Learning in MRFs and CRFs

Rahul G. Krishnan

New York University

Lab 12, Nov 20, 2016

Outline

Course Summary

- Represent objects in our world as discrete or continuous random variables
- Bayesian networks: Directed graphical model, joint distribution factorizes, causal view of the world
- Markov Random Fields: Undirected graphical model, positive potentials, partition function

Inference in Bayesian Networks

- Variational Inference:
  - Assume the solution lies within an approximate distribution
  - Solve an optimization problem to find the best parameters of that distribution
- Markov Chain Monte Carlo:
  - Metropolis Hastings (and Gibbs Sampling)
  - Hamiltonian Monte Carlo

Learning in Bayesian Networks

- Maximum likelihood estimation
- Fully observed Bayesian networks: The optimization problem decouples across the conditional probability distributions
- Bayesian networks with latent variables: The optimization problem decouples across the conditional probability distributions

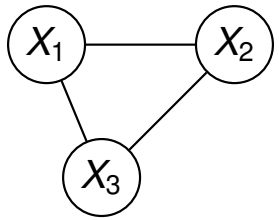
Inference in Undirected Graphical Models

- Belief Propagation : Exact on tree structured MRFs, approximate on loop graphs
- MCMC: Gibbs Sampling

Exponential Families: Recap

- $p(x;\eta) = \underbrace{h(x)}_{\text{Base Measure}} \exp(\underbrace{\eta}_{\text{Natural Parameters}} \cdot \underbrace{f(x)}_{\text{Sufficient Stat}} - \underbrace{\log Z(\eta)}_{\text{Log-Partition Function}})$
- Gradient of the Log Partition function = Marginals
- $\nabla_{\eta_i} \log Z(\eta) = \sum_x p(x) f_i(x) = \mathbb{E}_p[f_i(x)]$

Feature Functions



- We're going to show that for a choice of the feature function, $p(x;\theta)$ in an MRF lies in the exponential family
- Assume a binary pairwise MRF over three random variables
- $f(X)$ takes as input the assignment to each of X_1, X_2, X_3 and returns a vector
- $f(X)$ is of size $2 * 3 + 4 * 3 = 18$ (show on board)
- Indicator vectors denote events $[X_1 = 0, X_1 = 1, X_2 = 0, X_2 = 1 \ldots, (X_1 = 0, X_2 = 0), (X_1 = 0, X_2 = 1) \ldots]$

Parameterization

- $p(x;\theta) = \frac{1}{Z(\theta)} \prod_c \phi_c(x_c;\theta_c)$
- $\phi_c(x_c;\theta_c)$ is our potential table that selects a positive number based on the configuration to variables in clique x_c
- $\phi_c(x_c;\theta_c) = \exp(\theta_c \cdot f_c(x_c))$
- $f_c(x_c)$ takes an assignment to variables in the clique and returns a one-hot encoding of it (show example)
- θ_c is of the same size as $f_c(x_c)$
- This parameterization is fully general. No loss of information.

Parameterization

- MRFs lie in the exponential family

$$
\begin{aligned}
p(x;\theta) &= \frac{1}{\log Z(\theta)} \prod_c \phi_c(x_c;\theta_c) \\
&= \exp(-\log Z(\theta)) \prod_c \exp(\theta_c \cdot f_c(x_c)) \\
&= \exp(\sum_c \theta_c \cdot f_c(x_c) - \log Z(\theta)) \\
&= \exp(\theta \cdot f(x) - \log Z(\theta))
\end{aligned}
$$

- Denoting $\theta = \{\theta_c\}_c$ and $f(x)$ as returning a concatenation of vectors $f_c(x_c)$
- Parts of the vector θ can be shared across cliques!
- Will return to this idea when we talk about CRFs

Learning

- Two methods we'll skim over:
- Maximum Entropy Learning
  - Find the maximum entropy distribution that matches the emperical moments
  - Turns out that maximum entropy distribution lies in the exponential family
- Maximum Likelihood Estimation
- Learning with Pseudo-Likelihood
  - Use a surrogate loss function.
  - Instead of $\log p(x;\theta)$ use $\sum_{i=1}^{n} \log p(x_i|x_{N(i)};\hat{\theta})$
  - Tractable objective (small partition functions)

