Undirected Graphical Model Application

Aryan Arbabi

CSC 412 Tutorial

February 1, 2018
Outline

Example - Image Denoising
  Formulation
  Inference
  Learning
Undirected Graphical Model

- Also called Markov Random Field (MRF) or Markov networks.
- Nodes in the graph represent variables, edges represent probabilistic interactions.
- Examples:
  - Chain model for NLP problems.
  - Grid model for computer vision problems.
Parameterization

\( \mathbf{x} = (x_1, \ldots, x_m) \), a vector of random variables
\( \mathcal{C} \), set of cliques in the graph
\( \mathbf{x}_c \) is the subvector of \( \mathbf{x} \) restricted to clique \( c \)
\( \theta \), model parameters

- **Product of Factors**

\[
p_\theta(\mathbf{x}) = \frac{1}{Z(\theta)} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c | \theta_c)
\]

- **Gibbs distribution, sum of potentials**

\[
p_\theta(\mathbf{x}) = \frac{1}{Z(\theta)} \exp \left( \sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c | \theta_c) \right)
\]

- **Log-linear model**

\[
p_\theta(\mathbf{x}) = \frac{1}{Z(\theta)} \exp \left( \sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c) ^\top \theta_c \right)
\]
Partition Function

\[ Z(\theta) = \sum_{x} \exp \left( \sum_{c \in C} \phi_c(x_c | \theta_c) \right) \]

- This is usually hard to compute as the sum over all possible \( x \) is a sum over an exponentially large space.
- This makes inference and learning in undirected graphical models challenging.
A Simple Image Denoising Example

Observe as input a noisy image $x$

Want to predict a clean image $y$

$\mathbf{x} = (x_1, ..., x_m)$ is the observed noisy image, each pixel $x_i \in \{-1, +1\}$. $\mathbf{y} = (y_1, ..., y_m)$ is the output, each pixel $y_i \in \{-1, +1\}$.

We can model the conditional distribution $p(y|x)$ as a grid-structured MRF for $y$. 
Model Specification

\[ p(y|x) = \frac{1}{Z} \exp \left( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \right) \]

- Very similar to an Ising model on \( y \), except that we are modeling the conditional distribution.
- \( \alpha, \beta, \gamma \) are model parameters.
- The higher \( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \) is, the more likely \( y \) is for the given \( x \).
Model Specification

\[ p(y|x) = \frac{1}{Z} \exp \left( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \right) \]

- \( \alpha \sum_i y_i \) represents the ‘prior’ for each pixel to be +1. Larger \( \alpha \) encourages more pixels to be +1.
- \( \beta \sum_{i,j} y_i y_j \) encourages smoothness when \( \beta > 0 \). If neighboring pixels \( i \) and \( j \) take the same output then \( y_i y_j = +1 \) otherwise the product is -1.
- \( \gamma \sum_i x_i y_i \) encourages the output to be the same as the input when \( \gamma > 0 \), we believe only a small part of the input data is corrupted.
Making Predictions

Given a noisy input image \( x \), we want to predict what the corresponding clean image \( y \) is.

- We may want to find the most likely \( y \) under our model \( p(y|x) \), this is called **MAP inference**.
- We may want to get a few candidate \( y \) from our model by sampling from \( p(y|x) \).
- We may want to find representative candidates, a set of \( y \) that has high likelihood as well as diversity.
- More...
MAP Inference

\[
\mathbf{y}^* = \arg\max_{\mathbf{y}} \frac{1}{Z} \exp \left( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \right)
\]

= \arg\max_{\mathbf{y}} \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i

- As \( \mathbf{y} \in \{-1, +1\}^m \), this is a combinatorial optimization problem. In many cases it is (NP-)hard to find the exact optimal solution.

- Approximate solutions are acceptable.
Iterated Conditional Modes

Idea: instead of finding the best configuration of all variables $y_1, ..., y_m$ jointly, optimize one single variable at a time and iterate through all variables until convergence.

