Undirected Graphical Model Application

Aryan Arbabi

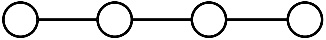
CSC 412 Tutorial

February 1, 2018

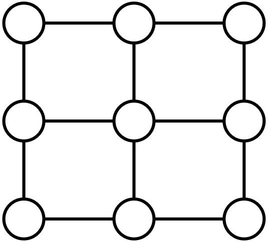
Outline

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

$\mathbf{x} = (x_1, ..., x_m)$, 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
θ, 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

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

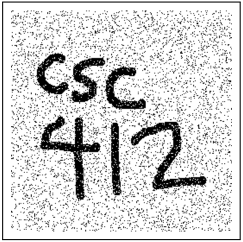
Partition Function

$$Z(\theta) = \sum_{\mathbf{x}} \exp\left(\sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c | \theta_c)\right)$$

- This is usually hard to compute as the sum over all possible $\mathbf{x}$ is a sum over an exponentially large space.
- This makes inference and learning in undirected graphical models challenging.

A Simple Image Denoising Example

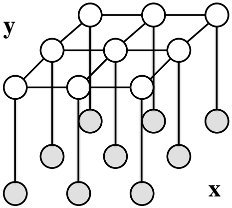
Observe as input
a noisy image $\mathbf{x}$



Want to predict
a clean image $\mathbf{y}$

- $\mathbf{x} = (x_1, ..., x_m)$ 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 $p(\mathbf{y}|\mathbf{x})$ as a grid-structured MRF for $\mathbf{y}$.

Model Specification



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

- Very similar to an Ising model on $\mathbf{y}$, except that we are modeling the conditional distribution.
- ~~α, β, γ 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 $\mathbf{y}$ is for the given $\mathbf{x}$.

Model Specification

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

- $\alpha \sum_i y_i$ represents the 'prior' for each pixel to be $+1$. Larger α 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 $\mathbf{x}$, we want to predict what the corresponding clean image $\mathbf{y}$ is.

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

MAP Inference

$$
\begin{aligned}
\mathbf{y}^* &= \underset{\mathbf{y}}{\operatorname{argmax}} \quad \frac{1}{Z} \exp\left(\alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i\right) \\
&= \underset{\mathbf{y}}{\operatorname{argmax}} \quad \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i
\end{aligned}
$$

- 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 $y_1, \ldots, y_m$ jointly, optimize one single variable at a time and iterate through all variables until convergence.

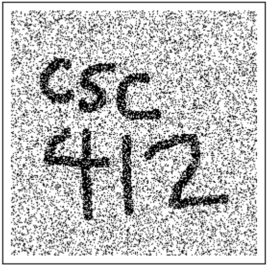
- Optimizing a single variable is much easier than optimizing a large set of varibles jointly - usually we can find the exact optimum for a single variable.
- For each j, we hold $y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_m$ fixed and find

$$
\begin{aligned}
y_j^* &= \underset{y_j \in \{-1,+1\}}{\operatorname{argmax}} \quad \alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i x_i y_i \\
&= \underset{y_j \in \{-1,+1\}}{\operatorname{argmax}} \quad \alpha y_j + \beta \sum_{i \in \mathcal{N}(j)} y_i y_j + \gamma x_j y_j \\
&= \operatorname{sign}\left(\alpha + \beta \sum_{i \in \mathcal{N}(j)} y_i + \gamma x_j \right)
\end{aligned}
$$

Results

Inference with Iterated Conditional Modes,
$\alpha = 0.1, \beta = 0.5, \gamma = 0.5$

Input



Output

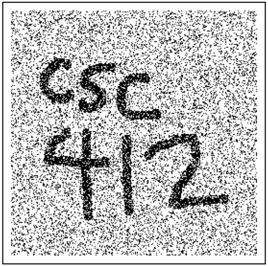
Ground-Truth

Find the Best Parameter Setting

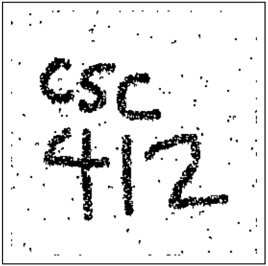
Different parameter settings result in different models

$\alpha = 0.1, \gamma = 0.5$

$\beta = 0.1$



$\beta = 0.2$



$\beta = 0.5$

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 $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$
- 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 $\mathbf{y}^{(n)}$ will have high probability under our model for $\mathbf{x}^{(n)}$.

