Structured Prediction for Scene Understanding I

Raquel Urtasun

University of Toronto

June 20, 2014
Goal of this lecture

- Understand what structured prediction is
- Learn how to formulate a problem to be successful in practice
What is structured prediction?
In "typical" machine learning

\[ f : \mathcal{X} \rightarrow \mathbb{R} \]

the input \( \mathcal{X} \) can be anything, and the output is a real number (e.g., classification, regression)

In Structured Prediction

\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]

the input \( \mathcal{X} \) can be anything, and the output is a complex object (e.g., image segmentation, parse tree)
In "typical" machine learning

\[ f : \mathcal{X} \rightarrow \mathbb{R} \]

the input \( \mathcal{X} \) can be anything, and the output is a real number (e.g., classification, regression)

In Structured Prediction

\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]

the input \( \mathcal{X} \) can be anything, and the output is a \textbf{complex} object (e.g., image segmentation, parse tree)

In this lecture \( \mathcal{Y} \) is a discrete space, ask me later if you are interested in continuous variables.
In "typical" machine learning
\[ f : \mathcal{X} \rightarrow \mathbb{R} \]
the input \( \mathcal{X} \) can be anything, and the output is a real number (e.g., classification, regression)

In Structured Prediction
\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]
the input \( \mathcal{X} \) can be anything, and the output is a \textbf{complex} object (e.g., image segmentation, parse tree)

In this lecture \( \mathcal{Y} \) is a discrete space, ask me later if you are interested in continuous variables.
We want to predict multiple random variables which are related

- **Computer Vision:**
  - Semantic Segmentation (output: pixel-wise labeling)
  - Object detection (output: 2D or 3D bounding boxes)
  - Stereo Reconstruction (output: 3D map)
  - Scene Understanding (output: 3D bounding box reprinting the layout)
Structured Prediction and its Applications

We want to predict multiple random variables which are related

- Natural Language processing
  - Machine Translation (output: sentence in another language)
  - Parsing (output: parse tree)

- Computational Biology
  - Protein Folding (output: 3D protein)
Why structured?

- Independent prediction is good but...

- Neighboring pixels should have same labels (if they look similar).
Why structured?

- Independent prediction is good but...

- Neighboring pixels should have same labels (if they look similar).
A graphical model defines

- A family of probability distributions over a set of random variables
- This is expressed via a graph, which encodes the conditional independences of the distribution

Two types of graphical models: Directed and undirected
Bayesian Networks

- The graph $G = (V, E)$ is acyclic and directed
- Factorization over distributions by conditioning on parent nodes

$$p(y) = \prod_{i \in V} p(y_i | y_{pa}(i))$$

**Example**

$$p(y) = p(y_l | y_k)p(y_k | y_i, y_j)p(y_i)p(y_j)$$
Undirected Graphical Model

- Also called Markov Random Field, or Markov Network
- Graph $G = (V, E)$ is undirected and has no self-edges
- Factorization over cliques

$$p(y) = \frac{1}{Z} \prod_{r \in R} \psi_r(y_r)$$

with $Z = \sum_{y \in Y} \prod_{r \in R} \psi_r(y_r)$ the partition function

- Example

$$p(y) = \frac{1}{Z} \psi(y_i, y_j) \psi(y_j, y_k) \psi(y_i) \psi(y_j) \psi(y_k)$$

- **Difficulty**: Exponentially many configurations
- Undirected models will be the focus of this lecture
Factor Graph Representation

- Graph $G = (V, \mathcal{F}, \mathcal{E})$, with variable nodes $V$, factor nodes $\mathcal{F}$ and edges $\mathcal{E}$
- **Scope** of a factor $N(F) = \{ i \in V : (i, F) \in \mathcal{E} \}$
- Factorization over factors

$$p(y) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)})$$
Factor graphs are explicit about the factorization

Figure: from [Nowozin et al]
They define the family of distributions and thus the *capacity*.

*Figure*: from [Nowozin et al]
Markov Random Fields vs Conditional Random Fields

- Markov Random Fields (MRFs) define
  \[ p(y) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}) \]

- Conditional Random Fields (CRFs) define
  \[ p(y|x) = \frac{1}{Z(x)} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}; x) \]

  *x* is not a random variable (i.e., not part of the probability distribution)
The probability is completely determined by the energy

\[ p(y) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}) \]

\[ = \frac{1}{Z} \exp \left( \log(\psi_F(y_{N(F)})) \right) \]

\[ = \frac{1}{Z} \exp \left( - \sum_{F \in \mathcal{F}} E_F(y_F) \right) \]

where \( E_F(y_F) = -\log(\psi_F(y_{N(F)})) \)
Factor graphs define a family of distributions

