2. Place "balls" at each node in x_A (or x_B)
3. Let the "balls" "bounce" around according to some rules. If any of the balls reach any of the nodes in x_B from x_A then $x_A \not\perp x_B \mid x_C$. Otherwise $x_A \perp x_B \mid x_C$

Example
Question: Is $x_2 \perp x_3 \mid \{x_1,\ x_6\}$

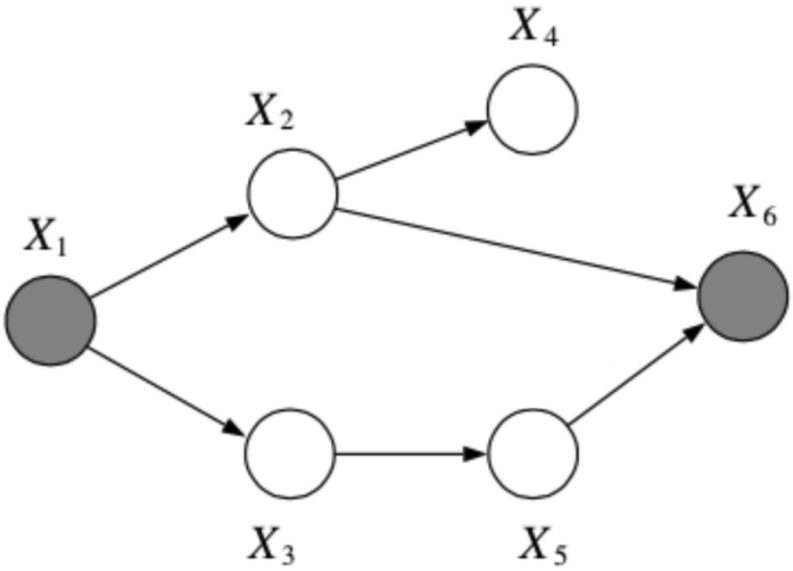


Figure 2.5: An illustration of explaining away

Answer: No. By Bayes ball algorithm, x_2 can travel to x_5, and hence can travel to x_3.

Moralization

Like I said, I personally do not like Bayes algorithm. Instead, another one called "moralization" is more straightforward and easier to use. We can follow the procedure:

1. **Draw the ancestral graph**
   We only keep the ancestor of the mentioned nodes. That said, in the previous example, we only keep the ancestors of $\{x_2, x_3, x_1 x_6\}$. Hence we have the entire graph except the node x_4. Note the ancestors includes **their parents, parents' parents etc**.

2. **"Moralize" the ancestral graph by "marrying" the parents**
   If two nodes have the same children, such as x_2 and x_5, then we draw a line between these two nodes.

3. **"Disorient" the graph**
   Ignore the directions by replacing the arrows to edges.

4. **Delete the givens and their edges**
   In the previous example, the givens are the x_1 and x_4.

5. **Find the answer**
   After we finished the step 1 to 4, we then justify whether the two nodes are connected or disconnected. If **connected**, then the two nodes are conditionally **dependent**. Otherwise **disconnected**, the two nodes are conditionally **independent**. In the previous example, we can easily find that x_2 and x_3 are d-separated by x_1 and x_6.

Question: What about the marginal independence, such as $x_2 \perp x_3$?
Answer: We use the same way as above without step 4.

2.3 Undirected Graphical Models

The undirected graphical models are also called the **Markov random fields (MRFs)**. Compare to graphical models, we have no more directed edges; instead, the dependencies are now described as undirected graphs. Moreover, **Markov blanket** is the set of nodes that makes X_i conditionally independent of all other nodes. **Clique** is a subset of nodes that every two nodes are connected by an edge. **Maximal clique** a clique that can not be extended by including one more adjacent vertex.

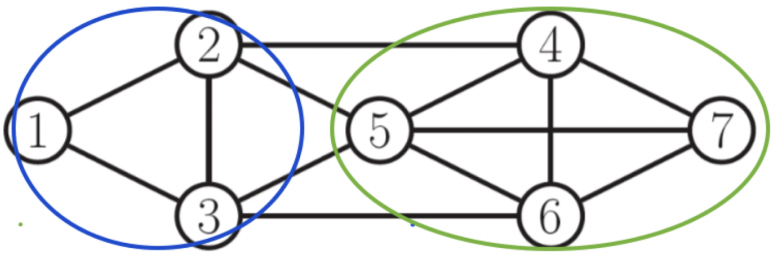


Figure 2.6: An example of Markov random fields: {1, 2, 3} is a clique and {4, 5, 6, 7} is a maximal clique

Distribution induced by MRFs

- Let $X = (X_1, \ldots, X_m)$ be the set of all random variables in our graph G.
- Let $\mathcal{C}$ be the set of all maximal cliques of G.
- The distribution p of X factorizes with respect to G if

$$p(x) \propto \prod_{C \in \mathcal{C}} \psi_{\mathcal{C}}(x_C)$$

for some nonnegative potential functions ψ_C, where $x_C = (x_i)_{i \in C}$.

The density can be factorized to cliques is also called the **Hammersley-Clifford Theorem**.
Global markov properties: $X_A \perp X_B \mid X_S$ if the sets A and B are separated by S in G (every path from A to B has to pass S). Therefore, the joint distribution described on above is:

$$\begin{aligned} p(x) \propto & \psi_{1,2,3}(x_1, x_2, x_3)\, \psi_{2,3,5}(x_2, x_3, x_5)\, \psi_{2,4,5}(x_2, x_4, x_5) \\ & \times \psi_{3,5,6}(x_3, x_5, x_6)\, \psi_{4,5,6,7}(x_4, x_5, x_6, x_7) \end{aligned}$$

Not all DAGMs can be represented as MRFs. .

Relations between exponential families and MRFs

- Consider a parametric family of factorized distributions

$$p(x \mid \theta) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{C}} \psi_c(x_C \mid \theta_C), \quad \theta = (\theta_C)_{c \in \mathcal{C}}.$$

- We can write this in an exponential form:

$$p(x \mid \theta) = \exp\{\sum_{C \in \mathcal{C}} \log \psi_C(x_C \mid \theta_C) - \underbrace{\log Z(\theta)}_{=A(\theta)}\}$$

- Suppose the potentials have a log-linear form

$$\log \psi_C(x_C \mid \theta_C) = \theta_C^\top \phi_C(x_C)$$

we then get the exponential family

$$p(x \mid \theta) = \exp\{\sum_{C \in \mathcal{C}} \theta_C^\top \phi_C(x_C) - \underbrace{\log Z(\theta)}_{=A(\theta)}\}$$

Question: When the potentials have a log-linear form?
Solution: Suppose we have the random vector x_1 and x_2, and they are all binary. Then

there are four possible values of $(x_1,\ x_2)$: (0, 0), (1, 0), (0, 1), (1, 1). We take

$$\theta_{1,2} := \begin{bmatrix} \log \psi_{1,2}(0,0) \\ \log \psi_{1,2}(0,1) \\ \log \psi_{1,2}(1,0) \\ \log \psi_{1,2}(1,1) \end{bmatrix} \in \mathbb{R}^4$$

and let $\psi_{1,2}\left(x_1, x_2\right)$ be the function that satisfies

$$\phi_{1,2}(0,0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \phi_{1,2}(0,1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \phi_{1,2}(1,0) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \phi_{1,2}(1,1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

$$\Rightarrow \phi_{1,2}\left(x_1, x_2\right) = \begin{bmatrix} \left(1-x_1\right)\left(1-x_2\right) \\ \left(1-x_1\right) x_2 \\ x_1\left(1-x_2\right) \\ x_1 x_2 \end{bmatrix}$$

We then have a linear form:

$$\log \psi_C\left(x_{12} \mid \theta_{12}\right) = \theta_{12}^{\top} \phi_{12}\left(x_{12}\right)$$

Ising model

The Ising model is an example of MRFs, which is used to model magnets. It has a form of potential function:

$$\psi_{st}(x_s,\ x_t) = e^{J_{st} x_s x_t}$$

Equivalently:

$$\psi_{st}(-1,\ -1) = \psi_{st}(1,\ 1) = e^{J_{st}}$$

$$\psi_{st}(1,\ -1) = \psi_{st}(-1,\ 1) = e^{-J_{st}}$$

$$\psi_{st}(x_s,\ x_t) = 0 \text{ if the two nodes are not connected}$$

In terms of the distribution in Ising model, we also incldue the node potential $\psi_s(x_s) = e^{b_s x_s}$:

$$p(x) \propto \prod_{s \sim t} \psi_{st}\left(x_s, x_s\right) \prod_s \psi_s\left(x_s\right) = \exp\left\{ \sum_{s \sim t} J_{st} x_s x_t + \sum_s b_s x_s \right\}$$

Recall: Multivariate normal distribution
$X = (X_1, \ldots, X_m) : \mu \in \mathbb{R}^m$ and Σ symmetric positive definite $m \times m$ matrix. Write $X \sim N_m(\mu, \Sigma)$ if the density of the vector X is

$$f(\boldsymbol{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{m/2}} (\det \Sigma)^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \mu)^T \Sigma^{-1} (\boldsymbol{x} - \mu) \right).$$

Denote $K = \Sigma^{-1}$ then

$$f(\boldsymbol{x}; \mu, \Sigma) \propto \prod_{\boldsymbol{s}} e^{-\frac{1}{2} K_{ss} (x_s - \mu_s)^2} \prod_{\boldsymbol{s} < t} e^{-K_{st} (x_s - \mu_s)(x_t - \mu_t)}$$

Intuitively, we can also visualize the conditional independencies between each variables (similar to Bayes ball). Just like the Ising model, here we can use concentration matrix K to represent the relationship between variables.
Conditional independence:

- $X_i \perp X_j$ if and only if $\Sigma_{ij} = 0$.
- $X_i \perp X_j \mid X_C$ if and only if $\Sigma_{ij} - \Sigma_{i,C}\Sigma_{C,C}^{-1}\Sigma_{C,j} = 0$
- Let $R = V \backslash \{i, j\}$. The following are equivalent:
  - $X_i \perp X_j \mid X_R$
  - $\Sigma_{ij} - \Sigma_{i,R}\Sigma_{R,R}^{-1}\Sigma_{R,j} = 0$
  - $(\Sigma^{-1})_{ij} = 0$

Hence we have the **Gaussian Graphical models**

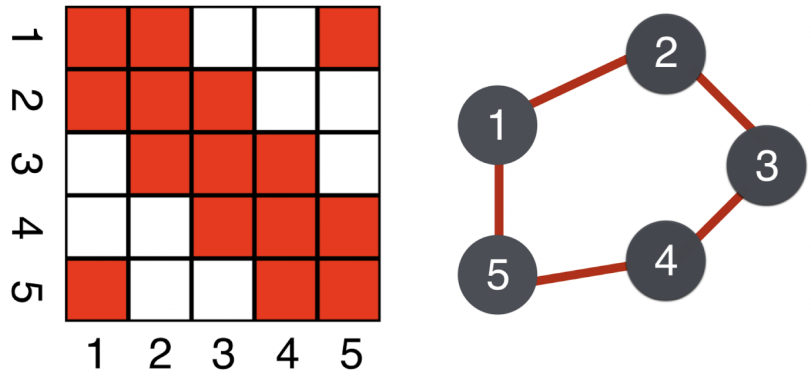


Figure 2.7: An example of Gaussian Graphical model

$K_{ij} = 0$ if and only if $X_i \perp X_j \mid X_{rest}$. For example, $X_1 \perp X_4 \mid X_{rest}$

3 Exact Inference

3.1 Introduction to exact inference

We want to explore the inference from the probabilistic graphical models. In notations, we have

- x_E represent the observed evidence
- x_F represent the unobserved varaible we want to infer
- $x_R = x\{x_E,\ x_F\}$ represent the remaining variables

For the conditional probability $p(x_F|x_E)$, by Bayes theorem:

$$p\left(x_F \mid x_E\right) = \frac{p\left(x_F, x_E\right)}{p\left(x_E\right)} = \frac{p\left(x_F, x_E\right)}{\sum_{x_F} p\left(x_F, x_E\right)}$$

Moreover, we can also marginalize the extraneous variables to have the marginal joint distribution:

$$p\left(x_F, x_E\right) = \sum_{x_R} p\left(x_F, x_E, x_R\right)$$

The below equation give us an intuition of exact inference, which is to marginalize other variables. However, when the number of variables increases, the order we marginalize will affect the computational cost. Hence we have to choose an appropriate **elimination order**. More specifically, we want to have the exact inference for one variable in DAGMs or MRFs.