- Optimizing a single variable is much easier than optimizing a large set of variables jointly - usually we can find the exact optimum for a single variable.

- For each $j$, we hold $y_1, ..., y_{i-1}, y_{i+1}, ..., y_m$ fixed and find

$$y_j^* = \arg\max_{y_j \in \{-1, +1\}} \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i$$

$$= \arg\max_{y_j \in \{-1, +1\}} \alpha y_j + \beta \sum_{i \in \mathcal{N}(j)} y_i y_j + \gamma x_j y_j$$

$$= \text{sign} \left( \alpha + \beta \sum_{i \in \mathcal{N}(j)} y_i + \gamma x_j \right)$$
Results

Inference with Iterated Conditional Modes,
\(\alpha = 0.1, \beta = 0.5, \gamma = 0.5\)
Find the Best Parameter Setting

Different parameter settings result in different models

\[ \alpha = 0.1, \gamma = 0.5 \]

\[ \beta = 0.1 \]

\[ \beta = 0.2 \]

\[ \beta = 0.5 \]

How to choose the best parameter setting?

- Manually tune the parameters?
The Learning Approach

When the number of parameters becomes large, it is infeasible to tune them by hand.

Instead we can use a data set of training examples to learn the optimal parameter setting automatically.

- Collect a set of training examples - pairs of \((x^{(n)}, y^{(n)})\)
- Formulate an objective function that evaluates how well our model is doing on this training set
- Optimize this objective to get the optimal parameter setting

This objective function is usually called a loss function (and we want to minimize it).
Maximum Likelihood

Maximize the log-likelihood, or minimize the negative log-likelihood of data

▶ So that the true output $y^{(n)}$ will have high probability under our model for $x^{(n)}$.

$$L = -\frac{1}{N} \sum_n \log p(y^{(n)}|x^{(n)})$$

▶ $L$ is a function of model parameters $\alpha$, $\beta$ and $\gamma$

$$L = -\frac{1}{N} \sum_n \left[ \left( \alpha \sum_i y_i^{(n)} + \beta \sum_{i,j} y_i^{(n)} y_j^{(n)} + \gamma \sum_i y_i^{(n)} x_i^{(n)} \right) \right. \\
- \log \sum_y \exp \left( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i y_i x_i^{(n)} \right) \right]$$
Maximum Likelihood

Minimize $L$ using gradient-based methods. For example for $\beta$

\[
\frac{\partial L}{\partial \beta} = -\frac{1}{N} \sum_n \left[ \sum_{i,j} y_i^{(n)} y_j^{(n)} - \frac{\sum_y \exp(\ldots) \sum_{i,j} y_i y_j}{\sum_y \exp(\ldots)} \right]
\]

\[
= -\frac{1}{N} \sum_n \left[ \sum_{i,j} y_i^{(n)} y_j^{(n)} - \sum_y p(y|x^{(n)}) \sum_{i,j} y_i y_j \right]
\]

\[
= -\frac{1}{N} \sum_n \left[ \sum_{i,j} y_i^{(n)} y_j^{(n)} - \sum_{i,j} \mathbb{E}_{p(y|x^{(n)})}[y_i y_j] \right]
\]

$\mathbb{E}_{p(y|x^{(n)})}[y_i y_j]$ is usually hard to compute as it is a sum over exponentially many terms.

\[
\mathbb{E}_{p(y|x^{(n)})}[y_i y_j] = \sum_y p(y|x^{(n)}) y_i y_j
\]
Pseudolikelihood

- The partition function makes it hard to use exact gradient-based method.
- Pseudolikelihood avoids this problem by using an approximation to the exact likelihood function.

\[
p(y | x) = \prod_j p(y_j | y_1, \ldots, y_{j-1}, x)
\]

\[
\approx \prod_j p(y_j | y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_m, x) = \prod_j p(y_j | y_{-j}, x)
\]

- \(p(y_j | y_{-j}, x)\) does not have the partition function problem.

