Markov Random Fields

Tutorial for KTH PGM DD2420

1 Introduction and Overview

Probabilistic graphical models can be divided into two groups of models, namely *directed* and *undirected* graphical models. Directed graphical models, or Bayesian Networks, are representations of a probability distribution, where conditionally dependence is defined as directed edges between random variables. In undirected graphical models, the direct dependencies between the variables cannot be naturally described, so instead we identify which variables that are dependent and then define the strength of their interactions[3]. These models can also be referred to as Markov Random Fields (MRFs). This tutorial gives a brief introduction to MRFs and their properties. We will also outline how to perform inference using *message passing* in MRFs and demonstrate how to perform approximate inference in an image denoising task with *loopy belief propagation*.

Exercise 1.1. As a motivation for using undirected graphical models, let us revisit the directed graphical model to the right from the Misconception example in [3] and discuss an example of its independence properties. Four students Alice, Bob, Charles, and Debbie are working in pairs on a homework assignment, where the following pairs meet: Alice and Bob (A, B), Bob and Charles (B, C), Charles and Debbie (C, D), and Debbie and Alice (D, A). There is a possible misconception about how to solve the assignment which each of the students might have figured out what it is. Each student may then influence their study partners with the correct understanding of the assignment or the misconception.

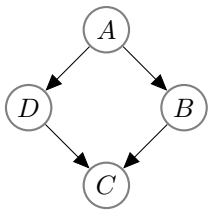


Fig. 1: Adapted from [3].

As Alice and Charles never speak to each other directly, we have that Alice and Charles opinions are conditionally independent given that Bob and Debbie are fully certain about their view of the problem, i.e., $A \perp\!\!\!\perp C \,|\, \{B, D\}$. Similarly, we have that $B \perp\!\!\!\perp D \,|\, \{A, C\}$. Can these independence conditions be modeled with the directed graphical model in Figure 1?

1.1 Markov Random Fields

We could see in the exercise above that the desired independence properties for the Misconception example are impossible to represent with the directed graph in Figure 1. In this case, the interactions between the students are symmetric rather than directed, which is why we would like to have a model that can represent the affinities between variables without any specific directions of the influences. Undirected graphical models, or Markov Random Fields (MRFs), implement this idea by letting the edges between the nodes correspond to interactions between neighboring variables.

The affinity between related variables are represented by a (often) non-negative function called a *factor*. To give an example, suppose that we want to represent the fact that Alice and Bob are more likely to agree than disagree. The factor $\phi(A, B)$ defining the interactions between A and B could then be defined as

$$\phi(A, B) = \begin{cases} 30 & \text{if } A = B = 1 \\ 10 & \text{if } A = B = 0 \\ 1 & \text{otherwise,} \end{cases}$$

which indicates that Alice and Bob are very likely to agree with each other.

Now when we have represented local interactions between related variables using factors, we also would like to model the possible configurations of values that can occur in the graph with a joint

probability distribution. In MRFs, this is done by multiplying all factors in the graph as

$$\tilde{p}(A, B, C, D) = \phi(A, B)\phi(B, C)\phi(C, D)\phi(D, A).$$

However, since there are no guarantees that this joint is a legal distribution it is normalized as

$$p(A, B, C, D) = \frac{1}{Z}\tilde{p}(A, B, C, D),$$

where the normalizing constant $Z = \sum_{A,B,C,D} \tilde{p}(A, B, C, D)$ is referred to as the *partition function*. The joint distribution can now be used to answer queries about different configurations in the graph by calculating marginals and conditional probabilities.

The benefit of this representation is that we can modify the entries in a factor without worrying about the normalization constraints or interactions of the other factors. However, the disadvantage is that the effects of changes in the undirected graph are not always intuitively understandable. It is important to note that the undirected graph identifies the dependence of variables and the factors define the strength of their interactions rather than corresponding to marginal probabilities. We quote Koller and Friedman [3] to bring more intuition to this fact: *"A factor is only one contribution to the overall distribution. The distribution as a whole has to take into consideration the contributions from all of the factors involved."* See Example 4.2 in [3] (pg. 108) for an example of how each factor in the graph influences the marginal distribution $p(A, B)$ in the Misconception example.