Example 3.1.1. Suppose we have the simple chain for four variables A, B, C, D

$$A \to B \to C \to D$$

where:

$$x_F = \{D\},\ x_E = \{A,\ B,\ C\},\ x_R = \{\}$$

We want to compute the exact inference of D, $p(D)$ In the simple chain settings, we can express the joint distribution as:

$$\begin{aligned} p(D) &= \sum_{A,B,C} p(A,B,C,D) \\ &= \sum_C \sum_B \sum_A p(A)p(B \mid A)p(C \mid B)p(D \mid C) \end{aligned}$$

If we choose an elimination order:

$$\begin{aligned} p(D) &= \sum_{A,B,C} p(A,B,C,D) \\ &= \sum_C p(D \mid C) \left(\sum_B p(C \mid B) \left(\sum_A p(A)p(B \mid A) \right) \right) \\ &= \sum_C p(D \mid C) \sum_B p(C \mid B) \sum_A p(A)p(B \mid A) \\ &= \sum_C p(D \mid C) \sum_B p(C \mid B)p(B) \\ &= \sum_C p(D \mid C)p(C) \\ &= \sum_C p(D,\ C) \end{aligned}$$

The above example give us an intuition that we can firstly marginalize the node with no children (C first). Equivalently, we start the nodes that comes early in the induced ordering of the DAG.

3.2 Sum-product algorithm

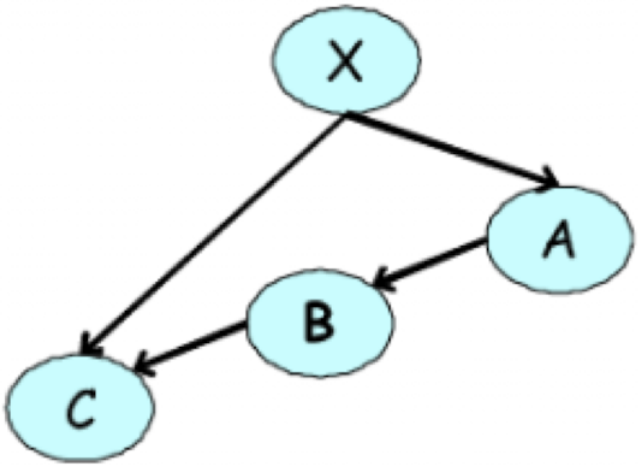


Figure 3.1: An example of Sum-product inference

Example 3.2.1.
To express the joint distribution, we introduce the nonnegative factors ψ:

$$\begin{aligned} p(A,B,C) &= \sum_X p(A,\ B,\ C,\ X) \\ &= \sum_X p(X)p(A \mid X)p(B \mid A)p(C \mid B,X) \\ &= \sum_X \psi_1(X)\psi_2(A,X)\psi_3(A,B)\psi_4(X,B,C) \\ &= \psi_3(A,B)\sum_X \psi_1(X)\psi_2(A,X)\psi_4(X,B,C) \\ &= \psi_3(A,B)\tau(A,B,C) \end{aligned}$$

We eventually end with a new factor τ and it is an example of how sum-product algorithm works.

In general, we compute the the conditional distribution $p(x_F \mid x_E)$ using the **sum-product algorithm**. Denote $\mathcal{F}$ as the set of potentials or factors.

$$p\left(x_F \mid x_E\right) \propto \tau\left(x_F, x_E\right) = \sum_{x_R} \prod_{C \in \mathcal{F}} \psi_C\left(x_C\right)$$

For DAGMs, $\mathcal{F}$ is $\{i\} \cup$ parents(i), $\forall$ nodes i
For MRFs, $\mathcal{F}$ is all maximal cliques.

We will see more about exact inference on message passing

4 Message passing

4.1 TrueSkill latent variable models

Latent variables are variables that can only be inferred indirectly through a mathematical model from other observable variables that can be directly observed or measured.

TrueSkill model is a player ranking system for competitive games. Our goal is to infer the skills of players in a competitive game, based on the results of the games. Hence, each player's skill is a **latent variable**, and we assume they are fixed. Moreover, we also assume each player's skill is independent such as independent Gaussian prior.

For each game, the probability that player i beats player j is

$$p(i \text{ beats } j) = \sigma(z_i - z_j), \text{ where}$$

$$\sigma(y) = \frac{1}{1 + \exp(-y)}$$

Hence the entire joint likelihood of players and games is:

$$\begin{aligned} & p\left(z_1, z_2, \ldots z_N, \text{ game 1, game 2, .. game } T\right) \\ &= \underbrace{\left[\prod_{i=1}^{N} p\left(z_i\right)\right]}_{\text{prior}} \underbrace{\left[\prod_{\text{games}} p\left(\text{i beats j} \mid z_i, z_j\right)\right]}_{\text{likelihood}} \end{aligned}$$

However, it is hard to compute the posterior

$$\begin{aligned} & p\left(z_1, z_2 \mid \text{ game 1, game 2, } \ldots \text{ game } T\right) \\ &= \int \cdots \int p\left(z_1, z_2, z_3 \ldots z_N \mid x\right) dz_3 \ldots dz_N \end{aligned}$$

Message passing can help us to compute the posterior.

4.2 Message Passing

From the last section, we can see that determining an optimal elimination order is hard, and the resulting marginalization can be still costly. Therefore, we introduce the tress as any elimination ordering from leaves to the root is optimal.

Inference in trees

A graph is $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is vertices and $\mathcal{E}$ is edges. A leaf in a tree is the nodes where it has only one neighbor. We can regard a tress as MRFs, with joint distribution:

$$p\left(x_1, x_2, \ldots, x_n\right) = \underbrace{\frac{1}{Z}}_{\text{normalization}} \overbrace{\prod_{i \in \mathcal{V}} \psi\left(x_i\right)}^{\text{nodes}} \underbrace{\prod_{(i,j) \in \mathcal{E}} \psi_{ij}\left(x_i, x_j\right)}_{\text{edges}}$$

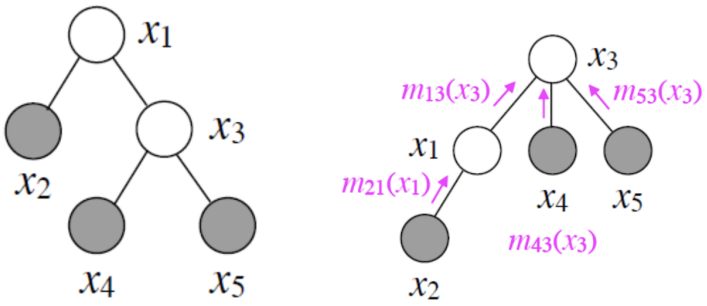


Figure 4.1: An example of tree and message passing

Example 4.2.1. Above shows a tree where $x_E = \{\bar{x_2}, \bar{x_4}, \bar{x_5}\}$, $x_R = x_1$. We want to compute $p(x_3 \mid x_E)$:

$$\begin{aligned} p(x_3|x_E) &= \frac{p\left(x_3, x_E\right)}{\sum_{x_3'} p\left(x_3', x_E\right)} \\ &= \frac{1}{Z^E}\, p(x_3, x_E) \end{aligned}$$

Express $p(x_3, x_E)$, we have

$$\begin{aligned} &= \sum_{x_1} \psi_1\left(x_1\right) \psi_2\left(\bar{x}_2\right) \psi_3\left(x_3\right) \psi_4\left(\bar{x}_4\right) \psi_5\left(\bar{x}_5\right) \psi_{12}\left(x_1, \bar{x}_2\right) \psi_{13}\left(x_1, x_3\right) \psi_{34}\left(x_3, \bar{x}_4\right) \psi_{35}\left(x_3, \bar{x}_5\right) \\ &= \frac{1}{Z^E} \underbrace{\psi_4\left(\bar{x}_4\right) \psi_{34}\left(x_3, \bar{x}_4\right)}_{m_{43}(x_3)} \underbrace{\psi_5\left(\bar{x}_5\right) \psi_{35}\left(x_3, \bar{x}_5\right)}_{m_{53}(x_3)} \psi_3\left(x_3\right) \sum_{x_1} \psi_1\left(x_1\right) \psi_{13}\left(x_1, x_3\right) \underbrace{\psi_2\left(\bar{x}_2\right) \psi_{12}\left(x_1, \bar{x}_2\right)}_{m_{21}(x_1)} \\ &= \frac{1}{Z^E} m_{43}\left(x_3\right) m_{53}\left(x_3\right) \psi_3\left(x_3\right) \underbrace{\sum_{x_1} \psi_1\left(x_1\right) \psi_{13}\left(x_1, x_3\right) m_{21}\left(x_1\right)}_{m_{13}(x_3)} \\ &= \frac{1}{Z^E} \psi_3\left(x_3\right) m_{43}\left(x_3\right) m_{53}\left(x_3\right) m_{13}\left(x_3\right) \\ &= \frac{\psi_3\left(x_3\right) m_{43}\left(x_3\right) m_{53}\left(x_3\right) m_{13}\left(x_3\right)}{\sum_{x_3'} \psi_3\left(x_3'\right) m_{43}\left(x_3'\right) m_{53}\left(x_3'\right) m_{13}\left(x_3'\right)} \end{aligned}$$

Sum-product algorithm

The above example shows how we can pass the message and eventually compute the conditional distribution. For each passed message such as $m_{43}(x_3)$, it uses the sum-product algorithm we introduced in the chapter of exact inference. To summarize, we have:

- If x_j **unobserved**, the message sent from variable j to $i \in N(j)$ is

$$m_{j \to i}\left(x_i\right) = \sum_{x_j} \psi_j\left(x_j\right) \psi_{ij}\left(x_i, x_j\right) \prod_{k \in N(j) \backslash i} m_{k \to j}\left(x_j\right)$$

- If x_j is **observed**, the message is

$$m_{j \to i}\left(x_i\right) = \psi_j\left(\bar{x}_j\right) \psi_{ij}\left(x_i, \bar{x}_j\right) \prod_{k \in N(j) \backslash i} m_{k \to j}\left(\bar{x}_j\right)$$

- Once the message passing stage is complete, we can compute our beliefs as

$$b(x_i) = p(x_i \mid x_E) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \to i}(x_i).$$

4.3 Belief Propagation on Trees

The above algorithm shows how the message is passed from the leaves to the root. In a formal setting, we have four steps:

1. Choose the root r arbitrarily.
2. Pass the message from leaves to r.
3. **Pass the message from the root to the leafs**. Note that here we only pass the message from the root to the **unobserved** nodes.
4. Compute the beliefs/marginals for all unobserved nodes.

