Machine Learning Foundations
Probabilistic graphical models

Anastasia Dubrovina
Computer Science Dept.
Stanford University
In this lecture

- A reminder
  - Supervised learning - regression, classification
  - Unsupervised learning - clustering
  - Dimensionality reduction

- Probabilistic graphical models
  - Types of graphical models
  - Inference
  - Learning
Supervised learning

• We are given a dataset of examples and correct outputs
  \((x_i, y_i)\)

• Our goal: learn to map new input to output

• Regression = output \(y\) is continuous

• Classification = output \(y\) is discrete

Source: Andrew Ng
Supervised learning for 3D data - examples

- Problem: mesh point labeling

- This is a classification problem
Supervised learning for 3D data - examples

- Problem: shape completion

Classification:
- part labels

Regression:
- part orientations
Unsupervised learning

- Here, we are given a dataset of **unlabeled** examples
- Goal: derive structure from the data
- We will encounter
  - Clustering
  - Dimensionality reduction
    - Principal Component Analysis (PCA)

Sources: Andrew Ng, CS233, Fast and efficient spectral clustering (Mathworks post)
Unsupervised learning for 3D data - example

- Problem:
  unsupervised co-segmentation of a set of shapes in descriptor space

Image: Unsupervised Co-Segmentation of a Set of Shapes via Descriptor-Space Spectral Clustering, Sidi et al.’11
Dimensionality reduction for 3D data - examples

- Morphable model for synthesis of 3D faces

Image: A morphable model for the synthesis of 3D faces, Blanz and Vetter’99
Dimensionality reduction for 3D data - examples

• Model of the space of human bodies

Image: The space of human body shapes: reconstruction and parameterization from range scans, Allen et al.'03
Probabilistic graphical models
Graphical models for shape analysis

Anguelov et al., NIPS’04
• Registration posed as MRF
• Pointwise and pairwise surface properties
• Optimization: Loopy BP

Sung et al., SIGGRAPH Asia’14
• Part labels and orientation prediction
• Point segmentation
• Optimization: TRW-S
Graphical models for shape synthesis

Chaudhuri et al., TOG’11
- Part compatibility model
- Optimization: likelihood-weighted sampling

Kalogerakis et al., TOG’12
- Generative model
- Optimization: structure learning

Figure 3: Illustrative example. A small dataset of tables (top) and the probabilistic model learned for this dataset (bottom).
Probabilistic graphical models

- Represent the world as a collection of random variables
  \[ X = \{X_1, \ldots, X_n\} \text{ with joint distribution } p(X_1, \ldots, X_n) \]

- Compactly represent \( p(X_1, \ldots, X_n) \)

- Learn distribution from the data

- Perform inference = compute conditional distributions (CPD) given observations
  \[ p(X_i | X_1 = x_1, \ldots, X_n = x_n) \]

Adapted from CS228 slides by Stefano Ermon
Types of graphical models

- Bayesian Nets (BN)
- Markov Random Fields (MRF)
- Conditional Random Fields (CRF)
Bayesian nets

- A Bayesian network is a **directed acyclic graph (DAG)** $G=(V,E)$ that specifies a joint distribution over $X$ as a product of local conditional distributions.

\[
p(x_1, \ldots, x_n) = \prod_{i \in V} p(x_i | x_{Pa(i)})
\]

\[
p(d, i, g, s, l) = p(d)p(i)p(g | i, d)p(s | i)p(l | g)
\]

- # of degrees of freedom?
- And if we don’t assume conditional independence?

Adapted from CS228 slides by Stefano Ermon
Markov random fields

- **Indirected** graphical models - vs. DAGs for Bayesian nets

- Can represent certain dependencies more easily than BN
- Used more in practice - useful for symmetric interactions
- Harder to interpret, learn and perform inference

Adapted from CS228 slides by Stefano Ermon
MRF - formal definition

• An **MRF** is defined by an *undirected graph*:
  • one node for each random variable
  • *undirected* edges represent dependencies

Adapted from CS228 slides by Stefano Ermon
MRF - formal definition

• An **MRF** is defined by an **indirected graph**:  
  - one node for each random variable  
  - **undirected** edges represent dependencies

• Potential functions, or **factors**, are associated with **cliques** $C$ of the graph

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

Adapted from CS228 slides by Stefano Ermon
MRF - formal definition

• An **MRF** is defined by an *indirected graph*:
  • one node for each random variable
  • **undirected** edges represent dependencies

• Potential functions, or **factors**, are associated with *cliques* $C$ of the graph

\[
p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c)
\]

• $Z$ is the **partition function** - normalizes the distribution

\[
Z = \sum_{\hat{x}_1, \ldots, \hat{x}_n} \prod_{c \in C} \phi_c(\hat{x}_c)
\]

Adapted from CS228 slides by Stefano Ermon
Conditional random fields

• “A framework for building probabilistic models to segment and label sequence data (and more)” [3]

• Specifies the probabilities of possible label sequences given an observation sequence $X$

• Does not expend modeling effort on the observations, which may have complex dependencies and are fixed at test time
CRF - formal definition

• A CRF is a Markov network on variables $\mathbf{X} \cup \mathbf{Y}$, which specifies conditional distribution

$$p(y|x) = \frac{1}{Z(x)} \prod_{c \in C} \phi_c(x_c, y_c)$$

with partition function

$$Z(x) = \sum_{\hat{y}} \prod_{c \in C} \phi_c(x_c, \hat{y}_c)$$

• Difference with a standard Markov net - sum only over $\mathbf{Y}$

Adapted from CS228 slides by Stefano Ermon
Inference in graphical models
Inference in graphical models

