Markov Random Fields

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA’ DI PISA

DAVIDE.BACCIU@UNIPI.IT
Markov Random Fields (MFRs)

- Undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (a.k.a. Markov Networks)

- Nodes $v \in \mathcal{V}$ represent random variables $X_v$
  - Shaded $\Rightarrow$ observed
  - Empty $\Rightarrow$ un-observed

- Edges $e \in \mathcal{E}$ describe bi-directional dependencies between variables (constraints)

Graph often coherent with data structure
Likelihood Factorization

Define $X = X_1, \ldots, X_N$ as the RVs associated to the $N$ nodes in the undirected graph $\mathcal{G}$

$$P(X) = \frac{1}{Z} \prod_C \psi_C(X_C)$$

- $X_C \rightarrow$ RV associated with nodes in the maximal clique $C$
- $\psi_C(X_C) \rightarrow$ potential function for clique $C$
- $Z \rightarrow$ partition function ensuring normalization

$$Z = \sum_X \prod_C \psi_C(X_C)$$
Potential Functions

- Potential functions $\psi_C(X_C)$ are not probabilities!
- Express which configurations of the local variables are preferred
- If we restrict to **strictly positive potential functions**, the Hammersley-Clifford theorem provides guarantees on the distribution that can be represented by the clique factorization

**Definition (Boltzmann distribution)**

A convenient and widely used strictly positive representation of the potential functions is

$$\psi_C(X_C) = \exp\{-E(X_C)\}$$

where $E(X_C)$ is the **energy function**
Factorizing Potential Functions

In general, we will assume to work with MRF where the partition functions factorize as

$$\psi_C(X_C) = \exp\left(\sum_k \theta_{Ck} f_{Ck}(X_C)\right)$$

where

- $f_{Ck}$ (or $f_k$) are feature functions or sufficient statistics to compute the potential of clique $C$
- $\theta_{Ck} \in \mathbb{R}$ are parameters
- $k$ indexes over the available feature functions

Undirected graphical models do not express the factorization of potentials into feature functions $\Rightarrow$ factor graphs
Factor Graphs

- RV are again circular nodes
- Factors $f_{ck}$ are denoted as square nodes
- Edges connect a factor to the RV

\[
\psi(X_1, X_2, X_3) = f(X_1, X_2, X_3) \\
\psi(X_1, X_2, X_3) = f_a(X_1, X_2, X_3) f_b(X_2, X_3)
\]
Sum-Product Inference

- A powerful class of exact inference algorithms (Belief Propagation)
- Use factor graph representation to provide a unique algorithm for directed/undirected models
- Inference is feasible for chain and tree structures
  - Forward-backward algorithm in HMMs
  - Computationally more impacting in MRF due to partition function
- Inference in general MRF
  - Restructure the graph to obtain a tree-like structure to perform message passing (junction tree algorithm, Chow-Liu)
  - Approximated inference (variational, sampling)

Constrain the MRF to obtain tractable classes of undirected models
Restricting to Conditional Probabilities

In ML a part of the random variables can be assumed to be always observable ⇒ input data
- $X_k$ - observable inputs in the factor $k$
- $Y_k$ - hidden (or partly observable) RV
- $f_k(X_k, Y_k)$ - factor feature function

Under this assumption we can directly model the conditional distribution

$$P(Y|X) = \frac{1}{Z(X)} \prod_k \exp\{\theta_k f_k(X_k, Y_k)\}$$

where $X$ is the joint input that is always available

$$Z(X) = \sum_y \prod_k \exp\{\theta_k f_k(X_k, Y_k = y_k)\}$$
Conditional Random Field (CRF)

Constrained MRF models representing input-conditional distributions

\[ P(Y|X, \theta) = \frac{1}{Z(X)} \exp(\theta_1 f_1(X_i, Y_i) + \theta_2 f_2(Y_i, Y_j) + \theta_3 f(X_j, Y_j) + \ldots) \]
Feature functions

What does a feature function $f_k(X_k, Y_k)$ do?