A formal definition of a Markov Random Field is a probability distribution over variables $x_1, ..., x_n$, where each x_i correspond to a node in an undirected graph [4]. The distribution p is defined as

$$p(x_1, ..., x_n) = \frac{1}{Z}\prod_{c\in\mathcal{C}} \phi_c(x_c),$$

where $\mathcal{C}$ denotes the set of cliques in the graph, and where the factor ϕ_c is a non-negative function over all variables in a clique c. The normalizing constant, or *partition function*, is defined as

$$Z = \sum_{x_1,...,x_n} \prod_{c\in\mathcal{C}} \phi_c(x_c),$$

where the sum is over all possible settings of each variable $x_1, ..., x_n$. Note that the distribution p may contain factors over many different types of cliques, e.g., single nodes, edges, and triangles. As in the Misconception example above, we chose to define the factors over edges without specifying unary factors over single nodes.

1.2 Independencies in Markov Random Fields

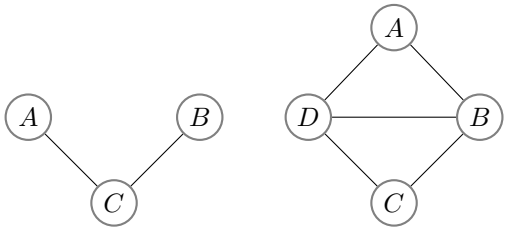
In directed graphs, we could check conditional independencies between variables using *d-separation*, which involved testing if the paths connecting two sets of nodes were blocked. However, this definition of paths being blocked becomes somewhat vague whenever a *v-structure* of directionalities, e.g. $A \rightarrow C \leftarrow B$, is present in the graph. Since the parent-child node relation is removed in undirected graphs, the definition of what independencies that can be described become much more simple: the variables A and B are dependent if they are connected by a path with unobserved variables. For example, suppose there is an undirected graph $A - C - B$, then the nodes A and B are dependent when C is unobserved. If C is observed, then the path between A and B will be blocked and thus they will be independent, i.e. $A \perp\!\!\!\perp B|C$.

The *Markov blanket* $\text{MB}(X)$ for a node X is defined as the minimal set of nodes that should be observed to make X independent of all other nodes in the graph. This definition holds for both directed and undirected graphs, but takes a simpler form in undirected graphs because a node is independent of all other nodes conditioned only on its neighboring nodes.

Exercise 1.2 (Cliques in MRFs).

a) What is the definition of a clique in a graphical model?

b) What are the cliques in graph $\mathcal{G}_1$ and $\mathcal{G}_2$?

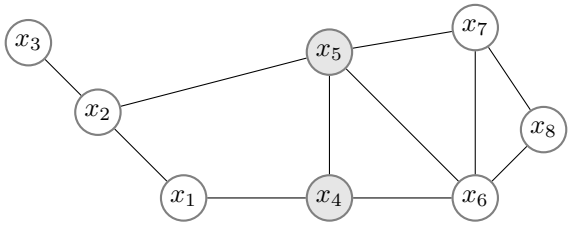


$(\mathcal{G}_1)$ $(\mathcal{G}_2)$

Exercise 1.3 (Independencies).

a) Which nodes (or set of nodes) are independent in graph $\mathcal{G}_3$ when x_4 and x_5 are observed?

b) What is the Markov blanket for node x_2 and node x_6 in graph $\mathcal{G}_3$?