Example 4.3.1. We can still infer the previous example (image), and shows the complete belief propagation process. From Figure 4.1, our goal is to compute the beliefs

$$p(x_3|\bar{x_2}, \bar{x_4}, \bar{x_5})$$

and

$$p(x_1|\bar{x_2}, \bar{x_4}, \bar{x_5})$$

Step 1: Choose x_3 to be our root.
Step 2: Pass the message from leaves to x_3.

$$\begin{aligned}
m_{5 \to 3}(x_3) &= \psi_5(\bar{x}_5)\, \psi_{35}(x_3, \bar{x}_5) \\
m_{2 \to 1}(x_1) &= \psi_2(\bar{x}_2)\, \psi_{12}(x_1, \bar{x}_2) \\
m_{4 \to 3}(x_3) &= \psi_4(\bar{x}_4)\, \psi_{34}(x_3, \bar{x}_4) \\
m_{1 \to 3}(x_3) &= \sum_{x_1} \psi_1(x_1)\, \psi_{13}(x_1, x_3)\, m_{2 \to 1}(x_1)
\end{aligned}$$

Step 3: Pass the message from the root to other unobserved nodes:

$$m_{3 \to 1}(x_1) = \sum_{x_3} \psi_3(x_3)\, \psi_{13}(x_1, x_3)\, m_{4 \to 3}(x_3)\, m_{5 \to 3}(x_3)$$

Step 4: Calculate the beliefs for x_1 and x_3.

$$\begin{aligned}
b(x_1) &\propto \psi_1(x_1)\, m_{2 \to 1}(x_1)\, m_{3 \to 1}(x_1) \\
b(x_3) &\propto \psi_3(x_3)\, m_{1 \to 3}(x_3)\, m_{4 \to 3}(x_3)\, m_{5 \to 3}(x_3)
\end{aligned}$$

Loopy Belief Propagation

The inference we just introduced has limits that the graph (MRF) must be a tree. However, in most cases such as TrueSkill models, we do not have such prerequisites. What we do instead is to keep passing the message until convergence. This is called the Loopy Belief Propagation. It works as follows:

- Initialize all messages uniformly:

$$m_{i\to j}(x_j) = (1/k, \ldots, 1/k)$$

  where k is the number of states x_j can take.

- Keep running BP updates until **the messages** "converges":

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\,\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

  and (sometimes) normalized for stability.

- It will generally not converge, but often works fine.

- Compute beliefs $b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$.

Max-product belief propagation

The main difference between the sum-product and max-product belief propagation is that we are now **maximizing** over x_j instead of summarizing them.

$$m_{j\to i}(x_i) = \max_{x_j} \psi_j(x_j)\,\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

Now the beliefs are called max-marginals:

$$\hat{b}(x_i) = \max_{x_i} p\left(x_i, x_{\setminus_i}\right) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i).$$

MAP inference: take $\hat{x}_i := \arg\max_{x_i} \hat{b}(x_i)$ for all $i \notin E$.
Here is a numerical example of message passing.

5 Monte Carlo Methods

5.1 Introduction to Monte Carlo Methods

In formal definitions, **Monte Carlo methods** is a **collection of methods** involving the use of simulated random numbers to estimate some functions of a probability distribution. Simply put, we use Monte Carlo methods to draw samples from an unknown distribution $p(x)$. After obtaining the samples, we can then estimate $p(x)$ via different methods such as moments, KDE, etc.

A sample from a distribution $p(x)$ is a single realization x whose probability distribution is $p(x)$. However, we assume that the density form $p(x)$ can be evaluated within a multivariate constant Z. That said:

$$\begin{aligned} p(x) &= \frac{\tilde{p}(x)}{Z} \\ &\propto \tilde{p}(x) \end{aligned}$$

Here, $p(x)$ is very hard to evaluate due to the multivariate constant Z. Hence our primary goal is $p(\tilde{x})$.

Main objectives of sampling

Using the Monte Carlo methods, we aim to solve one or both of the following problems:

1. Generate the samples $\{x^{(i)}\}_{r=1}^{R}$ from the unkown distribution $p(x)$.
2. To estimate the expectation of functions, $\psi(x)$, under the distribution $p(x)$

$$\Phi = \mathop{\mathbb{E}}_{x\sim p(x)}[\phi(x)] = \int \phi(x)p(x)dx$$

***Note**: The notation $\mathop{\mathbb{E}}_{x\sim p(x)}[\phi(x)]$ rerpresent the expected of value of the test function $\psi(x)$ and x are the samples generated from $p(x)$.

Simple Monte Carlo

Given $\left\{x^{(r)}\right\}_{r=1}^{R} \sim p(x)$ we can estimate the expectation $\mathop{\mathbb{E}}_{x\sim p(x)}[\phi(x)]$ using the estimator $\hat{\phi}$:

$$\Phi := \mathop{\mathbb{E}}_{x\sim p(x)}[\phi(x)] \approx \frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right) := \hat{\Phi}$$

$\hat{\Phi}$ *follows the same from the Law of Large Numbers (LLN).*
Moreover, $\hat{\Phi}$ is **unbiased** and its variance is $\frac{1}{R}\text{var}[\phi(x)]$, proof can be found here.

5.2 Three Sampling Methods

Ancestral Sampling

Ancestral sampling follows the same algorithm we saw from the conditional depencies of DAGMs. We start to sample the nodes with no parents and then sample any other conditional distribution at next steps.

Example 5.2.1. Suppose we have a DAGM whose joint distribution can be factorized as follows:

$$\begin{aligned} p\left(x_{1,\ldots,5}\right) &= \prod_{i}^{5} p\left(x_i \mid \text{parents}\left(x_i\right)\right) \\ &= p\left(x_1\right) p\left(x_2 \mid x_1\right) p\left(x_3 \mid x_1\right) p\left(x_4 \mid x_2, x_3\right) p\left(x_5 \mid x_3\right) \end{aligned}$$

The sampling process works as follows:

1. Start by sampling from $p\left(x_1\right)$.
2. Then sample from $p\left(x_2 \mid x_1\right)$ and $p\left(x_3 \mid x_1\right)$.
3. Then sample from $p\left(x_4 \mid x_2, x_3\right)$.
4. Finally, sample from $p\left(x_5 \mid x_3\right)$.

For example, to sample $p(x_2|x_1)$, we take x_1 as the time of a day and x_2 as the temperature. Different time on a day will have different distribution of temperature. For example, we can assume $x_2 \sim N(T(x_1), 3^2)$.

However, the main shortcuts of acestral sampling is that it can not compute the normlizing constant Z, where

$$p(x) = \frac{\tilde{p}(x)}{Z}$$
$$Z = \int \tilde{p}(x)dx$$

To compute Z, one **bad idea** is to divide x in to many intervals. However, it will be very hard if we have high-dimensional data.

To accommodate the high-dimensioanal errors, we have two other sampling methods which are **Importance Sampling** and **Rejection Sampling**.

Importance Sampling

Instead of sampling $p(x)$, we can choose $q(x)$ which is easier to sample from (like Gaussian). Our goal is still to estimate the expectation of $\phi(x)$. That said, we have:

$$p(x) = \frac{\tilde{p}(x)}{Z_p}$$
$$q(x) = \frac{\tilde{q}(x)}{Z_q}$$

We then generate R samples from $q(x)$: $\{x^{(r)}\}_{r=1}^{r=R}$. If the samples are also generated from $p(x)$, then we can use simple Monte Carlo; however, these are generated from $q(x)$.
To correct this, we introduce the weights, which works as:

1. we define weights as $\tilde{w}_r = \frac{\tilde{p}\left(x^{(r)}\right)}{\tilde{q}\left(x^{(r)}\right)} = \frac{Z_p}{Z_q}\frac{p\left(x^{(r)}\right)}{q\left(x^{(r)}\right)}$ and notice that

$$\frac{1}{R}\sum_{r=1}^{R}\tilde{w}_r \approx \mathop{\mathbb{E}}_{x\sim q(x)}\left[\frac{\tilde{p}(x)}{\tilde{q}(x)}\right] = \frac{Z_p}{Z_q}\int\frac{p(x)}{q(x)}q(x)dx = \frac{Z_p}{Z_q}$$

2. Finally, we rewrite our estimator under q

$$\phi = \int \phi(x)p(x)dx = \int \phi(x)\cdot\frac{p(x)}{q(x)}\cdot q(x)dx \approx \frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\frac{p\left(x^{(r)}\right)}{q\left(x^{(r)}\right)} = (*)$$

3. However, the estimator relies on p. It can only rely on $\tilde{p}$ and $\tilde{q}$.

$$\begin{aligned}(*) =& \frac{Z_q}{Z_p}\frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\cdot\frac{\tilde{p}\left(x^{(r)}\right)}{\tilde{q}\left(x^{(r)}\right)} = \frac{Z_q}{Z_p}\frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\cdot\tilde{w}_r \\ \approx& \frac{\frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\cdot\tilde{w}_r}{\frac{1}{R}\sum_{r=1}^{R}\tilde{w}_r} = \sum_{r=1}^{R}\phi\left(x^{(r)}\right)\cdot w_r = \hat{\Phi}_{iw}\end{aligned}$$

where $w_r = \frac{\tilde{k}_r}{\sum_{r=1}^{R}\tilde{w}_r}$ and $\hat{\Phi}_{iw}$ is our importance weighted estimator.

Rejection Sampling

The common thing between importance sampling and rejection sampling is that we use introduce an easier distribution $q(x)$ to estimate $p(x)$. However, in rejection sampling, we assume:

$$c\tilde{q}(x) > \tilde{p}(x), \forall x$$

The algorithm works as follows:

1. Generate two random numbers x and u.
   1.1 x is generated from $q(x)$.
   1.2 u is generated uniformly from the interval $[0, c\tilde{q}(x)]$.
2. Accept or reject the sample x by comparing the value of u with $\tilde{p}(x)$
   2.1 **Reject** x if $u > \tilde{p}(x)$
   2.2 Otherwise x is accepted; x is added to our set of samples $\left\{x^{(r)}\right\}$.

The reason why Rejection Sampling work is that $\mathbb{P}_{x\sim q}(x \in A \mid u \leq \tilde{p}(x)) = \mathbb{P}_{x\sim p}(x \in A)$(*Full derivation here*).

In other words, rejection sampling also has shortcuts. In high-dimensional case, the c will be very large and hence the acceptance rate will be very small. Moreover, such c will also be hard to define.

6 Markov Chain Monte Carlo

6.1 Introduction to Markov Chains

In most cases, we assume the samples are independent. However, it is a poor assumption as this may rarely happen in reality. Therefore, we assume the data are dependent and hence we have **sequential data**. More specifically, we can use **markov chain** to model the sequential data.