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

- L is a function of model parameters α, β and γ

$$\begin{aligned} L \;\; = \;\; & -\frac{1}{N} \sum_n \left[\left(\alpha \sum_i y_i^{(n)} + \beta \sum_{i,j} y_i^{(n)} y_j^{(n)} + \gamma \sum_i y_i^{(n)} x_i^{(n)} \right) \right. \\ & \left. - \log \sum_{\mathbf{y}} \exp \left(\alpha \sum_i y_i + \beta \sum_{i,j} y_i y_j + \gamma \sum_i y_i x_i^{(n)} \right) \right] \end{aligned}$$

Maximum Likelihood

Minimize L using gradient-based methods. For example for β

$$
\begin{aligned}
\frac{\partial L}{\partial \beta} &= -\frac{1}{N}\sum_n \left[\sum_{i,j} y_i^{(n)} y_j^{(n)} - \frac{\sum_{\mathbf{y}} \exp(\ldots) \sum_{i,j} y_i y_j}{\sum_{\mathbf{y}} \exp(\ldots)} \right] \\
&= -\frac{1}{N}\sum_n \left[\sum_{i,j} y_i^{(n)} y_j^{(n)} - \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(n)}) \sum_{i,j} y_i y_j \right] \\
&= -\frac{1}{N}\sum_n \left[\sum_{i,j} y_i^{(n)} y_j^{(n)} - \sum_{i,j} \mathbb{E}_{p(\mathbf{y}|\mathbf{x}^{(n)})}[y_i y_j] \right]
\end{aligned}
$$

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

$$
\mathbb{E}_{p(\mathbf{y}|\mathbf{x}^{(n)})}[y_i y_j] = \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(n)}) y_i y_j
$$

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.

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x}) &= \prod_j p(y_j|y_1, ..., y_{j-1}, \mathbf{x}) \\
&\approx \prod_j p(y_j|y_1, ..., y_{j-1}, y_{j+1}, ..., y_m, \mathbf{x}) = \prod_j p(y_j|\mathbf{y}_{-j}, \mathbf{x})
\end{aligned}
$$

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

$$p(y_j|\mathbf{y}_{-j}, \mathbf{x}) = \frac{\frac{1}{Z}\exp(...)}{\sum_{y_j}\frac{1}{Z}\exp(...)} = \frac{\exp(...)}{\sum_{y_j}\exp(...)}$$

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

Pseudolikelihood

For our denoising model,

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

Pseudolikelihood

For our denoising model,

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

Therefore

$$\begin{aligned} L &= -\frac{1}{N}\sum_n \log p(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}) \approx -\frac{1}{N}\sum_n\sum_j \log p(y_j^{(n)}|\mathbf{y}_{-j}^{(n)},\mathbf{x}^{(n)}) \\ &= -\frac{1}{N}\sum_n\sum_j \left[\left(\alpha + \beta\sum_{i\in\mathcal{N}(j)} y_i^{(n)} + \gamma x_j^{(n)}\right) y_j^{(n)} \right. \\ &\quad \left. -\log\sum_{y_j\in\{-1,+1\}}\exp\left(\left(\alpha + \beta\sum_{i\in\mathcal{N}(j)} y_i^{(n)} + \gamma x_j^{(n)}\right) y_j\right)\right] \end{aligned}$$

Pseudolikelihood

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

The key term $\mathbb{E}_{p(y_j | \mathbf{y}_{-j}^{(n)}, \mathbf{x}^{(n)})}[y_j]$ 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 α, β, γ that maximizes the pseudolikelihood will be $\alpha^*, \beta^*, \gamma^*$.
- You can prove it yourself.

Comments

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

- We can use different α, γ parameters for different i, different β 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.

Comments

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

- 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.