Course Outline

- Regression Techniques
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
- Deep Reinforcement Learning
- Probabilistic Graphical Models
  - Bayesian Networks
  - Markov Random Fields
  - Inference (exact & approximate)
- Deep Generative Models
  - Generative Adversarial Networks
  - Variational Autoencoders

Topics of This Lecture

- Recap: Exact inference
  - Sum-Product algorithm
  - Max-Sum algorithm
  - Junction Tree algorithm
- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials
- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

Recap: Factor Graphs

- Joint probability
  - Can be expressed as product of factors:
  \[ p(x) = \prod_{f \in \mathcal{F}} f(x) \]
  - Factor graphs make this explicit through separate factor nodes.
- Converting a directed polytree
  - Conversion to undirected tree creates loops due to moralization!
  - Conversion to a factor graph again results in a tree!

Recap: Sum-Product Algorithm

- Objectives
  - Efficient, exact inference algorithm for finding marginals.
- Procedure:
  - Pick an arbitrary node as root.
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.
- Computational effort
  - Total number of messages = 2 \cdot \text{number of graph edges}.

\[ p(x) = \prod_{v \in \mathcal{V}} \mu_{v \rightarrow x}(x) \]
Max-Sum Algorithm

• Objective: an efficient algorithm for finding
  – Value \( x^{\text{max}} \) that maximises \( p(x) \);
  – Value of \( p(x^{\text{max}}) \).
  
  \( \Rightarrow \) Application of dynamic programming in graphical models.

• In general, maximum marginals \( \neq \) joint maximum.
  – Example:

\[
\begin{array}{c|cc}
  & y = 0 & y = 1 \\
  x = 0 & 0.3 & 0.7 \\
  x = 1 & 0.3 & 0.0 \\
\end{array}
\]

\[
\arg\max_x p(x, y) = 1 \quad \arg\max_x p(x) = 0
\]

Max-Sum Algorithm – Key Ideas

• Key idea 1: Distributive Law (again)

\[
\max(ab, ac) = a \max(b, c)
\]

\[
\max(a + b, a + c) = a + \max(b, c)
\]

\( \Rightarrow \) Exchange products/summations and max operations exploiting the tree structure of the factor graph.

• Key idea 2: Max-Product \( \rightarrow \) Max-Sum

  – We are interested in the maximum value of the joint distribution

  \[
p(x^{\text{max}}) = \max p(x)
\]

  \( \Rightarrow \) Maximize the product \( p(x) \).

  – For numerical reasons, use the logarithm.

  \[
  \ln \left( \max_p p(x) \right) = \max_x \ln p(x).
\]

  \( \Rightarrow \) Maximize the sum (of log-probabilities).

Max-Sum Algorithm

• Maximizing over a chain (max-product)

\[
\begin{array}{c}
x_1 \quad \cdots \quad x_N \quad x_N + 1
\end{array}
\]

• Exchange max and product operators

\[
p(x^{\text{max}}) = \max p(x) = \max_{x_1} \cdots \max_{x_N} p(x)
\]

\[
= \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} \left[ \prod_{i=1}^N \phi_{x_{i-1}, x_i}(x_{i-1}, x_i) \phi_{x_i, x_{i+1}}(x_i, x_{i+1}) \right]
\]

\[
= \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} \left[ \prod_{i=1}^N \phi_{x_{i-1}, x_i}(x_{i-1}, x_i) \phi_{x_i, x_{i+1}}(x_i, x_{i+1}) \right]
\]

• Generalizes to tree-structured factor graph

\[
\max p(x) = \max_{x_1} \cdots \max_{x_N} \prod_{i=1}^N \max_{x_i} f_i(x_i, x_{i+1})
\]

Max-Sum Algorithm

• Initialization (leaf nodes)

\[
\mu_{x \rightarrow f}(x) = 0 \quad \mu_{f \rightarrow x}(x) = \ln f(x)
\]

• Recursion

  – Messages

\[
\mu_{f \rightarrow x}(x) = \max_{x_1} \ln f(x, x_1, \ldots, x_M) + \sum_{m \in \text{neighbors}(x)} \mu_{x_m \rightarrow f}(x_m)
\]

\[
\mu_{x \rightarrow f}(x) = \sum_{i=\text{neighbors}(x)} \mu_{f \rightarrow x_i}(x_i)
\]

  – For each node, keep a record of which values of the variables gave rise to the maximum state:

\[
\phi(x) = \arg \max_{x_1, \ldots, x_M} \ln f(x, x_1, \ldots, x_M) + \sum_{m \in \text{neighbors}(x)} \mu_{x_m \rightarrow f}(x_m)
\]
Max-Sum Algorithm

- **Termination (root node)**
  - Score of maximal configuration
    \[ p^\text{max} = \max_x \sum_{x \in \text{exp}(x)} \mu_{f, \in \cdot}(x) \]
  - Value of root node variable giving rise to that maximum
    \[ x^\text{max} = \arg \max_x \sum_{x \in \text{exp}(x)} \mu_{f, \in \cdot}(x) \]
  - Back-track to get the remaining variable values
    \[ x^\text{max}_{n-1} = \phi(x^\text{max}_n) \]

Visualization of the Back-Tracking Procedure

- **Example: Markov chain**

Topics of This Lecture

- **Recap: Exact inference**
  - Sum-Product algorithm
  - Max-Sum algorithm
  - Junction Tree algorithm
- **Applications of Markov Random Fields**
  - Motivation
  - Unary potentials
  - Pairwise potentials
- **Solving MRFs with Graph Cuts**
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

Junction Tree Algorithm

- **Motivation**
  - Exact inference on general graphs.
  - Works by turning the initial graph into a junction tree with one node per clique and then running a sum-product-like algorithm.
  - Intractable on graphs with large cliques.