We are interested in identifying individual members by parameters

\[ E_F(y_F) = -w^T \phi_F(y_F) \]

Figure: from [Nowozin et al]
Learning Tasks

- Estimation of the parameters $w$

$$E_F(y_F) = -w^T \phi_F(y_F)$$

- Learn the structure of the model
- Learn with hidden variables
Inference Tasks

Given an input $x \in \mathcal{X}$ we want to compute

- **MAP estimate** or minimum energy configuration

$$
\arg\max_{y \in \mathcal{Y}} p(y|x) = \arg\max_{y \in \mathcal{Y}} \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}; x, w)
$$

$$
= \arg\max_{y \in \mathcal{Y}} \exp\left(- \sum_{F \in \mathcal{F}} E_F(y_F, x, w)\right)
$$

$$
= \arg\min_{y \in \mathcal{Y}} \sum_{F \in \mathcal{F}} E_F(y_F, x, w)
$$

- **Marginals** $p(y_i)$ or max marginals $\max_{y_i \in \mathcal{Y}_i} p(y_i)$, which requires computing the partition function $Z$, i.e.,

$$
\log(Z(x, w)) = \log \sum_{y \in \mathcal{Y}} \exp(-E(y; x, w))
$$

$$
\mu_F(y_F) = p(y_F|x, w)
$$
Inference Tasks

Given an input $x \in \mathcal{X}$ we want to compute

- **MAP estimate** or minimum energy configuration

  $$\arg\max_{y \in \mathcal{Y}} p(y|x) = \arg\max_{y \in \mathcal{Y}} \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}; x, w)$$

  $$= \arg\max_{y \in \mathcal{Y}} \exp\left( - \sum_{F \in \mathcal{F}} E_F(y_F, x, w) \right)$$

  $$= \arg\min_{y \in \mathcal{Y}} \sum_{F \in \mathcal{F}} E_F(y_F, x, w)$$

- **Marginals** $p(y_i)$ or max marginals $\max_{y_i \in \mathcal{Y}_i} p(y_i)$, which requires computing the partition function $Z$, i.e.,

  $$\log(Z(x, w)) = \log \sum_{y \in \mathcal{Y}} \exp(-E(y; x, w))$$

  $$\mu_F(y_F) = p(y_F|x, w)$$
Inference in Markov Random Fields
Compute the MAP estimate is typically NP-hard

\[
\max_{y \in \mathcal{Y}} p(y|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} w^T \phi_r(y_r)
\]

Notable exceptions are:

- Belief propagation for tree-structure models
Compute the MAP estimate is typically NP-hard

\[
\max_{y \in \mathcal{Y}} p(y|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} w^T \phi_r(y_r)
\]

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with submodular potentials
Compute the MAP estimate is typically NP-hard

$$\max_{y \in Y} p(y|x) = \max_{y \in Y} \sum_{r \in R} w^T \phi_r(y_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice
Compute the MAP estimate is typically NP-hard

\[
\max_{y \in \mathcal{Y}} p(y|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} w^T \phi_r(y_r)
\]

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

Difficulties

- Deal with the exponentially many states in $y$
Compute the MAP estimate is typically NP-hard

\[
\max_{y \in \mathcal{Y}} p(y|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} w^T \phi_r(y_r)
\]

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

Difficulties

- Deal with the exponentially many states in \( y \)

We are going to see examples of the three techniques
Compute the MAP estimate is typically NP-hard

\[
\max_{y \in Y} p(y|x) = \max_{y \in Y} \sum_{r \in \mathcal{R}} w^T \phi_r(y_r)
\]

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with submodular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

Difficulties

- Deal with the exponentially many states in \( y \)

We are going to see examples of the three techniques
Belief Propagation

- Compact notation
  \[ \theta_r(y_r) = w^T \phi_r(y_r) \]

- Inference can be written as
  \[ \max_{y \in Y} \sum_{r \in \mathcal{R}} \theta_r(y_r) \]