\[
p(y_j | y_{-j}, x) = \frac{1}{Z} \exp(...)
\]

\[
= \frac{\exp(...)}{\sum_{y_j} \frac{1}{Z} \exp(\ldots)} = \frac{\exp(...)}{\sum_{y_j} \exp(\ldots)}
\]

The denominator is a sum over a single variable, which is easy to compute.
Pseudolikelihood

For our denoising model,

\[ p(y_j|y_{-j}, x) = \frac{\exp \left( \left( \alpha + \beta \sum_{i \in N(j)} y_i + \gamma x_j \right) y_j \right)}{\sum_{y_j \in \{ -1, +1 \}} \exp \left( \left( \alpha + \beta \sum_{i \in N(j)} y_i + \gamma x_j \right) y_j \right)} \]
Pseudolikelihood

For our denoising model,

\[ p(y_j | y_{-j}, x) = \frac{\exp \left( \left( \alpha + \beta \sum_{i \in \mathcal{N}(j)} y_i + \gamma x_j \right) y_j \right)}{\sum_{y_j \in \{-1, +1\}} \exp \left( \left( \alpha + \beta \sum_{i \in \mathcal{N}(j)} y_i + \gamma x_j \right) y_j \right)} \]

Therefore

\[ L = -\frac{1}{N} \sum_n \log p(y^{(n)} | x^{(n)}) \approx -\frac{1}{N} \sum_n \sum_j \log p(y_j^{(n)} | y_{-j}^{(n)}, x^{(n)}) \]

\[ = -\frac{1}{N} \sum_n \sum_j \left[ \left( \alpha + \beta \sum_{i \in \mathcal{N}(j)} y_i^{(n)} + \gamma x_j^{(n)} \right) y_j^{(n)} \right] \]

\[- \log \sum_{y_j \in \{-1, +1\}} \exp \left( \left( \alpha + \beta \sum_{i \in \mathcal{N}(j)} y_i^{(n)} + \gamma x_j^{(n)} \right) y_j \right) \]
Pseudolikelihood

\[
\frac{\partial L}{\partial \beta} = -\frac{1}{N} \sum_n \left[ \sum_{i,j} y_i^{(n)} y_j^{(n)} - \sum_j \sum_{i \in \mathcal{N}(j)} y_i^{(n)} \mathbb{E}_{p(y_j|y_{-j}^{(n)},x^{(n)})} [y_j] \right]
\]

\[
= -\frac{1}{N} \sum_n \sum_j \sum_{i \in \mathcal{N}(j)} y_i^{(n)} \left[ y_j^{(n)} - \mathbb{E}_{p(y_j|y_{-j}^{(n)},x^{(n)})} [y_j] \right]
\]

The key term \( \mathbb{E}_{p(y_j|y_{-j}^{(n)},x^{(n)})} [y_j] \) is easy to compute as it is an expectation over a single variable.

Then follow the negative gradient to minimize \( L \).
Pseudolikelihood

> If the data is generated from a distribution in the defined form with some $\alpha^*, \beta^*, \gamma^*$, then as $N \to \infty$, the optimal solution of $\alpha, \beta, \gamma$ that maximizes the pseudolikelihood will be $\alpha^*, \beta^*, \gamma^*$.

> You can prove it yourself.
Comments

\[ p(y|x) = \frac{1}{Z} \exp \left( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \right) \]

- We can use different \( \alpha, \gamma \) parameters for different \( i \), different \( \beta \) parameters for different \( i, j \) pairs to make the model more powerful.
- We can define the potential functions to have more sophisticated form, for example the pairwise potential can be some function \( \phi(y_i, y_j) \) rather than just a product \( y_i y_j \).
- The same model can be used for semantic image segmentation, where the output are object class labels for all pixels.
\[ p(y|x) = \frac{1}{Z} \exp \left( \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \right) \]

- We will study more methods to do inference (compute MAP or expectation) in the future.
- There are also many other loss functions that can be used as the training objective.