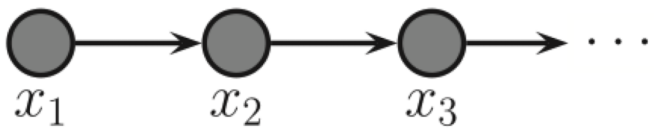


Figure 6.1: First order Markov Chain

Figure 6.1 shows an example of First order Markov chain.

$$\begin{aligned} p\left(x_t|x_{1:t-1}\right) &= p\left(x_t|x_{t-1}\right) \\ &= \prod_{t=1}^{T} p\left(x_t|x_{t-1}\right) \end{aligned}$$

From the graph, we can see that the first order means each node is conditioned on the previous node. We can therefore generalize to higher order Markov Chains.

- For second-order Markov Chains:

$$p\left(x_t|x_{1:t-1}\right) = p\left(x_t|x_{t-1}, x_{t-2}\right)$$

- For m-order Markov Chains:

$$p\left(x_{1:T}\right) = p\left(x_t|x_{t-1:t-m}\right)$$

We have two further defination:
Stationary(homongeneous) Markov Chain: The distribution generating the data does not change over time:

$$p(x_{t+1} = y|x_t = x) = p(x_{t+2} = y|x_{t+1} = x)$$

Non-stationary Markov Chain: $p(x_{t+1} = y|x_t = x)$ depends on time t.

Transition Matrix

If we assume x_t is discrete: $x_t \in \{1, 2, \cdots, K\}$. Then the conditional distribution $p(x_t|x_{t-1})$ can be written as a $K \times K$ matrix, which is called the **transition (stochastic) matrix** A. Each element on A represents the probability of transition from $t-1$ to t, i.e.

$$A_{ij} = p(x_t = j|x_{t-1} = i),\ A \in K \times K$$

We also have:

$$\begin{aligned} p\left(x_t = j\right) &= \sum_i p\left(x_t = j|x_{t-1} = i\right) p\left(x_{t-1} = i\right) \\ &= \sum_i A_{ij} p\left(x_{t-1} = i\right) \end{aligned}$$

The sum for each row in A equals to $\mathbf{1}$ as it represent all the possibilities of changing from $t-1$ to t ginve i at $t-1$.

$$\sum_i A_{ij} = 1$$

An example of transition matrix can be found here.
The n-step transition matrix $A(n)$is the probability of transferring from two states in n steps, where n can be considered as *"n"* times. In equation:

$$A_{ij}(n) = p\left(x_{t+n} = j|x_t = i\right)$$

Since the one-step transition matrix considers $t-1$ and t (or equivalently t and $t+1$)

$$A(1) = A$$

Chapman-Kolmogorov equations:

$$\begin{aligned} A_{ij}(m+n) &= \sum_{k=1}^{K} A_{ik}(m) A_{kj}(n) \\ \Leftrightarrow A(m+n) &= A(m)A(n) \end{aligned}$$

It says we can find an intermediate state k between i and j, which takes m steps from i to k, and n steps from k to j. It is similar to lattice paths on combinatorics.
$\Rightarrow A(n) = A(1+n-1) = A \cdot A(n-1) = A \cdot A \cdot A(n-2) = \cdots = A^n$.

Markov Language Models

Language models can be represented as Markov Chains. That said, each word is a state, and the probability of transition from one word to another can be also represented as transition matrix.

$$\begin{aligned} p\left(x_{1:T}|\theta\right) &= \pi\left(x_1\right) A\left(x_1, x_2\right) \cdots A\left(x_{T-1}, x_T\right) \\ &= \underbrace{\prod_{j=1}^{K} \pi_j^{1[x_1=j]}}_{\text{prob. starting with } x_1} \prod_{t=2}^{T} \prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{1[x_t=k, x_{t-1}=j]} \end{aligned}$$

6.2 Stationary distribution of a (homongeneous) Markov Chain

The stationary[1] distribution is the **long-term** distribution over states. It is also called the **steady state**. In notation, we have $\pi_t(j) = p(x_t = j)$, to be the probability of being in state j on time t.

- We have the initial distribution which is given by $\pi_0 \in \mathbb{R}^K$
- We then represent π_1 as:

$$\pi_1(j) = \sum_{i=1}^{K} p\left(x_1 = j | x_0 = i\right) \pi_0(i) = \sum_{i=1}^{K} A_{ij} \pi_0(i) = \sum_{i=1}^{K} \left(A^\top\right)_{ji} \pi_0(i)$$

$$\Rightarrow \pi_1 = A^T \pi_0$$

- We can therefore generalize to π_t.

$$\pi_t = A^\top \pi_{t-1} = A^\top A^\top \pi_{t-2} = \cdots = \left(A^\top\right)^t \pi_0$$

- If we do this for many steps; eventually, the distribution of π_t may converge:

$$\pi = A^T \pi$$

  which is the stationary distribution of the Markov chain[2]

Recall: Eigenvalues and Eigenvectors
Let T be a transformation, $\mathbf{v}$ be a vector and λ be a constant. If we have:

$$T\mathbf{v} = \lambda \mathbf{v}$$

Then $\mathbf{v}$ is the eigenvector of T and λ is the eigenvalue.

Therefore, it follows that the statioanary distribution π is an eigenvector of A^T with eigenvalue 1. In other words, A and A^T have the same eigenvalues and $A\mathbf{1} = \mathbf{1}$ as the row sum of A is 1. Thus, due to the equality of characteristics polynomials, 1 is also the eigenvalue of A^T. More importantly, **stationary distribution is NOT unique**.

[1] the "stationary" in "stationary distribution" is not as same as in "stationary Markov chain." Therefore, we usually called it as homongeneous Markov chains.

[2] An example of how to compute the stationary distribution can be found here.

Detailed balance equation

Markov Chain is called:

- **irreducible** if we can get from any state to any other state.
- **regular** if A^n has positive entries for some n.
- **time reversible** if there exists a distribution π such that

$$\underbrace{\pi_i A_{ij} = \pi_j A_{ji}}_{\textbf{detailed balance equation}} \qquad \text{for all } i, j.$$

  In other words, detailed balance means the probability of transition from one state to another and reverse back is **equally possible**.

Theorem 6.2.1. If a markov chain with transition matrix A satisfies the detailed balance with respect to distribution π, then π is a stationary distribution.

Recall in rejection sampling, we accept the samples independently. In contrast, we assume the samples are **dependent**, where each sample is a state in the Markov chain and $x^{(t)}$ has a probability distribution depending on x^{t-1}. In other words, we will **learn** from the previous samples. Hence the stationary distribution will be $p(x)$.

6.3 Metropolis-Hasting Algorithms

Compared to rejection sampling, metropolis-hasting prose a simple density q which depends on the current state $x^{(t)}$. If q is Gaussian, then its center will be $x^{(t)}$. As before, assume we can evaluate $\tilde{p}(x)$ for any x. The algorithm follows as:

- We firstly have a tentative new state x' is generated from the proposal density $q\left(x'|x^{(t)}\right)$. We accept the new state with probability
- We then compute:

$$A\left(x'|x^{(t)}\right) = \min\left\{1, \underbrace{\frac{\tilde{p}\left(x'\right) q\left(x^{(t)}|x'\right)}{\tilde{p}\left(x^{(t)}\right) q\left(x'|x^{(t)}\right)}}_{\mathcal{B}}\right\}$$

- for $\mathcal{B}$
  - If $\mathcal{B} > 1$, then accept the new state $\prime x$.
  - If $\mathcal{B} < 1$, we sample $u \sim N(0, 1)$. Accept if $u \leq \mathcal{B}$; otherwise reject.

Metropolis algorithm: Simpler version when $q\left(x'|x\right) = q\left(x \mid x'\right)$ for all x, x'.

Theorem 6.3.1. This procedure defines a Markov chain with stationary distribution $\pi(x)$ equal to the target distribution $p(x)$. $\pi(x) = p(x)$. *proof here*

6.4 Gibbs Samplign Procedure

Gibbs sampling is used when we have two or more dimensions to sample. For example, for bivariate gaussian distribution, $p(x, y)$ is difficult to compute but the conditional distribution $p(y|x)$ and $p(x|y)$ are easier. It works as follows:

- Suppose the vector x is d-dimensional

$$x = (x_1, \ldots, x_d)\,.$$

- Start with any $x^{(0)} = \left(x_1^{(0)}, \ldots, x_d^{(0)}\right)$. For $j = 1, \ldots, d$:
  - Sample $x_j^{(t)}$ from the conditional distribution given other components:

$$x_j^{(t)} \sim p\left(x_j \mid x_{-j}^{(t-1)}\right)$$

    Where $x_{-j}^{(t-1)}$ represents all the components of x except for x_j at their current values:

$$x_{-j}^{(t-1)} = \left(x_1^{(t)}, x_2^{(t)}, \ldots, x_{j-1}^{(t)}, x_{j+1}^{(t-1)}, \ldots, x_d^{(t-1)}\right)$$

- No accept/reject, only accept.

In bivariate gaussian cases, we just sample y and x iteratively.

Recap: Conditional Distributio on Bivariate Gaussian

Suppose we have two random variables X_1 and X_2, which follow Gaussian Distribution.

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N_2(\mu, \Sigma), \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}.$$

Then:

$$X_1 \mid X_2 = x_2 \sim N\left(\mu_1 + \rho\left(x_2 - \mu_2\right), 1 - \rho^2\right)$$
$$X_2 \mid X_1 = x_1 \sim N\left(\mu_2 + \rho\left(x_1 - \mu_1\right), 1 - \rho^2\right)$$

Given $X^{(0)} = (0, 0)$ we proceed iteratively for $t \geq 1$:

$$X_1^{(t)} \sim N\left(\mu_1 + \rho\left(x_2^{(t-1)} - \mu_2\right), 1 - \rho^2\right)$$
$$X_2^{(t)} \sim N\left(\mu_2 + \rho\left(x_1^{(t)} - \mu_1\right), 1 - \rho^2\right)$$

6.5 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo is a specialized Metropolis-Hasting algorithm. We will use physical analogy to make samples.

- We assume the **potential energy** for each data point is $E(x)$.
- The distribution $p(x)$:

$$p(x) \propto e^{-E(x)}, \quad \text{with} \quad E(x) = -\log(\tilde{p}(x))$$

- Introduce **momentum** v carrying the kinetic energy

$$K(v) = \frac{1}{2}\|v\|^2 = \frac{1}{2}v^\top v.$$

- The **total energy/hamiltonian** is the sum of potential energy and kinetic energy

$$H(x, v) = E(x) + K(v).$$

- The joint distribution will be:

$$p(x, v) \propto e^{-E(x)}e^{-K(v)} = e^{-E(x)-K(v)} = e^{-H(x,v)}$$

- Same as physic, we assume the energy is preserved. That said:
  - Frictionless ball rolling $(x, v) \to (x', v')$
  - $H(x, v) = H(x', v')$.
- Ideal Hamiltonian dynamics are reversible: reverse v and the ball will return to its start point! $(x, v) \to (x', v')$ but also $(x', -v') \to (x, -v)$

Compared to Metropolis-Hastings, Hamiltonian Monte Carlo generate the momentum and move to a new postion, which is the new state x'.