- For the example
  \[ \max_{y_i, y_j, y_k, y_l} \{ \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_G(y_k, y_l) \} \]
Belief Propagation

\[ \theta^*(y) = \max_{y_i, y_j, y_k, y_l} \{ \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_H(y_k, y_l) \} \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_l} \theta_H(y_k, y_l) \]
Belief Propagation

\[ \theta^*(y) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_l} \theta_H(y_k, y_l) \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \rightarrow y_k}(y_k) \]
Belief Propagation

\[ \theta^*(\mathbf{y}) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_l} \theta_H(y_k, y_l) \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \rightarrow Y_k}(y_k) \]
\[ \theta^*(y) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \rightarrow y_k}(y_k) \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + r_{G \rightarrow y_j}(y_j) \]
Belief Propagation

\[ \theta^*(y) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \rightarrow y_k}(y_k) \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + r_{G \rightarrow y_j}(y_j) \]
Tree Generalization

\[
\theta^*(y) = \max_{y_i, y_j, y_k, y_l, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k)
\]

\[
= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k)
\]
Tree Generalization

\[ \theta^*(y) = \max_{y_i, y_k, y_k, y_l, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k) \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k) \]

\[ = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H\rightarrow y_k}(y_k) + r_{I\rightarrow y_k}(y_k) \]
$\theta^*(y) = \max_{y_i, y_j, y_k, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k)$

$= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k)$

$= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_l} \theta_H(y_l, y_k) + r_{H\rightarrow Y_k}(y_k) + r_{I\rightarrow Y_k}(y_k)$

$= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_l} \theta_H(y_l, y_k) + q_{y_k \rightarrow G}(y_k)$
$\theta^*(y) = \max_{y_i, y_j, y_k, y_r, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k)$

$\theta^*(y) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k)$

$\theta^*(y) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{Y_k \rightarrow Y_k}(y_k) + r_{R \rightarrow Y_k}(y_k)$

$\theta^*(y) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + q_{Y_k \rightarrow G}(y_k)$
Factor Graph Max Product

Iteratively updates and passes messages:

- $r_{F \rightarrow Y_i} \in \mathbb{R}^{Y_i}$: factor to variable message
- $q_{Y_i \rightarrow F} \in \mathbb{R}^{Y_i}$: variable to factor message

Figure: from [Nowozin et al]
Variable to factor

- Let $M(i)$ be the factors adjacent to variable $i$, $M(i) = \{ F \in F : (i, F) \in E \}$
- Variable-to-factor message

$$q_{y_i \rightarrow F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \rightarrow y_i}(y_i)$$

Figure: from [Nowozin et al]
Factor to variable

- Factor-to-variable message

\[
r_{F \rightarrow y_i}(y_i) = \max_{y'_F \in Y_F, y'_i = y_i} \left( \theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \rightarrow F}(y'_j) \right)
\]

Figure: from [Nowozin et al]
Message Scheduling

1. Select one variable as tree root
2. Compute leaf-to-root messages
3. Compute root-to-leaf messages

Figure: from [Nowozin et al]
Max Product v Sum Product

Max sum version of max-product

1. Compute leaf-to-root messages

\[ q_{y_i \to F}(y_i) = \sum_{F' \in \mathcal{M}(i) \setminus \{F\}} r_{F' \to y_i}(y_i) \]

2. Compute root-to-leaf messages

\[ r_{F \to y_i}(y_i) = \max_{y_i' \in \mathcal{Y}_F, y_i' = y_i} \left( \theta(y_i') + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \to F}(y_j') \right) \]

Sum-product

1. Compute leaf-to-root messages

\[ q_{y_i \to F}(y_i) = \sum_{F' \in \mathcal{M}(i) \setminus \{F\}} r_{F' \to y_i}(y_i) \]

2. Compute root-to-leaf messages

\[ r_{F \to y_i}(y_i) = \log \sum_{y_i' \in \mathcal{Y}_F, y_i' = y_i} \exp \left( \theta(y_i') + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \to F}(y_j') \right) \]
Max Product v Sum Product

Max sum version of max-product

1. Compute leaf-to-root messages

\[ q_{y_i \rightarrow F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \rightarrow y_i}(y_i) \]

2. Compute root-to-leaf messages

\[ r_{F \rightarrow y_i}(y_i) = \max_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \left( \theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \rightarrow F}(y'_j) \right) \]

Sum-product

1. Compute leaf-to-root messages

\[ q_{y_i \rightarrow F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \rightarrow y_i}(y_i) \]

2. Compute root-to-leaf messages

\[ r_{F \rightarrow y_i}(y_i) = \log \sum_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \exp \left( \theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \rightarrow F}(y'_j) \right) \]
Computing marginals

- Partition function can be evaluated at the root

\[
\log Z = \log \sum_{y_r} \exp \left( \sum_{F \in M(r)} r_{F \to y_r}(y_r) \right)
\]

- Marginal distributions, for each factor

\[
\mu_F(y_F) = p(y_F) = \frac{1}{Z} \exp \left( \theta_F(y_F) + \sum_{i \in N(F)} q_{y_i \to F}(y_i) \right)
\]
Computing marginals