- **Main steps**
  1. If starting from directed graph, first convert it to an undirected graph by **moralization**.
  2. Introduce additional links by **triangulation** in order to reduce the size of cycles.
  3. **Find cliques** of the moralized, triangulated graph.
  4. Construct a new graph from the **maximal cliques**.
  5. Remove minimal links to break cycles and get a
    \[ \Rightarrow \text{Apply regular message passing} \] to perform inference.
Junction Tree Algorithm

1. Convert to an undirected graph through moralization.
   - Marry the parents of each node.
   - Remove edge directions.

Image source: Z. Gharahmani
Slide adapted from Zoubin Gharahmani

Junction Tree Algorithm

2. Triangulate
   - Such that there is no loop of length > 3 without a chord.
   - This is necessary so that the final junction tree satisfies the "running intersection" property (explained later).

Image source: Z. Gharahmani
Slide adapted from Zoubin Gharahmani

Junction Tree Algorithm

3. Find cliques of the moralized, triangulated graph.

Image source: Z. Gharahmani
Slide adapted from Zoubin Gharahmani

Junction Tree Algorithm

4. Construct a new junction graph from maximal cliques.
   - Create a node from each clique.
   - Each link carries a list of all variables in the intersection.
     - Drawn in a "separator" box.

Image source: Z. Gharahmani
Slide adapted from Zoubin Gharahmani

Junction Tree Algorithm

5. Remove links to break cycles → junction tree.
   - For each cycle, remove the link(s) with the minimal number of shared nodes until all cycles are broken.
   - Result is a maximal spanning tree, the junction tree.

Image source: Z. Gharahmani
Slide adapted from Zoubin Gharahmani

Junction Tree – Properties

- Running intersection property
  - If a variable appears in more than one clique, it also appears in all intermediate cliques in the tree.
  - This ensures that neighboring cliques have consistent probability distributions.
- Local consistency → global consistency
Interpretation of the Junction Tree

- Undirected graphical model
- Junction tree

\[
P(U) = \prod \frac{P(\text{Clique})}{P(\text{Separator})}
\]

\[
P(A,B,C) = P(A,B) P(B,C) / P(B)
\]

Junction Tree: Example 1

- Algorithm
  1. Moralization
  2. Triangulation (not necessary here)
- (b) Moral graph
- (c) Junction graph

Junction Tree: Example 2

- Without triangulation step
  - The final graph will contain cycles that we cannot break without losing the running intersection property!
- (b) Without triangulation

Junction Tree: Example 2

- When applying the triangulation
  - Only small cycles remain that are easy to break.
  - Running intersection property is maintained.
- (b) With triangulation
Junction Tree Algorithm

• Good news
  – The junction tree algorithm is efficient in the sense that for a given graph there does not exist a computationally cheaper approach.

• Bad news
  – This may still be too costly.
  – Effort determined by number of variables in the largest clique.
  – Grows exponentially with this number (for discrete variables).
  ⇒ Algorithm becomes impractical if the graph contains large cliques!

Loopy Belief Propagation

• Alternative algorithm for loopy graphs
  – Sum-Product on general graphs.
  – Strategy: simply ignore the problem.
  – Initial unit messages passed across all links, after which messages are passed around until convergence
  • Convergence is not guaranteed!
  • Typically break off after fixed number of iterations.
  • Approximate but tractable for large graphs.
  • Sometime works well, sometimes not at all.

Topics of This Lecture

- Recap: Exact inference
  – Sum-Product algorithm
  – Max-Sum algorithm
  – Junction Tree algorithm

- Applications of Markov Random Fields
  – Motivation
  – Unary potentials
  – Pairwise potentials

- Solving MRFs with Graph Cuts
  – Graph cuts for image segmentation
  – s-t mincut algorithm
  – Extension to non-binary case
  – Applications

Markov Random Fields (MRFs)

• What we’ve learned so far…
  – We know they are undirected graphical models.
  – Their joint probability factorizes into clique potentials,
    \[ p(x) = \frac{1}{Z} \prod \phi_C(x_C) \]
    which are conveniently expressed as
    \[ \psi_C(x_C) = \exp\{-E(x_C)\} \]
  – We know how to perform inference for them.
    • Sum/Max-Product BP for exact inference in tree-shaped MRFs.
    • Loopy BP for approximate inference in arbitrary MRFs.
    • Junction Tree algorithm for converting arbitrary MRFs into trees.

• But what are they actually good for?
  And how do we apply them in practice?