Leap-frog integrator

Like we just introduced, we want to move to a new state (x', v'). The numerical approximation is finished by leap-frog integrator. One complete leap-frog integrator works as follows. Let ϵ as the step size:

1. Momentum half update

$$v\left(t + \frac{\epsilon}{2}\right) = v(t) + \frac{\epsilon}{2}\frac{\mathrm{d}v}{\mathrm{d}t}(t) = v(t) - \frac{\epsilon}{2}\left.\frac{\partial E}{\partial x}\right|_{(x(t))}$$

2. Position full update

$$x(t + \epsilon) = x(t) + \epsilon\frac{\mathrm{d}x}{\mathrm{d}t}(t) = x(t) + \epsilon\left.\frac{\partial K}{\partial v}\right|_{\left(v\left(t+\frac{\epsilon}{2}\right)\right)}$$

3. Momentum full update

$$v(t + \epsilon) = v\left(t + \frac{\epsilon}{2}\right) - \frac{\epsilon}{2}\left.\frac{\partial E}{\partial x}\right|_{(x(t+\epsilon))}$$

The resulting $x(t + \epsilon)$ and $v(t + \epsilon)$ are the x' and v'. We do a fixed number of leap-frog steps.

Full HMC algorithm

The HMC algorithm (run until it mixes):

- Current position: $\left(x^{(t-1)}, v^{(t-1)}\right))$
- Sample momentum: $v^{(t)} \sim \mathcal{N}(0, I)$.
- Start at $(x, v) = \left(x^{(t-1)}, v^{(t)}\right)$ and run Leapfrog integrator for L steps and reach (x', v')
- Accept new state $(x', -v')$ with probability:

$$\min\left\{1, \underbrace{\frac{\exp\left(H\left(x^{(t-1)}, v^{(t-1)}\right)\right)}{\exp\left(H\left(x', v'\right)\right)}}_{\mathcal{B}}\right\}$$

  Similar to Metropolis-hastings. For $\mathcal{B}$:

  - If $\mathcal{B} > 1$, then accept the new state $\prime x$.
  - If $\mathcal{B} < 1$, we sample $u \sim N(0, 1)$. Accept if $u \leq \mathcal{B}$; otherwise reject.

- **Low energy** points are favored.

6.6 MCMC Diagnostics

We use $\hat{R}$ and $\hat{n}_{eff}$ to diagnose MCMC.

- Once $\hat{R}$ is near 1 , and $\hat{n}_{\text{eff}}$ is more than 10 per chain for all scalar estimands we collect the mn simulations, (excluding the burn-in).
- We can then draw inference based on our samples. However:
- Even if the iterative simulations appear to have converged, passed all tests etc. It may still be far from convergence!
- When we declare "convergence" - we mean that all chains appear stationary and well mixed.

7 Hidden Markov Models

7.1 Introduction to Hidden Markov Models (Chains)

For Markov chains, the assumptions may be restrictive. The state of the variable may not be fully observed; instead, each state is contributed by a hidden variable z_i.

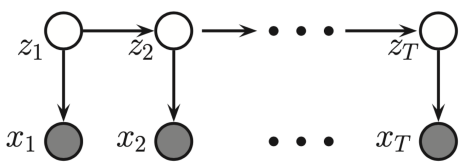


Figure 7.1: An example of Hidden Markov Chain

Above is an example of Hidden Markov Chain. It hides the temporal dependence by keeping it as unobserved state. For each observation x_t, we associated it with a **hidden/latent** state z_t. The joint distribution can be written as:

$$p\left(x_{1:T}, z_{1:T}\right) = p\left(z_1\right) \prod_{t=2}^{T} p\left(z_t \mid z_{t-1}\right) \prod_{t=1}^{T} p\left(x_t \mid z_t\right)$$

Hidden Markov Models are not limited by the assumptions of Markov Chains. In HMMs, we only need to know three sets of distributions:

1. **Initial distribution**. This is the probability of the first hidden state being i

$$\pi(i) = p(z_1 = i)$$

2. **Transition distribution**. The probability of hidden state of transition from state i to j.

$$A_{ij} = p(z_{t+1} = j | z_t = i)$$

3. **Emission probability**. The probability of an observed random variable x_t given its hidden variable.

$$p(x_t = j | z_t = i)$$

Moreover, we have two main objectives.

1. We want to compute the probability of a laten sequence given an observed sequence, which is:

$$p(z_{1:t} | x_{1:t})$$

   This is achieved by **Forward-backward algorithm**

2. Find the most likely unobservable sequence.

$$z^\star = \underset{z_{1:T}}{\operatorname{argmax}} p\left(z_{1:T} \mid x_{1:T}\right)$$

   using the **Viterbi algorithm**

7.2 Forward-backward algorithm

This task of hidden state inference breaks down into the following:

- **Filtering**: compute posterior over current hidden state, $p\left(z_t \mid x_{1:t}\right)$.
- **Prediction**: compute posterior over future hidden state, $p\left(z_{t+k} \mid x_{1:t}\right)$.
- **Smoothing**: compute posterior over past hidden state, $p\left(z_t \mid x_{1:T}\right) \quad 1 \leq t < T$.

Forward Recursion

Goal: Compute the **filtered marginals**

$$\alpha_t(j) = p\left(z_t = j \mid x_{1:t}\right)$$

given the initial distribution, transition distribution and emission probability described above. It has two steps:

1. **Prediction step**: compute the one-step-ahead predictive density:

$$\begin{aligned} p\left(z_t = j \mid x_{1:(t-1)}\right) &= \sum_{i=1}^{K} p\left(z_{t-1} = i, z_t = j \mid x_{1:(t-1)}\right) \\ &= \sum_{i=1}^{K} p\left(z_t = j \mid z_{t-1} = i, x_{1:(t-1)}\right) p\left(z_{t-1} = i \mid x_{1:(t-1)}\right) \\ &= \sum_{i=1}^{K} p\left(z_t = j \mid z_{t-1} = i\right) p\left(z_{t-1} = i \mid x_{1:(t-1)}\right) = \sum_{i=1}^{K} A_{ij}\alpha_{t-1}(i) = \left(A^\top \alpha_{t-1}\right)_j \end{aligned}$$

2. **Update step**: Denoting $\lambda_t(j) = p\left(x_t \mid z_t = j\right)$ (here x_t is fixed and $\lambda_t \in \mathbb{R}^K$)

$$\begin{aligned} \alpha_t(j) = p\left(z_t = j \mid x_{1:t}\right) &= p\left(z_t = j \mid x_{1:(t-1)}, x_t\right) \\ &\propto p\left(z_t = j, x_t \mid x_{1:(t-1)}\right) \\ &= p\left(x_t \mid z_t = j, x_{1:(t-1)}\right) p\left(z_t = j \mid x_{1:(t-1)}\right) \\ &= p\left(x_t \mid z_t = j\right) p\left(z_t = j \mid x_{1:(t-1)}\right) = \lambda_t(j) p\left(z_t = j \mid x_{1:(t-1)}\right) \end{aligned}$$

Backward Recursion

Goal: Compute:

$$\beta_t(i) = p\left(x_{(t+1):T} \mid z_t = i\right)$$

$$\begin{aligned} \beta_t(i) &= p\left(x_{(t+1):T} \mid z_t = i\right) \\ &= \sum_{j=1}^{K} p\left(z_{t+1} = j, x_{t+1}, x_{(t+2):T} \mid z_t = i\right) \\ &= \sum_j p\left(x_{(t+2):T} \mid z_{t+1} = j, z_t = i, x_{t+1}\right) p\left(x_{t+1} \mid z_{t+1} = j, z_t = i\right) p\left(z_{t+1} = j \mid z_t = i\right) \\ &= \sum_j p\left(x_{(t+2):T} \mid z_{t+1} = j\right) p\left(x_{t+1} \mid z_{t+1} = j\right) p\left(z_{t+1} = j \mid z_t = i\right) \\ &= \sum_j \beta_{t+1}(j)\lambda_{t+1}(j)A_{ij} \end{aligned}$$

Smoothing

In smoothing inference, We can break the chain into two parts, the past and the future, by conditioning on z_t :

$$\begin{aligned} \gamma_t := p\left(z_t \mid x_{1:T}\right) &\propto p\left(z_t, x_{1:T}\right) = p\left(z_t, x_{1:t}, x_{(t+1):T}\right) \\ &= p\left(z_t, x_{1:t}\right) p\left(x_{(t+1):T} \mid z_t, x_{1:t}\right) = p\left(z_t, x_{1:t}\right) p\left(x_{(t+1):T} \mid z_t\right) \\ &\propto p\left(z_t \mid x_{1:t}\right) p\left(x_{(t+1):T} \mid z_t\right) \\ &= (\text{ Forward Recursion})(\text{Backward Recursion}) \end{aligned}$$

- Here we use the conditional independence $x_{(t+1):T} \perp x_{1:t} \mid z_t$.
- We know how to perform forward recursion from the previous part.

After we complete the forward and backward recursion, we now can compute the marginal probabilities of $p(z_{1:t}|x_{1:t})$, $p(z_t|x_{1:T})$.

7.3 Viterbi Algorithm

The Viterbi Algorithm is used to compute the most probable laten sequence.

$$\hat{z} = \arg\max_{z_{1:T}} p\left(z_{1:T} \mid x_{1:T}\right)$$

It is a specialized version of max-product inference. The forward pass uses max-product, and the backward use the traceback procedure to recover the most probable sequence. It works as follows:

1. Define δ_t via
$$\delta_t(j) = \max_{z_1,\ldots,z_{t-1}} p\left(z_{1:(t-1)}, z_t = j, x_{1:t}\right)$$
which is the probability of ending up in state j at time t, by taking the most probable path.

2. Moreover,
$$\begin{aligned}
\delta_t(j) &= \max_{z_1,\ldots,z_{t-1}} p\left(z_{1:(t-1)}, z_t = j, x_{1:(t-1)}, x_t\right) \\
&= \max_{z_1,\ldots,z_{t-1}} p\left(z_{1:(t-1)}, x_{1:(t-1)}\right) p\left(z_t = j \mid z_{t-1}\right) p\left(x_t \mid z_t = j\right) \\
&= \max_i \max_{z_1,\ldots,z_{t-2}} p\left(z_{1:(t-2)}, z_{t-1} = i, x_{1:(t-1)}\right) p\left(z_t = j \mid z_{t-1} = i\right) p\left(x_t \mid z_t = j\right) \\
&= \max_i \delta_{t-1}(i) A_{ij} \lambda_t(j)
\end{aligned}$$

Keep track of the most likely previous state: $\theta_t(j) = \arg\max_i \delta_{t-1}(i) A_{ij} \lambda_t(j)$.
In practice:

- Initialize the algorithm with
$$\delta_1(j) = p\left(z_1 = j, x_1\right) = \pi_j \lambda_1(j).$$
- and terminate with
$$z_T^* = \arg\max_i \delta_T(i)$$
- Then, we compute the most probable sequence of states using traceback:
$$z_t^* = \theta_{t+1}\left(z_{t+1}^*\right)$$

8 Variational Inference

8.1 Introduction to Variational Inference

variational inference is an inference method which can help us to approximate a tractable inference. "Variational" refers to the technique in which complex probability distributions are approximated using simpler, parameterized distributions by optimizing over the parameters. We already see a similar application in rejection sampling, where $q(x)$ is the simpler distribution and help us to estimate the desired distribution, $p(x)$.