ML Learning

- Unlike Bayesian networks, likelihood in MRFs does not decompose across clique potentials (partition function)
- $\theta^{ML} = \arg\max_\theta \frac{1}{|\mathcal{D}|}\theta \cdot \sum_{x \in \mathcal{D}} f(x) - |\mathcal{D}| \log Z(\theta)$
- Gradient based learning typically needs two key ingredients:
  - A method for probablistic inference (gradients of $\log Z(\theta)$ are the marginals under the model)
  - A method that yeilds an upper bound on the log-partition function
- Once again we see inference lie within the inner loop of learning!

Introduction

- A CRF is a Markov random field on X, Y that specifies a conditional distribution
- $p(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \phi_c(y_c, x)$
- Normalization sums over latent y and is conditioned on x, $Z(x) = \sum_{\hat{y}} \prod_{c \in C} \phi_c(\hat{y}_c, x)$
- $\phi_c(x_c, y_c) = \exp(\theta_c \cdot f_c(x_c, y_c))$
- θ is learned
- Typically use parameter sharing. eg. in POS tagging, edge potentials are the same for all edges, same for node potentials

Marginal or MAP Inference

We can use any of the techniques we saw earlier to learn the parameters of the CRF, except:

- At test time, we only do MAP inference for which the partition function is irrelevant
- If we care about test time prediction, can we do learning to optimize this loss directly?
- We'll assume for the moment that we have some method to MAP inference in CRFs/MRFs

Learning w/ MAP Inference

- Key idea is to optimize the classification error directly
- We want prob. of true label under the model $p(y|x) \geq p(y'|x)$ for all other y'
- The emperical risk is zero when $\theta \cdot (f(x, y) - f(x, y')) > 0$ for all $y' \neq y$
- No log partition function
- Linear Program. Exponential constraint.

Structured Perceptron

- Simple algorithm that iteratively moves θ in a directly that encourages emperical risk to be zero
- There are exponentially many other possible y', which one do we choose? Use the MAP assignment given by y_{MAP}
- $\theta \leftarrow \theta + f(x^m, y^m) - f(x^m, y_{MAP})$

Relaxation 1

We can allow for slack in the constraints since we may not find a perfect θ

- This leads to the Structural SVM algorithm
- Minimizes the Hinge-Loss

Relaxation 2

Penalize errors differently

- This leads to the Structural SVM algorithm with margin scaling

Stochastic Subgradient

- An algorithm for solving the structural SVM with margin scaling
- The subgradient is $f(x^m, \hat{y}) - f(x^m, y^m)$.
- Contrast to learning with marginal inference

Code Sample

- Many open-source CRF packages available
- In ps7, you will use pystruct3:
  `https://pystruct.github.io/user_guide.html`
- Lets go through it to make sure the setup, format and everything else is clear

MAP Inference

- Rich area of research at the intersection of combinatorial optimization and machine learning
- Local Search: Iterated Conditional Modes
- Branch and Bound: Exhaustive search with pruning
- Dual Decomposition: Decompose MAP problem into simpler sub-problems and tie these subproblems together

Overview

- Available at `https://arxiv.org/abs/1503.02351`
- Asks the question, instead of learning linear feature functions, can we learn parametric non-linear ones?
- Can we parameterize the θ as a function of the original input?

Why would we want to do this?

- Best of both worlds, feature representation with deep learning and modeling structure in label spaces with CRFs
- Allows us to have patch-specific edge-potentials
- Simplify MAP estimation. (eg. if one part of the graph does is easy to predict for compared to another)

Overview of their algorithm

The authors use it for the task of image segmentation

- Use a deep network to predict local representation (feature functions)
- **Technical Tool:** Optimize a mean-field approximation to the conditional density parameterized by the CRF
- The final marginals used at test time
- Backpropagate through the entire procedure for learning

Take away: There is lots of research to be done in exploring the intersection of deep learning and CRFs.