- Suppose we are given a BN / MRF / CRF over $X$
- Given observations $E$, we are typically interested in
  - Conditional probability queries
    \[
    p(Y|E = e) = \frac{p(Y, E)}{p(e)}
    \]
  - Maximum a posteriori (MAP) inference, or most probable explanation
    \[
    MAP(Y|E = e) = \arg\max_y p(y|e) = \arg\min_y E(y; e)
    \]
(see next slides for details)
Algorithms for inference (partial list)

- Exact inference
  - Variable elimination
  - Message passing, or belief propagation (BP)
  - ...
- Approximate inference
  - Loopy belief propagation (Loopy BP)
  - Sampling algorithms
  - MAP inference
    - Approximate max-product belief propagation
    - Integer programming and linear programming relaxation
    - ...
    - ...
    - ...
Special case - MAP inference

• The MAP inference task

\[ \arg \max_y p(y) \]
\[ p(y) = \frac{1}{Z} \prod_{c \in C} \phi_c(y_c) \]

• Since the normalization is a constant, this is equivalent to

\[ \arg \max_y \prod_{c \in C} \phi_c(y_c) \]

(called the max-product inference)

• Since log is monotonic, let \( \theta_c(y_c) = \log \phi_c(y_c) \)

• The above becomes equivalent to

\[ \arg \max_y \sum_{c \in C} \theta_c(y_c) \]

(called max-sum)
MAP as an optimization problem

• Consider a pairwise MRF (cliques of size 1, 2)

• Given an MRF specified by a graph \( G=(V,E) \), MAP can be written as

\[
\arg \max_y \sum_{i \in V} \theta_c(y_i) + \sum_{ij \in E} \theta_{ij}(y_i, y_j)
\]

• Can be solved using
  • Tree-reweighed message passing (TRW-S) algorithm [2]
  • Conversion into integer linear programming, or relaxed into linear programming problem [1]
  • Same energy functional can be optimized using graph-cuts [4,5]
Learning in graphical models
Learning in graphical models

- A graphical model has two components: **graph structure** and the **associated potentials**

- Several possible ways to acquire a model:
  - Use expert knowledge to determine the graph and the potentials.
  - Use data+learning to determine the potentials, i.e., **parameter learning**.
  - Use data+learning to determine the graph, i.e., **structure learning**.

- Manual design is difficult to do and can take a long time for an expert

- We usually have access to a set of examples from the distribution we wish to model, e.g., a set of emails annotated by a human (spam or not-spam).
Parameter learning

• Assume some underlying distribution $p^*$
• Given is a dataset of $m$ IID samples from $p^*$
  $$D = \{ x^{(1)}, \ldots, x^{(m)} \}$$
• Goal: given a family of models $M$ with a fixed structure, learn its parameters, so that it captures the distribution $p^*$
• Computing $p^*$ exactly is not achievable in general
  • Limited data provides rough approximation to $p^*$
  • Computational resources
• Therefore, we want to construct the “best” approximation to $p^*$
Parameter learning - MLE estimation

- We want to construct a model \( \hat{\mathcal{M}} \) as close as possible to \( p^* \)
- Possible measure of “closeness” - \textbf{Kullback-Leibler divergence} between \( p^* \) and \( \hat{p} \)

\[
D(p^* \| \hat{p}) = \mathbb{E}_{x \sim p^*} \left[ \log \left( \frac{p^*(x)}{\hat{p}(x)} \right) \right] = \sum_x p^*(x) \log \left( \frac{p^*(x)}{\hat{p}(x)} \right)
\]

- Minimizing KL divergence is equivalent to maximizing the expected log-likelihood

\[
\mathbb{E}_{x \sim p^*} \left[ \log \hat{p}(x) \right]
\]
Parameter learning - MLE estimation (II)

- Minimizing KL divergence is equivalent to maximizing the expected log-likelihood

\[
E_{x \sim p^*} \left[ \log \hat{p}(x) \right]
\]

- \(p^*\) is unknown - resort to maximal empirical likelihood learning

\[
\max_{\hat{\mathcal{M}}} \mathbb{E}_\mathcal{D} \left[ \log \hat{p}(x) \right] = \max_{\hat{\mathcal{M}}} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \log \hat{p}(x)
\]

- Equivalently, maximize the likelihood of the data - density estimation

\[
\hat{p}(x^{(1)}, \ldots, x^{(m)}) = \prod_{x \in \mathcal{D}} \hat{p}(x)
\]
Parameter learning in BN and MRF

• For Bayesian nets, MLE produces a closed-form solution

\[
\theta^*_{x_i|pa(x_i)} = \frac{\#(x_i, pa(x_i))}{\#(pa(x_i))}
\]

where \(\#(y)\) is the number of times \(y\) appears in \(\mathcal{D}\)

• For MRF/CRF, MLE learning is hard
  • Requires inference to compute \(Z(\theta)\)
  • Either use approximate inference - sampling, or
  • use pseudo-likelihood to avoid inference (out of our scope)

Adapted from CS228 slides by Stefano Ermon
Structure learning

- Two main approaches
  - Constraint-based: test independencies, and add edges accordingly
  - Score-based: search for network structures that maximize the probability of observing the given data set $\mathcal{D}$
    \[
    \arg\max_G LL(\mathcal{D}|G)
    \]
- More about specific methods
  - In the following lectures
  - In CS228 notes and in [1]

Adapted from CS228 slides by Stefano Ermon