- Represent couplings or constraints between random variables
- Often very simple, such as linear functions

- Make noisy binary pixel $X_i$ and its clean version $Y_i$ have same sign
  $f_i(X_i, Y_i) = X_iY_i$

- Constrain nearby interpretations to be similar
  $f_{ij}(Y_i, Y_j) = Y_i^T Y_j$
Discriminative Learning in Graphical Models

\( \mathbf{X} \) is always observable input while \( \mathbf{Y} \) can be unobserved

- Let us simplify the problem by considering to have a single \( Y \) and multiple \( \mathbf{X} \)
- Let us assume that we can observe the \( Y^n \) corresponding to \( \mathbf{X}^n \) for some samples \( n \)
- We can use this information to fit \( \theta \) in \( P(Y|\mathbf{X}, \theta) \)
- What does \( P(Y|\mathbf{X'}, \theta) \) do for a new \( \mathbf{X'} \) sample without observable \( Y' \)? Performs a prediction (e.g. classification if \( Y \) is multinomial)

The model above describes the Logistic Regression/Classifier: a discriminative version of Naive Bayes
A CRF for Sequences

The undirected and discriminative equivalent of an HMM

Is this all about substituting emission probability with feature $f_e$ and transition distribution with $f_t$?
A Generalization of HMM

Modeling relative influence of suffix and prefix symbols

\[ P(Y|X, \theta) = \frac{1}{Z(X)} \prod_t \exp\{\theta_p f_p(X_{t-1}, Y_t) + \theta_c f_c(X_t, Y_t) + \theta_s f_s(X_{t+1}, Y_t) + \theta_t f_t(Y_{t-1}, Y_t)\} \]
Generic LCRF Formulation

Modeling explicitly input influence on transition

General Linear CRF Likelihood:

\[ P(Y|X, \theta) = \frac{1}{Z(X)} \prod_t \prod_k \exp(\theta_k f_k(Y_t, Y_{t-1}, X_t)) \]

Use indicator variables in \( f_k \) definition to include or disregard the influence of specific RV, e.g. \( 1_{Y_t=i} 1_{X_t=o} \)
Is there an equivalent of the **smoothing problem** in LCRF? Yes: $P(Y_t, Y_{t-1}|X)$

- Solved by (exact) **forward-backward** inference
- Sum-product message passing on the LCRF factor graph

$$P(Y_t, Y_{t-1}|X) \propto \alpha_{t-1}(Y_{t-1})\psi_t(Y_t, Y_{t-1}, X_t)\beta_t(Y_t)$$

**Clique weighting**

$$\psi_t(Y_t, Y_{t-1}, X_t) = \exp\{\theta_{fe}(X_t, Y_t) + \theta_{ft}(Y_{t-1}, Y_t)\}$$

**Forward Message**

$$\alpha_t(i) = \sum_j \psi_t(i, j, X_t)\alpha_{t-1}(j)$$

**Backward Message**

$$\beta_t(j) = \sum_i \psi_{t+1}(i, j, X_{t+1})\beta_{t+1}(i)$$
Other Inference Problems

- Max-product inference can be performed as in the Viterbi algorithm for HMM
- The computationally expensive part is the computation of exponential summation in \( Z(\mathbf{X}) \) term
  - The forward-backward algorithm computes it efficiently as normalization term of \( P(Y_t, Y_{t-1}|\mathbf{X}) \)
- Exact inference in CRF other than chain-like is likely to be computationally impractical
  - Markov Chain Monte Carlo (sample \( y \) rather than estimate \( P(y) \))
  - Variational Belief Propagation (reduce to message passing on trees)
Training LCRF

Maximum (conditional) log-likelihood

$$\max_{\theta} \mathcal{L}(\theta) = \max_{\theta} \sum_{n=1}^{n} \log P(y^n|x^n, \theta)$$

Substituting LCRF conditional formulation

$$\mathcal{L}(\theta) = \sum_{n} \sum_{t} \sum_{k} \theta_k f_k(Y^n_t, Y^n_{t-1}, X^n_t) - \sum_{n} \log Z(X^n)$$
Training LCRF

Maximum (conditional) log-likelihood

$$\max_\theta \mathcal{L}(\theta) = \max_\theta \sum_{n=1}^{n} \log P(y^n|x^n, \theta)$$

Substituting LCRF conditional formulation

$$\mathcal{L}(\theta) = \sum_n \sum_t \sum_k \theta_k f_k (Y_t^n, Y_{t-1}^n, X_t^n) - \sum_n \log Z(X^n) - \sum_k \frac{\theta_k^2}{2\sigma^2}$$

Penalized with a regularization term, e.g. based on $||\theta||^2$
Optimizing the Likelihood

- Typically $L(\theta)$ cannot be maximized in closed form
- Use partial derivatives
  \[
  \frac{\partial L(\theta)}{\partial \theta_k} = \sum_{n,t} f_k(Y^n_t, Y^n_{t-1}, X^n_t) - \sum_{n, t, y, y'} f_k(y, y', X^n_t) P(y, y'|X^n) - \frac{\theta_k}{\sigma^2}
  \]
- First term is $\mathbb{E}[f_k]$ under the empirical distribution (i.e. with $y, y'$ clamped)
- Second term is the $\mathbb{E}[f_k]$ under model distribution
- When gradient is zero these are equal (apart for regularization)
Stochastic Gradient Descent