The posterior distribution of latent variable models

Recall the joint distribution $p(x, z) = p(x)p(x|z)$. x are the observed data points, z are the unobserved (latent) data points. Moreover:

$$\underbrace{p(z|x)}_{\text{posterior}} = \frac{\overbrace{p(x|z)}^{\text{likelihood}}\overbrace{p(z)}^{\text{prior}}}{\underbrace{p(x)}_{\text{normalizing factor}}}$$

Example 8.1.1. The TrueSkill model we introduced earlier is indeed a latent variable model.

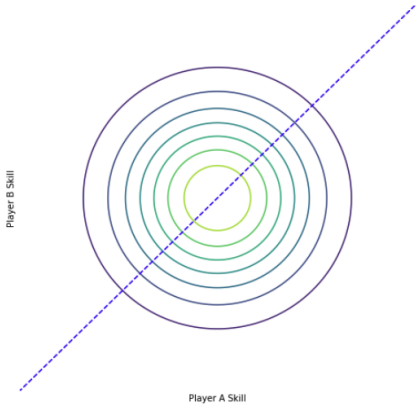


Figure 8.1: Contour plot between two players

The above graph shows the contour graph between the two players. Since we assume both players follows Gaussian distribution, we are uncertain about players skills and hence we observe perfect "circles." Each contour represent different values of pdf. It goes higher when we move to the center. The blue dashed line represent equal skills between the two players. Here is 3D view of the bivariate Gaussian distribution.

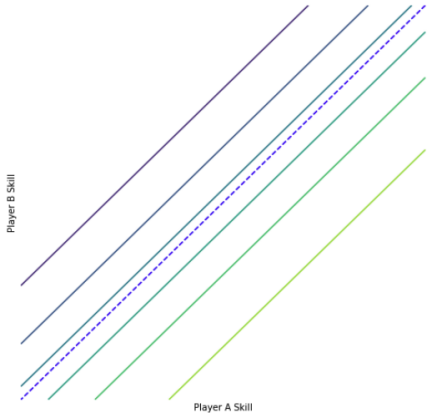


Figure 8.2: 2D projection of likelihood

The above graph shows the 2D projection of the likelihood $p(i\text{i beats j}|z_i, z_j) = \sigma(z_i - z_j)$.

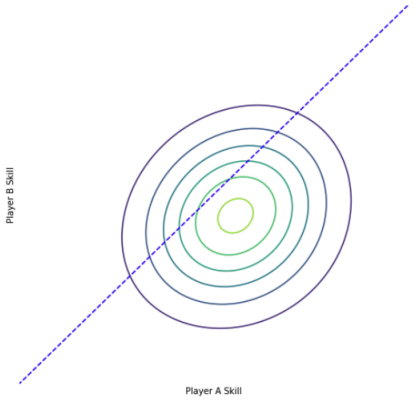


Figure 8.3: The posterior

The contours are not perfect "circles" anymore, which are not Gaussian anymore. However, the center moves to the left and we conclude that the guess player A is better than B.

8.2 Kullback-Leibler divergence

From the TrueSkill model, the normalizing factor $p(x) = \int p(x, z)dz$ is intractable when z is high-dimensional. To estimate $p(x)$, like previously explained, our procedure is to approximate it using a simpler distribution $q(x)$. We measure the closeness/similarity between p and q using **KL Divergence**.

$$\mathrm{KL}(q(z)\|p(z|x)) = \int q(z) \log \frac{q(z)}{p(z|x)} dz = \mathop{\mathbb{E}}_{z\sim q} \log \frac{q(z)}{p(z|x)}$$

- $\mathrm{KL}(q\|p) \geq 0$
- $\mathrm{KL}(q\|p) = 0 \Leftrightarrow q = p$
- $\mathrm{KL}(q\|p) \neq \mathrm{KL}(p\|q)$
- KL divergence is not a metric, since it is not symmetric.

$\mathrm{KL}(q\|p)$ and $\mathrm{KL}(p\|q)$ corresponds to different projection.

Information (I-) Projection

For I-projection, we want to find the $\star q$ such that:

$$q^* = \arg\min_{q\in Q} \mathrm{KL}(q\|p) = \mathbb{E}_{x\sim q(x)} \log \frac{q(x)}{p(x)}$$

- We want to minimize KL divergence from q (approximating distribution) to p (target distribution).
- We focuses on the simpler model that captures the main features of the target distribution.
- Thus I-projection will not yield the correct moments.

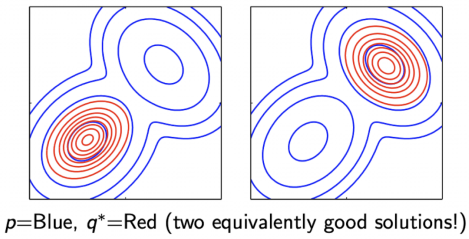


Figure 8.4: An example of I-projection

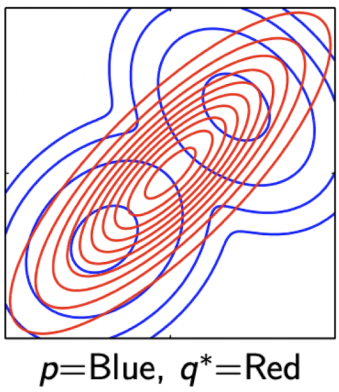
Moment (M-) Projection

For I-projection, we want to find the $\star q$ such that:

$$q^* = \arg\min_{q \in Q} \mathrm{KL}(p\|q) = \mathbb{E}_{x\sim q(x)} \log \frac{p(x)}{q(x)}$$

- We want to minimize KL divergence from p (target distribution) to p (approximating distribution).
- We Preserves the characteristics of p as much as possible in q
- Thus M-projection will yield the correct moments.

Figure 8.5: An example of M-projection



8.3 Mean-field approach

We know that MRF can be written as exponential families:

$$p(x|\theta) = \exp\left\{\sum_{C\in\mathcal{C}} \phi_C\left(x_C\right) - \log Z(\theta)\right\}$$

By performing I-projection:

$$\mathrm{KL}(q\|p) = \mathbb{E}_{x\sim q(x)} \log \frac{q(x)}{p(x|\theta)} = \mathbb{E}_{x\sim q(x)} \left[\log q(x) - \sum_{C\in\mathcal{C}} \phi_C\left(x_C\right) + \log Z(\theta)\right]$$

Denote **entropy** as $H(p) = -\mathbb{E}_{x\sim p(x)} \log p(x)$, which measures the uncertainty in p

$$\arg\min_{q\in Q} \mathrm{KL}(q\|p) = \arg\max_{q\in Q} \left\{ \sum_{C\in\mathcal{C}} \mathbb{E}_q \left[\phi_C\left(x_C\right)\right] + H(q) \right\}$$

Using the KL divergence, we need to define a nice set of Q, if $p \in Q$, then we are done. However, it is nearly impossible and we should take **mean-field approach**.

The key assumption for mean-field approach is that we assume Q is composed of those distributions that factor out:

$$q(x) = \prod_{i\in V} q_i\left(x_i\right)$$

Therefore, q^* is now equivalent to:

$$q^* = \arg\max_{q\in Q} \sum_{C\in\mathcal{C}} \underbrace{\sum_{x_C} q_C\left(x_C\right)\phi_C\left(x_C\right)}_{\mathbb{E}_q[\phi_C(x_C)]} + H(q)$$

For the entropy:

$$\begin{aligned} H(q) = \mathbb{E}_q[-\log q(x)] &= -\sum_x q(x)\log q(x) = -\sum_x q(x)\left[\sum_i \log q_i\left(x_i\right)\right] \\ &= -\sum_i \sum_x \left[q_i\left(x_i\right)\log q_i\left(x_i\right)\right]\frac{q(x)}{q_i\left(x_i\right)} = -\sum_i \sum_{x_i} \left[q_i\left(x_i\right)\log q_i\left(x_i\right)\right]\sum_{x_i}\frac{q(x)}{q_i\left(x_i\right)} \\ &= -\sum_i \sum_{x_i} \left[q_i\left(x_i\right)\log q_i\left(x_i\right)\right] = \sum_i H\left(q_i\right) \end{aligned}$$

This tells us the general result on how to find $H(q)$ from each q_i, we can thus write everything as a function of q_i for $i \in V$.

8.4 Evidence Lower Bound (ELBO)

$$\begin{aligned} \mathrm{KL}\left(q_\phi(z)\|p(z\mid x)\right) &= \mathop{\mathbb{E}}_{z\sim q_\phi} \log\frac{q_\phi(z)}{p(z\mid x)} \\ &= \mathop{\mathbb{E}}_{z\sim q_\phi}\left[\log\left(q_\phi(z)\cdot\frac{p(x)}{p(z,x)}\right)\right] \\ &= \mathop{\mathbb{E}}_{z\sim q_\phi}\left[\log\frac{q_\phi(z)}{p(z,x)}\right] + \mathop{\mathbb{E}}_{z\sim q_\phi}\log p(x) \\ &:= -\underbrace{\mathcal{L}(\phi)}_{ELBO} + \log p(x) \end{aligned}$$

Further simplification, we have:

$$\begin{aligned} \mathcal{L}(\phi) &= \mathop{\mathbb{E}}_{z\sim q_\phi}\left[\log p(z,x) - \log q_\phi(z)\right] \\ &= \underbrace{\mathop{\mathbb{E}}_{z\sim q_\phi}\left[\log p(z,x)\right]}_{\text{expected log-join}} + \underbrace{\mathop{\mathbb{E}}_{z\sim q_\phi}\left[-\log q_\phi(z)\right]}_{\text{entropy}} \end{aligned}$$

Rearrange, we have:

$$\mathcal{L}(\phi) + \underbrace{\mathrm{KL}\left(q_\phi(z) \| p(z \mid x)\right)}_{\geq 0} = \log p(x)$$
$$\Rightarrow \mathcal{L}(\phi) \leq \log p(x)$$

Therefore, maximizing the ELBO is equivalent to minimize the KL divergence.

8.4.1 Maximizing the KL divergence

Since $\nabla\mathcal{L}(\phi)$ is the direction of the steepest ascent of $\mathcal{L}(\phi)$. In other words, imagine you are climbing a mountain, the corresponding $\nabla\mathcal{L}(\phi)$ at any points represent the fastest direction (way) to climb upwards. Equivalently, $-\nabla\mathcal{L}(\phi)$ is the direction of the steepest descent of $\mathcal{L}(\phi)$.

We will use method of **gradient descent.**

$$\phi_{t+1} = \phi_t + \underbrace{s_t}_{\text{negative}} \nabla\mathcal{L}\left(\phi_t\right)$$

In general, to estimate the gradient of $\mathbb{E}(f(Y, \phi))$, if Y is independent of ϕ, then:

$$\nabla_\phi \mathbb{E}(f(Y, \phi)) = \mathbb{E}\left(\nabla_\phi f(Y, \phi)\right)$$

However, since z is not independent of ϕ, meaning that

$$\nabla_\phi \mathop{\mathbb{E}}_{z\sim q_\phi}\left[\log p(x, z) - \log q_\phi(z)\right] \neq \mathop{\mathbb{E}}_{z\sim q_\phi}\left[\nabla_\phi\left(\log p(x, z) - \log q_\phi(z)\right)\right]$$

Reparametrization Trick

We can solve this problem by reparametrization trick. Suppose that $z \sim q_\phi$ has the same distribution as $T(\epsilon, \phi)$, where ϵ is a random variable whose distribution p_0 does not depend on ϕ. In this case, to sample $z \sim q_\phi$ by:

- Sampling a random variable $\epsilon \sim p_0$,
- Deterministically computing $z = T(\epsilon, \phi)$.