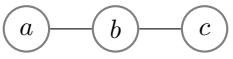


$(\mathcal{G}_3)$

1.3 Message Passing in MRFs

Given a probabilistic graphical model (PGM), we are interested in performing *inference* where we answer queries about the possible configurations in the graph by using the joint distribution. Often, we focus on two types of queries, namely computing marginals of some variable in the graph after all other variables have been summed out, and computing posterior probabilities of a variable given evidence of an event happening in other variables in the graph. Message passing, or *belief propagation*, is an algorithm for answering such queries in PGMs. The main idea of message passing is that sent messages include information about the sender node's belief of which state the receiver node is in. Next, we will use message passing for computing posterior probabilities.

Example. Consider the 3-node chain with pairs (a, b) and (b, c) with the joint distribution is given by $p(a, b, c) = Z^{-1}\phi(a, b)\phi(b, c)$. We are interested in the posterior $p(b|c)$, which is given by $p(b|c) = p(b, c)/p(c)$ using Bayes' rule.



We begin by computing the marginal $p(c)$ and we do so by summing out variables a and b:

$$p(c) = \sum_a \sum_b p(a, b, c) = Z^{-1} \sum_a \sum_b \phi(a, b)\phi(b, c) = Z^{-1} \sum_b \phi(b, c) \underbrace{\left(\sum_a \phi(a, b) \right)}_{\mu_a(b)}$$

$$= Z^{-1} \underbrace{\left(\sum_b \phi(b, c)\mu_a(b) \right)}_{\mu_b(c)} = Z^{-1}\mu_b(c),$$

where $\mu_b(c)$ denotes the incoming message to node c from b. Hence, a marginal of a node variable is the sum of incoming messages to the node, and an outgoing message is obtained by multiplying the incoming message by the local factor and summing out the node variable. Similarly as with the marginal, we can compute the joint $p(b, c)$ by only summing out a:

$$p(b, c) = \sum_a p(a, b, c) = Z^{-1} \sum_a \phi(a, b)\phi(b, c) = Z^{-1}\phi(b, c) \underbrace{\left(\sum_a \phi(a, b) \right)}_{\mu_a(b)} = Z^{-1}\phi(b, c)\mu_a(b).$$

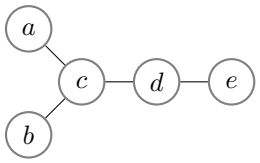
The posterior $p(b, c)$ is then given by

$$p(b|c) = \frac{p(b,c)}{p(c)} = \frac{\phi(b,c)\mu_a(b)}{\mu_b(c)}.$$

Note that we could reuse the message $\mu_a(b)$ when computing $p(b, c)$ which is one of the main benefits with message passing.

Exercise 1.4 (Pen-and-Paper). How do we compute $p(a|c)$ in the 3-node graph from the example above?

Exercise 1.5 (Pen-and-Paper). Now we consider a graph that has a slightly more challenging structure. How do we compute the posterior probabilities **a)** $p(d|c)$ and **b)** $p(a|d)$?



2 Pairwise MRFs for Images

Undirected graphical models have been applied extensively in image processing and computer vision [2]. One reason for their popularity in those fields is the general assumption that there exist spatial dependencies between pixels saying that neighboring pixels are likely to be similar. To this end, we can use a variant of MRFs called *pairwise MRFs* that are suitable for modeling such structure in images. Figure 3 shows one type of pairwise MRF with two kinds of nodes, namely the observations $\mathbf{y} = \{y_1, ..., y_4\}$ and the hidden states $\mathbf{x} = \{x_1, ..., x_4\}$. The observations usually correspond to the pixel intensities, while what the hidden states represent is more dependent on the task. For example, in image segmentation, the hidden state corresponds to a discrete label for some entity that the user wishes to localize in the image. As in the image denoising exercise below, the hidden state corresponds to a binary pixel value that is believed to be the original pixel value in an image corrupted by noise.

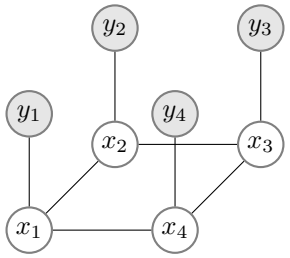


Fig. 3: Pairwise MRF.