Markov Random Fields

• Allow rich probabilistic models.
  – But built in a local, modular way.
  – Learn local effects, get global effects out.

• Very powerful when applied to regular structures.
  – Such as images...

Applications of MRFs

• Movie “No Way Out” (1987)
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation

Results by [Roth & Black, CVPR’05]

"True" image content

Noisy observations

"Smoothness constraints"

Observation process

Super-resolution

Convert a low-res image into a high-res image!
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow

MRF Structure for Images

- Basic structure
  - Observation model
    - How likely is it that node $x_i$ has label $L_i$ given observation $y_i$?
    - This relationship is usually learned from training data.
  - Neighborhood relations
    - Simplest case: 4-neighborhood
    - Discourage neighboring pixels to have different labels.
    - This can either be learned or be set to fixed "penalties".

MRF Nodes as Pixels

- Original image
- Degraded image
- Reconstruction from MRF modeling pixel neighborhood statistics

These neighborhood statistics can be learned from training data!
MRF Nodes as Patches

Image patches

Scene patches

More general relationships expressed by potential functions $\Phi$ and $\Psi$.

Network Joint Probability

- Interpretation of the factorized joint probability

$$P(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

Energy Formulation

- Energy function

$$E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j)$$

  - Single-node (unary) potentials $\phi$
    - Encode local information about the given pixel/patch.
    - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?
  - Pairwise potentials $\psi$
    - Encode neighborhood information.
    - How different is a pixel/patch's label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)

How to Set the Potentials? Some Examples

- Unary potentials
  - E.g., color model, modeled with a Mixture of Gaussians
    $$\phi(x_i, y_i; \theta_k) = \log \sum_k \theta_k p(k|x_i) \mathcal{N}(y_i; \mu_k, \Sigma_k)$$

- Pairwise potentials $\psi$
  - Potts Model
    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.
  - Extension: "contrast sensitive Potts model"
    $$\psi(x_i, x_j, g_{ij}(y); \theta_\beta) = \theta_\beta g_{ij}(y) \delta(x_i \neq x_j)$$

How to Set the Potentials? Some Examples

- Pairwise potentials
  - Potts Model
    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.
  - Extension: "contrast sensitive Potts model"
    $$\psi(x_i, x_j, g_{ij}(y); \theta_\beta) = \theta_\beta g_{ij}(y) \delta(x_i \neq x_j)$$

  - where
    $$g_{ij}(y) = e^{-\beta |y_i - y_j|}$$

    - Discourages label changes except in places where there is also a large change in the observations.

Extension: Conditional Random Fields (CRF)

- Idea: Model conditional instead of joint probability

  - Unary potential
    $$\phi(D| x_i, x_j)$$

  - Prior Potts model
    $$\psi(D| x_i)$$

- Energy formulation

$$E(x) = \sum_{i \in S} \phi(D| x_i) + \sum_{j \in N_i} (\phi(D| x_i, x_j) + \psi(x_i, x_j)) + \text{cost}$$

- Uniform Prior (Potts Model)
Example: MRF for Image Segmentation

- **MRF structure**
  - Unary potential \( \phi(x_i) \)
  - Pairwise potential \( \phi(D(x_i, x_j)) \)

Data (D) | Unary likelihood | Pair-wise Terms | MAP Solution
--- | --- | --- | ---

Energy Minimization

- **Goal:**
  - Infer the optimal labeling of the MRF.
- **Many inference algorithms are available,** e.g.
  - Simulated annealing
  - Iterated conditional modes (ICM)
  - Belief propagation
  - Graph cuts
  - Variational methods
  - Monte Carlo sampling

- **Recently, Graph Cuts have become a popular tool**
  - Only suitable for a certain class of energy functions.
  - But the solution can be obtained very fast for typical vision problems (~1 MPixel/sec).

Topics of This Lecture

- **Recap: Exact inference**
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm
- **Applications of Markov Random Fields**
  - Motivation
  - Unary potentials
  - Pairwise potentials
- **Solving MRFs with Graph Cuts**
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

Graph Cuts for Binary Problems

- **Idea:** convert MRF into source-sink graph

Minimum cost cut can be computed in polynomial time (max-flow/min-cut algorithms)

Simple Example of Energy

\[
E(L) = \sum_p D_p(L_p) + \sum_{pq \in N} W_{pq} \cdot \delta(L_p \neq L_q)
\]

Adding Regional Properties

- **Regional bias example**
  - Suppose \( I' \) and \( I \) are given “expected” intensities of object and background

\[
D_x(t) = \exp \left( -\frac{\| I_x - I' \|^2}{2\sigma^2} \right)
\]

**NOTE:** hard constrains are not required, in general.
Adding Regional Properties

“expected” intensities of object and background \( I^o \) and \( I^b \) can be re-estimated.

\[
D_s(t) \propto \exp(-||I^o - t||^2/2\sigma^2)
\]

\[
D_t(s) \propto \exp(-||I^b - s||^2/2\sigma^2)
\]

EM-style optimization

References and Further Reading

- A gentle introduction to Graph Cuts can be found in the following paper:

- Try the GraphCut implementation at https://pub.ist.ac.at/~vnk/software.html