If $z = T(\epsilon, \phi)$, then

$$\mathop{\mathbb{E}}_{z\sim q_\phi}\left[\log p(x, z) - \log q_\phi(z)\right] = \mathop{\mathbb{E}}_{\epsilon\sim p_0}\left[\log p(x, T(\epsilon, \phi)) - \log q_\phi(T(\epsilon, \phi))\right]$$

Equivalently

$$\begin{aligned}\nabla_\phi\mathcal{L}(\phi) &= \nabla_\phi\mathbb{E}_{z\sim q_\phi(z)}\left[\log p(x, z) - \log q_\phi(z)\right] \\ &= \nabla_\phi\mathbb{E}_{\epsilon\sim p_0(\epsilon)}\left[\log p(x, T(\phi, \epsilon)) - \log q_\phi(T(\phi, \epsilon))\right] \\ &= \mathbb{E}_{\epsilon\sim p_0(\epsilon)}\nabla_\phi\left[\log p(x, T(\phi, \epsilon)) - \log q_\phi(T(\phi, \epsilon))\right].\end{aligned}$$

so generating a sample $\epsilon_1, \ldots, \epsilon_m$ from p_0, we get

$$\nabla_\phi\mathcal{L}(\phi) \approx \underbrace{\frac{1}{m}\sum_{i=1}^{m}\nabla_\phi\left[\log p\left(x, T\left(\phi, \epsilon_i\right)\right) - \log q_\phi\left(T\left(\phi, \epsilon_i\right)\right)\right].}_{\text{Simple Monte Carlo}}$$

8.4.2 Stochastic Variational Inference

Recall: $\mathcal{L}(\theta, \phi_{1:N} \mid x_{1:N}) = \sum_{n=1}^{N} \mathcal{L}(\theta, \phi_n \mid x_n)$
Instead of computing the full gradient with respect to θ (which is in general not possible), we compute a simple Monte Carlo estimate of it.
For example, at each step we can draw a random minibatch of $B = |\mathcal{B}|$ examples from the dataset, and then make an approximation sub-sauple descent.

$$\mathcal{L}(\theta, \phi_{1:N} \mid x_{1:N}) \approx \frac{N}{B} \sum_{x_n \in \mathcal{B}} \mathcal{L}(\theta, \phi_n \mid x_n)$$

9 Gaussian Mixture Models

In simple words, Gaussian Mixture Models combine gaussian models into a complex model by taking a mixture of K multivariate Gaussian densities of the form:

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

where $\pi_k \geq 0, \sum_{k=1}^{K} \pi_k = 1$, and $N_m(x|\mu_k, \Sigma_k)$ is the m-dim Gaussian density.
Each Gaussian component has its own mean vector μ_k and covariance matrix Σ_k.
The parameters π_k are called the **mixing coefficients.**

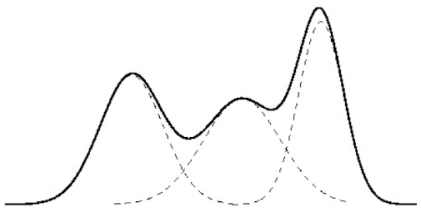


Figure 9.1: An example of Gaussian Mixture Models

10 Appendix

10.1 Example 1

The Bernoulli Naïve Bayes model parameterized by θ and π defines the following joint probability of x and c,

$$p(x, c|\theta, \pi) = p(c|\pi)p(x|c, \theta) = p(c|\pi)\prod_{j=1}^{D} p(x_j|c, \theta),$$

where $x_j|c, \theta \sim \text{Bernoulli}(\theta_{jc})$, i.e. $p(x_j|c, \theta) = \theta_{jc}^{x_j}(1-\theta_{jc})^{1-x_j}$, and $c|\pi$ follows a simple categorical distribution, i.e. $p(c|\pi) = \pi_c$.

Solution:

For $\mathbf{x^1}$, its likelihood function is:

$$L(\theta, \pi; x^1, c) = \prod_{c=1}^{10} \left[p(x^1, c \mid \theta, \pi)\right]^{1\{c^1=c\}} \tag{10.1}$$

$$= \prod_{c=1}^{10} \left[p(c \mid \pi) \prod_{j=1}^{784} p(x_j^1 \mid c, \theta)\right]^{1\{c^1=c\}} \tag{10.2}$$

$$= \prod_{c=1}^{10} \left[\pi_c \prod_{j=1}^{784} \theta_{jc}^{x_j^1}(1-\theta_{jc})^{1-x_j^1}\right]^{1\{c^1=c\}} \tag{10.3}$$

Therefore, the joint likelihood function for $\mathbf{x^1}, \cdots \mathbf{x^n}$ is:

$$L(\theta, \pi) = \prod_{i=1}^{n}\prod_{c=1}^{10} \left[\pi_c \prod_{j=1}^{784} \theta_{jc}^{x_j^i}(1-\theta_{jc})^{1-x_j^i}\right]^{1\{c^i=c\}} \tag{10.4}$$

$$\Rightarrow l(\theta, \pi) = \log\left(\prod_{i=1}^{n}\prod_{c=1}^{10} \left[\pi_c \prod_{j=1}^{784} \theta_{jc}^{x_j^i}(1-\theta_{jc})^{1-x_j^i}\right]^{1\{c^i=c\}}\right) \tag{10.5}$$

$$= \sum_{i=1}^{n}\sum_{c=1}^{10} 1\{c^i = c\}\left\{\log(\pi_c) + \sum_{j=1}^{784}\left[x_j^i \log(\theta_{jc}) + (1-x_j^i)\log(1-\theta_{jc})\right]\right\} \tag{10.6}$$

$$= \sum_{i=1}^{n}\sum_{c=1}^{9} 1\{c^i = c\}\left\{\log(\pi_c) + \sum_{j=1}^{784}\left[x_j^i \log(\theta_{jc}) + (1-x_j^i)\log(1-\theta_{jc})\right]\right\} \tag{10.7}$$

$$+ \sum_{i=1}^{n} 1\{c^i = 10\}\left\{\log(1-\sum_{c=1}^{9}\pi_c) + \sum_{j=1}^{784}\left[x_j^i \log(\theta_{j,10}) + (1-x_j^i)\log(1-\theta_{j,10})\right]\right\} \tag{10.8}$$

If we pick any $c \in [C]$ and $j \in [D]$:

$$\frac{\partial l(\theta, \pi)}{\partial \theta_{jc}} = \sum_{i=1}^{n} 1\{c^i = c\}\left(\frac{x_j^i}{\theta_{jc}} - \frac{1-x_j^i}{1-\theta_{jc}}\right) \tag{10.9}$$

Letting it equal to zero, we have:

$$\hat{\theta_{jc}} = \frac{\sum_{i=1}^{n} 1\{c^i = c\} x_j^i}{\sum_{i=1}^{n} 1\{c^i = c\}} \tag{10.10}$$

For π_c:

$$\frac{\partial l(\theta, \pi)}{\partial \pi_c} = \sum_{i=1}^{n} 1\{c^i = c\} \frac{1}{\pi_c} - \sum_{i=1}^{n} 1\{c^i = 10\} \frac{1}{1 - \sum_{c=1}^{9} \pi_c} \tag{10.11}$$

Letting $n_c = \sum_{i=1}^{n} 1\{c^i = c\}$, when it equals to zero, we have:

$$n_c(1 - \sum_{c=1}^{9} \hat{\pi_c}) = \hat{\pi_c} n_{10}, \quad \text{where } 1 \leq a \leq 9 \tag{10.12}$$

Summation on both sides over $1 \leq c \leq 9$, we have:

$$\sum_{c=1}^{9} n_c(1 - \sum_{c=1}^{9} \hat{\pi_c}) = \sum_{c=1}^{9} \hat{\pi_c} n_{10} \tag{10.13}$$

$$\Rightarrow (n - n_{10})(1 - \sum_{c=1}^{9} \hat{\pi_c}) = n_{10} \sum_{c=1}^{9} \hat{\pi_c} \tag{10.14}$$

$$\sum_{c=1}^{9} \hat{\pi_c} = \frac{n - n_{10}}{n} \tag{10.15}$$

Substituting back, we will have:

$$\hat{\pi_c} = \frac{n_c}{n}, \quad 1 \leq c \leq 9 \tag{10.16}$$

10.2 Example 2

We can write this distribution as an exponential family

$$p(x \mid \theta) = \theta^x (1 - \theta)^{1-x} \tag{10.17}$$

$$= \exp\{x \log(\theta) + (1 - x) \log(1 - \theta)\} \tag{10.18}$$

$$= \exp\left\{x \log\left(\frac{\theta}{1 - \theta}\right) + \log(1 - \theta)\right\} \tag{10.19}$$

Here,

$$T(x) = x$$

$$\eta = \log\left(\frac{\theta}{1 - \theta}\right)$$

$$A(\eta) = \log(1 + e^{\eta})$$

$$h(x) = 1$$

Notice that $A'(\eta) = \frac{e^{\eta}}{1 + e^{\eta}} = \theta$ is the mean of $T(X) = X$ and $A''(\eta) = \frac{e^{\eta}}{(1 + e^{\eta})^2} = \theta(1 - \theta)$ is the variance of X.

10.3 Derivations 1

We add and subtract $\mathbb{E}[t \mid x]$ and write

$$\begin{aligned}
\mathbb{E}[L] &= \iint (y(x) - t)^2 p(x,t) dx dt \\
&= \iint (y(x) - \mathbb{E}[t \mid x] + \mathbb{E}[t \mid x] - t)^2 p(x,t) dx dt \\
&= \iint (y(x) - \mathbb{E}[t \mid x])^2 p(x,t) dx dt + \iint (\mathbb{E}[t \mid x] - t)^2 p(x,t) dx dt \\
&\quad + 2 \iint (y(x) - \mathbb{E}[t \mid x])(\mathbb{E}[t \mid x] - t) p(x,t) dx dt
\end{aligned}$$

The last term is zero since

$$\begin{aligned}
&\iint (y(x) - \mathbb{E}[t \mid x])(\mathbb{E}[t \mid x] - t) p(x,t) dx dt \\
&= \iint (y(x) - \mathbb{E}[t \mid x])(\mathbb{E}[t \mid x] - t) p(t \mid x) p(x) dx dt \\
&= \int (y(x) - \mathbb{E}[t \mid x]) \{ \underbrace{\int (\mathbb{E}[t \mid x] - t) p(t \mid x) dt}_{=0} \} p(x) dx = 0
\end{aligned}$$

10.4 Example 3

This shape of tree is from Figure 4.1. To have concrete numbers, suppose all variables are binary $\{0, 1\}$ and take $\psi_i(x_i) \equiv 1$ with

$$\psi_{12} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad \psi_{13} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \psi_{34} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}, \quad \psi_{35} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

We have

$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z} \prod_{i=1}^{5} \psi_i(x_i) \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{34}(x_3, x_4) \psi_{35}(x_3, x_5).$$