In practice we can learn the $\theta$ parameters by SGD (or variants)

$$\theta^m = \theta^{m-1} - \nu_m \nabla \mathcal{L}_n(\theta^{m-1})$$

where

$$\nabla \mathcal{L}_{nk}(\theta) = \sum_t f_k(Y_t^n, Y_{t-1}^n, X_t^n) - \sum_t \sum_{y,y'} f_k(y, y', X_t^n) P(y, y'|X^n) - \frac{\theta_k}{N\sigma^2}$$

and $P(y, y'|X^n)$ is estimated by sum-product inference
Engineering Features

Linear CRF have found wide applications

- Text processing: POS-tagging, semantic role identification
- Bioinformatics: sequence alignment, protein structure prediction

Feature functions have often the form $f_k(X_k, Y_k) = 1_{Y_k = \hat{y}_k} q(X_c)$

- $f_k$ is non-zero only for a specific output configuration $\hat{y}_k$
- $f_k$ then depends only on $X_k$ (i.e. features are not shared by classes)

Observation functions $q(X_c)$: word begins with capital, ends with -ing, ...
MRF/CRF in Vision

○ Define **bi-dimensional lattice** on the image
  - Regular grid, patches, superpixels, segments

○ Background/Foreground segmentation
  - $X_i$ Observable label
  - $Y_i$ Region annotation as background/foreground

○ Impose constraints
  - $f_S(Y_i, X_i) \Rightarrow$ Cost of disregarding available annotation
  - $f_H(Y_i, Y_j) \approx [y_i \not= y_j] w_{ij} \Rightarrow$ Label affinity constraint weighted by region similarity $w_{ij}$
Background Segmentation
Background Segmentation
Image Completion

Image Completion

J. Yao, S. Fidler and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," ICCV 2012
Semantic Segmentation

J. Yao, S. Fidler and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," ICCV 2012
Integrating Prior Information

MRF Software

- **CRFsuite** - Fast implementation of linear/chain CRFs for NLP applications (native C++; Scikit-like package python-crfsuite)
- **PyStruct** - Python CRF package including 2D lattices, graph structures and several inference algorithms
- **pgmpy** - Python library for graphical models (includes CRF, MRF and more)
- **Pyro** - Ubers’ own PyTorch provide an implementation of Deep CRF
- **UGM** - Matlab library for Markov Random Fields
- CRF implementations (in particular linear) are present in major DL libraries (e.g. Tensorflow, PyTorch)
A Python Example

```python
from pgmpy.models import MarkovModel
from pgmpy.factors.discrete import DiscreteFactor
import numpy as np
from pgmpy.inference import BeliefPropagation

MM = MarkovModel();
# Add edges (and nodes if not existent)
MM.add_edges_from([(‘f1’, ‘f2’), (‘f2’, ‘f3’), (‘o1’, ‘f1’), (‘o2’, ‘f2’), (‘o3’, ‘f3’)])

# Generate transition feature
transition = np.array([10, 90, 90, 10]);
# Generate corresponding factor
factorH1 = DiscreteFactor([‘f1’, ‘f2’], cardinality=[2, 2], values=transition)
# Add it to the model
MM.add_factors(factorH1)

# Solve smoothing by belief propagation (i.e. estimate hidden RV)
belief_propagation = BeliefPropagation(MM)
ymax = belief_propagation.map_query(variables=[‘f1’, ‘f2’, ‘f3’],
evidence={‘o1’: toVal(‘class1’), ‘o2’: toVal(‘class1’), ‘o3’: toVal(‘class2’)})
...
```
Take Home Messages

- Markov Random Fields
  - Undirected graphical models
  - Allow to express constraints between RV without needing to use probabilities
  - Topology follows data structure/relations and allow embedding prior information
- Conditional Random Fields
  - Constrained MRF learning discriminative posteriors
  - Feature functions to model constraints (often simple hand-coded feature detectors)
  - Parameters allow to linearly combine features
- CRF/MRF are often used as final refinement (segmentation, POS tagging, ...)

DAVIDE BACCIU - ISPR COURSE
Next 2 Lectures

Bayesian Learning and Variational Inference

- Bayesian latent variable models
- Variational bound and its optimization
- Latent Dirichlet Allocation
  - Possibly the simplest Bayesian latent variable model
  - Variational Expectation-Maximization
  - Applications to machine vision