2.1 Application: Image Denoising

A simple and common illustration of an application for undirected graphs is image denoising [1]. Let the observed noisy image $\mathbf{y}$ be described by the binary pixel values $y_i \in \{0, 1\}$, where $i = 1, \ldots, N$ runs over all N pixels. The noisy image is obtained by taking an unknown noise-free image $\mathbf{x}$ with binary pixel values $x_i \in \{0, 1\}$ and randomly flipping each pixel value by a probability θ. Our goal is then to recover the original noise-free image.

We will treat the observed noisy pixels $\mathbf{y}$ and the hidden pixel values $\mathbf{x}$ as variable nodes in the undirected graph. Moreover, we need to specify the dependencies between the nodes with factors. We know that there is a correlation between the noisy pixel y_i and x_i due to the small noise level, and that neighboring pixels in images are strongly correlated in general. This prior knowledge can be represented by a pairwise MRF similar to the one shown in Figure 3. This graph has two types of pairwise cliques, namely $\{x_i, y_i\}$ and $\{x_i, x_j\}$, and we will choose simple functions for both cliques when defining the factors. Since we choose non-negative factors, it is convenient to express the factors as exponentials. For both cliques, we choose the factors

$$\phi(x_i, y_i) = \exp\{-\tau\,|x_i - y_i|\} \quad \text{and} \quad \phi(x_i, x_j) = \exp\{-\lambda\,[x_i \neq x_j]\},$$

where τ and λ are positive constants. The factor $\phi(x_i, x_j)$ is called the Potts model where $[x_i \neq x_j] = 1$ when x_i and x_j have different values.

Now when we have specified the factors in the graph, we define an energy function $E(\mathbf{x}, \mathbf{y})$ that we should minimize to recover the original binary image. The energy function is derived from the joint distribution as

$$p(\mathbf{x}, \mathbf{y}) = Z^{-1} \exp\{-E(\mathbf{x}, \mathbf{y})\},$$

where $E(\mathbf{x}, \mathbf{y}) = \sum_i \tau|x_i - y_i| + \sum_{i,j} \lambda[x_i \neq x_j]$. Minimizing the energy $E(\mathbf{x}, \mathbf{y})$ will maximize the joint $p(\mathbf{x}, \mathbf{y})$ without having the compute the partition function Z since it is constant. There are many different options for minimizing $E(\mathbf{x}, \mathbf{y})$, and we will use *loopy belief propagation* for image denoising in the next part.

Exercise 2.1. Complete the full derivation of the energy function $E(\mathbf{x}, \mathbf{y}) = \sum_i \tau|x_i - y_i| + \sum_{i,j} \lambda[x_i \neq x_j]$ from the joint distribution $p(\mathbf{x}, \mathbf{y}) = Z^{-1} \exp\{-E(\mathbf{x}, \mathbf{y})\}$ by using the factors given in the image denoising problem above.

2.2 Loopy Belief Propagation

Inference can quickly become a computationally heavy task in graphical models representing images. For example, a binary image with 100 pixels will have $2^{100} \approx 1.27 \cdot 10^{30}$ different combinations of possible images $\mathbf{x}$. Furthermore, the graphs for images often contain loops such that exact inference cannot be guaranteed. To this end, we can use approximate inference algorithms such as Loopy Belief Propagation (LBP) for solving the image denoising task. LBP is a message passing algorithm suitable for performing inference in pairwise MRFs. We will briefly outline the steps of LBP here before moving to the coding exercise of the image denoising task.