Since $x_E = \{x_2, x_4, x_5\}$, we can just fix the values of three variables: $\bar{x}_2 = 1, \bar{x}_4 = 1, \bar{x}_5 = 0$. Then

$$p(x_1, 1, x_3, 1, 0) = \frac{1}{Z} \psi_{12}(x_1, 1) \psi_{13}(x_1, x_3) \psi_{34}(x_3, 1) \psi_{35}(x_3, 0).$$

This gives

$$\begin{aligned}
p(0,1,0,1,0) &= \frac{1}{Z} 2 \cdot 2 \cdot 1 \cdot 1 = \frac{4}{Z} \\
p(0,1,1,1,0) &= \frac{1}{Z} 2 \cdot 1 \cdot 2 \cdot 1 = \frac{4}{Z} \\
p(1,1,0,1,0) &= \frac{1}{Z} 1 \cdot 1 \cdot 1 \cdot 1 = \frac{1}{Z} \\
p(1,1,1,1,0) &= \frac{1}{Z} 1 \cdot 2 \cdot 2 \cdot 1 = \frac{4}{Z}
\end{aligned}$$

We can also compute the joint distribution:

$$\begin{aligned} p\left(x_1, x_3 \mid x_2 = 1, x_4 = 1, x_5 = 0\right) &= \frac{p\left(x_1, 1, x_3, 1, 0\right)}{\sum_{x_1', x_3'=0}^{1} p\left(x_1', 1, x_3', 1, 0\right)} \\ &= \frac{\frac{1}{Z}\psi_{12}\left(x_1, 1\right)\psi_{13}\left(x_1, x_3\right)\psi_{34}\left(x_3, 1\right)\psi_{35}\left(x_3, 0\right)}{\frac{1}{Z}(4+4+1+4)} \\ &= \frac{\psi_{12}\left(x_1, 1\right)\psi_{13}\left(x_1, x_3\right)\psi_{34}\left(x_3, 1\right)\psi_{35}\left(x_3, 0\right)}{13} \\ &= \frac{1}{13}\begin{bmatrix} 4 & 4 \\ 1 & 4 \end{bmatrix} \end{aligned}$$

For the message passing:

$$\begin{aligned} m_{2\to1}\left(x_1\right) &= \psi_2(1)\psi_{12}\left(x_1, 1\right) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \\ m_{4\to3}\left(x_3\right) &= \psi_4(1)\psi_{34}\left(x_3, 1\right) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\ m_{5\to3}\left(x_3\right) &= \psi_5(0)\psi_{35}\left(x_3, 0\right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

Since x_3 is not observed we have

$$\begin{aligned} m_{3\to1}\left(x_1\right) &= \sum_{x_3} \psi_3\left(x_3\right)\psi_{13}\left(x_1, x_3\right) m_{4\to3}\left(x_3\right) m_{5\to3}\left(x_3\right) \\ &= \psi_3\left(0\right)\psi_{13}\left(x_1, 0\right) m_{4\to3}\left(0\right) m_{5\to3}\left(0\right) + \psi_3\left(1\right)\psi_{13}\left(x_1, 1\right) m_{4\to3}\left(1\right) m_{5\to3}\left(1\right) \\ &= \begin{bmatrix} 4 \\ 5 \end{bmatrix} \end{aligned}$$

From this we get

$$b\left(x_1\right) = p\left(x_1 \mid \bar{x}_2 = 1, \bar{x}_4 = 1, \bar{x}_5 = 0\right) \propto \psi_1\left(x_1\right) m_{2\to1}\left(x_1\right) m_{3\to1}\left(x_1\right) = \begin{bmatrix} 8 \\ 5 \end{bmatrix}$$

and so $p\left(x_1 = 1 \mid \bar{x}_2 = 1, \bar{x}_4 = 1, \bar{x}_5 = 0\right) = \frac{5}{13}$.
To compute $b\left(x_3\right) = p\left(x_3 \mid \bar{x}_2 = 1, \bar{x}_4 = 1, \bar{x}_5 = 0\right)$ we need to compute the message $m_{1\to3}$

$$m_{1\to3}\left(x_3\right) = \sum_{x_1} \psi_1\left(x_1\right)\psi_{13}\left(x_1, x_3\right) m_{2\to1}\left(x_1\right) = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

This gives

$$b\left(x_3\right) \propto \psi_3\left(x_1\right) m_{1\to3}\left(x_3\right) m_{4\to3}\left(x_3\right) m_{5\to3}\left(x_3\right) = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$

giving that $p\left(x_3 = 1 \mid \bar{x}_2 = 1, \bar{x}_4 = 1, \bar{x}_5 = 0\right) = \frac{8}{13}$.

10.5 Derivations 2

Unbiaseness

If the vectors $\left\{x^{(r)}\right\}_{r=1}^{R}$ are generated independently from $p(x)$, then the expectation of $\hat{\phi}$ is Φ. Indeed,

$$\begin{aligned}\mathbb{E}[\hat{\Phi}] &= \mathbb{E}\left[\frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\right] = \frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\left[\phi\left(x^{(r)}\right)\right] \\ &= \frac{1}{R}\sum_{r=1}^{R}\mathop{\mathbb{E}}_{x\sim p(x)}[\phi(x)] = \frac{R}{R}\mathop{\mathbb{E}}_{x\sim p(x)}[\phi(x)] \\ &= \Phi\end{aligned}$$

Variance

As the number of samples of R increases, the variance of $\hat{\phi}$ will decrease with rate $\frac{1}{R}$

$$\begin{aligned}\operatorname{var}[\hat{\phi}] &= \operatorname{var}\left[\frac{1}{R}\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\right] = \frac{1}{R^2}\operatorname{var}\left[\sum_{r=1}^{R}\phi\left(x^{(r)}\right)\right] \\ &= \frac{1}{R^2}\sum_{r=1}^{R}\operatorname{var}\left[\phi\left(x^{(r)}\right)\right] = \frac{R}{R^2}\operatorname{var}[\phi(x)] = \frac{1}{R}\operatorname{var}[\phi(x)]\end{aligned}$$

10.6 Derivations 3

WTS: $\mathbb{P}_{x\sim q}(x \in A \mid u \leq \tilde{p}(x)) = \mathbb{P}_{x\sim p}(x \in A)$
Recall:

- Note: $\mathbb{P}(u \leq \tilde{p}(x) \mid x) = \frac{\tilde{p}(x)}{c\tilde{q}(x)}$(remember we assume $\tilde{p}(x) < x\tilde{q}(x)$).
- $\forall A \subseteq \mathcal{X}: \quad \mathbb{P}_{x\sim p}(x \in A) = \int_A p(x)dx = \int \mathbf{1}_{\{x\in A\}}p(x)dx = \mathbb{E}_{x\sim p}\left[\mathbf{1}_{\{x\in A\}}\right]$
- Law of total expectation $\mathbb{E}[\mathbb{E}[Z \mid \mathcal{H}]] = \mathbb{E}Z$

We then have:

$$\begin{aligned}\mathbb{P}_{x\sim q}(x \in A \mid u \leq \tilde{p}(x)) &= \mathbb{P}_{x\sim q}(x \in A, u \leq \tilde{p}(x))/\mathbb{E}_{x\sim q}[\mathbb{P}(u \leq \tilde{p}(x) \mid x)] \\ &= \mathbb{E}_{x\sim q}\left[\mathbf{1}_{\{x\in A\}}\mathbb{P}(u \leq \tilde{p}(x) \mid x)\right]/\mathbb{E}_{x\sim q}\left[\frac{\tilde{p}(x)}{c\tilde{q}(x)}\right] \\ &= \mathbb{E}_{x\sim q}\left[\mathbf{1}_{\{x\in A\}}\frac{\tilde{p}(x)}{c\tilde{q}(x)}\right]/\frac{Z_p}{cZ_q} = \mathbb{P}_{x\sim p}(x \in A)\frac{Z_p}{cZ_q}/\frac{Z_p}{cZ_q} \\ &= \mathbb{P}_{x\sim p}(x \in A)\end{aligned}$$

10.7 Example 4

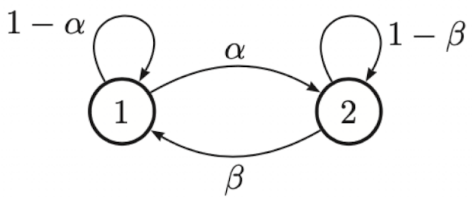


Figure 10.1: An example of transition matrix

The transition matrix is:

$$A = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$$

10.8 Example 4

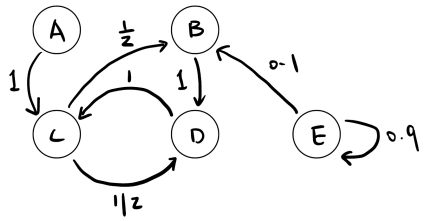


Figure 10.2: An example of stationary distribution

The transition matrix for the above markov chain is:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0.9 \end{bmatrix}$$

Notice that the stationary distribution $\pi = [\pi_A, \pi_B, \pi_C, \pi_D, \pi_E]$ The stationary distribution π is computed as follows:

1. π_A must be 0 as it is not possible that we are at state A at the next step. However, we may stay on A for the current step.

2. The obly way to be at state E at next step is that we are already at state E at the current step.
$$0.9\pi_E = \pi_E \Rightarrow \pi_E = 0$$

3. To stay at state B at the next step, we can stay at state C or E at the current step:
$$0.5\pi_C + 0.1\pi_E = \pi_B \Rightarrow \pi_C = 2\pi_B$$

4. Similarly, to stay at state C at the next step, we can stay at A or D.

$$1\pi_A + 1\pi_D = \pi_C$$

5. Notice that $\sum_i \pi_i = 1$

$$\pi = [0, 0.2, 0.4, 0.4, 0]$$

Equivalently:

$$A^T\pi = \pi$$
$$\pi = [0, 0.2, 0.4, 0.4, 0]$$

This result tells us that it is 0, 20, 40, 40 and 0 percent of staying at the five states at the current step. This also holds for the next, next next step etc.

10.9 Derivations 4

Recall $A(x' \mid x) = \min\left\{1, \frac{\tilde{p}(x')q(x||')}{\tilde{p}(x)q(x'|x)}\right\} = \min\left\{1, \frac{p(x')q(x|x')}{p(x)q(x'|x)}\right\}$. The resulting Markov chain has the following transition probabilities:

$$r(x' \mid x) = \begin{cases} q(x' \mid x) A(x' \mid x) & \text{if } x' \neq x \\ q(x \mid x) + \sum_{x' \neq x} q(x' \mid x)(1 - A(x' \mid x)) & \text{if } x' = x \end{cases}$$

WTS: Detailed Balance: $r(x' \mid x)\, p(x) = r(x \mid x')\, p(x')$. If $x \neq x'$

$$\begin{aligned} r(x' \mid x)\, p(x) &= p(x) q(x' \mid x) \min\left\{1, \frac{p(x')\, q(x \mid x')}{p(x) q(x' \mid x)}\right\} \\ &= \min\left\{p(x')\, q(x \mid x'), p(x) q(x' \mid x)\right\} \\ r(x \mid x')\, p(x') &= p(x')\, q(x \mid x') \min\left\{1, \frac{p(x) q(x' \mid x)}{p(x')\, q(x \mid x')}\right\} \\ &= \min\left\{p(x')\, q(x \mid x'), p(x) q(x' \mid x)\right\} \end{aligned}$$

Thus p is a stationary distribution of this Markov chain.