Let x_i and x_j be two neighboring binary variable nodes. The variables can be assigned with the labels $l \in \{0, 1\}$ which correspond to the recovered pixel values. We will perform message passing in the graph to update each node's belief of their binary label l. A message $\mu_{i \to j}(l)$ represents node x_i's belief about node x_j being assigned with label l. Before x_i can pass its message to x_j, x_i has to wait until it has received the messages from all its neighbors except x_j. This procedure is performed for a fixed number of steps or until convergence (when the messages are unchanged) given some ordering of directions to pass the messages. For updating the messages, we will the *sum-product algorithm* which uses the following steps:

LBP with the Sum-Product Algorithm:

1. Initialize all messages to 1, such that $\mu_{i \to j}^{(0)}(l) = 1 \, \forall \, i, j, l$.
2. Repeat for $t = 1 : T$ steps
   (a) Message update summing over all labels l':

$$\mu_{i \to j}^{(t)}(l) = \sum_{l' \in \{0,1\}} \psi_1(x_i = l', y_i) \, \psi_2(x_i = l', x_j = l) \prod_{k \in N(i) \setminus j} \mu_{k \to i}^{(t-1)}(l'),$$

   where $\psi_1(x_i, y_i) = \tau \, |x_i - y_i|$ and $\psi_2(x_i, x_j) = \lambda \, [x_i \neq x_j]$.

   (b) Normalize messages (useful for numerical stability):

$$\mu_{i \to j}^{(t)}(l) = \frac{\mu_{i \to j}^{(t)}(l)}{\sum_{l'} \mu_{i \to j}^{(t)}(l')}$$

   (c) Belief update (estimating marginal probabilities):

$$\beta(x_i = l) = \psi_1(x_i = l, y_i) \prod_{k \in N(i)} \mu_{k \to i}^{(t)}(l)$$

   (d) Get the recovered image $\hat{\mathbf{x}}^{(t)}$ where $\hat{x}_i = \arg\max_{l \in \{0,1\}} \beta(x_i = l)$

   (e) Compute energy function using $\hat{\mathbf{x}}^{(t)}$ to check convergence:

$$E(\mathbf{x}, \mathbf{y}) = \tau \sum_i |y_i - \hat{x}_i| + \lambda \sum_{i,j} [\hat{x}_i \neq \hat{x}_j],$$

3. Return the recovered image $\hat{\mathbf{x}}^{(T)}$.

(a) (b) (c)

Fig. 4: (a) Original image. (b) Noisy image using probability $\theta = 0.2$ for flipping pixel value. (c) Recovered binary image obtained using LBP with sum-product algorithm.

Exercise 2.2 (Coding). We will apply LBP to the image denoising task using code written in Python. We have provided the required packages to run the code in the file `requirements.txt`, and alternatively a Conda environment in `environment.yml`. See `README.md` for help with the installation. We will apply LBP to the noisy image in Figure 4b with the goal to recover the original image seen in Figure 4a. Note that LBP managed to fully recover the hole in the letter 'A' in Figure 4c which can be considered a satisfactory result for this image.

Finish the tasks in the following order:

a) Go through the code in `main.py` and `sum_product.py` and make sure you understand all parts.

b) Search for suitable values in the range $[1, 10]$ for the constants τ and λ, e.g., by implmenting a grid search over the given range. What values of τ and λ gives satisfactory results on the recovered image? How many number of iterations were needed?

c) Comment on how the values of τ and λ affect the final result.

3 Summary

In this tutorial, we gave a brief introduction to MRFs and their properties as well as a recap of how to perform inference in undirected graphs using *message passing*. Furthermore, we showed an example of how to implement a pairwise MRF for an image denoising task and how to perform approximate inference in such a model with loopy belief propagation. We hope that this tutorial will encourage the reader to study other kinds of MRFs, such as Conditional Random fields and Factor Graphs, or gain interest in other ways to perform approximate inference e.g. MCMC, as well as looking into how to perform parameter learning in PGMs.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov Random Fields for Vision and Image Processing*. The MIT Press, 2011.

[3] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[4] Volodymyr Kuleshov and Stefano Ermon. Introductory course to probabilistic graphical models. `https://ermongroup.github.io/cs228-notes/`, 2018.