- Partition function can be evaluated at the root

\[
\log Z = \log \sum_{y_r} \exp \left( \sum_{F \in M(r)} r_{F \rightarrow y_r}(y_r) \right)
\]

- Marginal distributions, for each factor

\[
\mu_F(y_F) = p(y_F) = \frac{1}{Z} \exp \left( \theta_F(y_F) + \sum_{i \in N(F)} q_{y_i \rightarrow F}(y_i) \right)
\]

- Marginals at every node

\[
\mu_{y_i}(y_i) = p(y_i) = \frac{1}{Z} \exp \left( \sum_{F \in M(i)} r_{F \rightarrow y_i}(y_i) \right)
\]
Generalizations to loops

- It is called **loopy belief propagation** (Perl, 1988)
- no schedule that removes dependencies
- Different messaging schedules (synchronous/asynchronous, static/dynamic)
- Slight changes in the algorithm
MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- **Inference task:**
  \[ \hat{y} = \arg \max_y \sum_r \theta_r(y_r) \]

- **Variables** \( b_r(y_r) \):
  \[
  \begin{bmatrix}
  b_1(0) \\
  b_1(1) \\
  b_2(0) \\
  b_2(1) \\
  b_{12}(0, 0) \\
  b_{12}(1, 0) \\
  b_{12}(0, 1) \\
  b_{12}(1, 1)
  \end{bmatrix}^T
  \begin{bmatrix}
  \theta_1(0) \\
  \theta_1(1) \\
  \theta_2(0) \\
  \theta_2(1) \\
  \theta_{12}(0, 0) \\
  \theta_{12}(1, 0) \\
  \theta_{12}(0, 1) \\
  \theta_{12}(1, 1)
  \end{bmatrix}
  \]

  \[ b_{12}(y_r) \in \{0, 1\} \]

  s.t.
MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- **Inference task:**
  \[
  \hat{y} = \arg\max_y \sum_r \theta_r(y_r)
  \]

- **Variables** \( b_r(y_r) \):
  \[
  \begin{bmatrix}
  b_1(0) \\
  b_1(1) \\
  b_2(0) \\
  b_2(1) \\
  b_{12}(0, 0) \\
  b_{12}(1, 0) \\
  b_{12}(0, 1) \\
  b_{12}(1, 1)
  \end{bmatrix}^T \begin{bmatrix}
  \theta_1(0) \\
  \theta_1(1) \\
  \theta_2(0) \\
  \theta_2(1) \\
  \theta_{12}(0, 0) \\
  \theta_{12}(1, 0) \\
  \theta_{12}(0, 1) \\
  \theta_{12}(1, 1)
  \end{bmatrix}
  \]

  \[
  \max_{b_1, b_2, b_{12}} \quad \text{s.t. } b_r(y_r) \in \{0, 1\} \\
  \sum_{y_r} b_r(y_r) = 1
  \]
MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- Inference task:

\[
\hat{y} = \arg \max_y \sum_r \theta_r(y_r)
\]

- Variables \( b_r(y_r) \):

\[
\begin{bmatrix}
    b_1(0) \\
    b_1(1) \\
    b_2(0) \\
    b_2(1) \\
    b_{12}(0, 0) \\
    b_{12}(1, 0) \\
    b_{12}(0, 1) \\
    b_{12}(1, 1)
\end{bmatrix}^T \begin{bmatrix}
    \theta_1(0) \\
    \theta_1(1) \\
    \theta_2(0) \\
    \theta_2(1) \\
    \theta_{12}(0, 0) \\
    \theta_{12}(1, 0) \\
    \theta_{12}(0, 1) \\
    \theta_{12}(1, 1)
\end{bmatrix}
\]

\[
\max_{b_1, b_2, b_{12}} \quad \text{s.t.} \quad \begin{align*}
    b_r(y_r) &\in \{0, 1\} \\
    \sum_{y_r} b_r(y_r) &= 1 \\
    \sum_{y_p \setminus y_r} b_p(y_p) &= b_r(y_r)
\end{align*}
\]
MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- Inference task:
  \[ \hat{y} = \arg \max_y \sum_r \theta_r(y_r) \]

- Variables \( b_r(y_r) \):

\[
\begin{bmatrix}
  b_1(0) \\
  b_1(1) \\
  b_2(0) \\
  b_2(1) \\
  b_{12}(0, 0) \\
  b_{12}(1, 0) \\
  b_{12}(0, 1) \\
  b_{12}(1, 1)
\end{bmatrix}^T
\begin{bmatrix}
  \theta_1(0) \\
  \theta_1(1) \\
  \theta_2(0) \\
  \theta_2(1) \\
  \theta_{12}(0, 0) \\
  \theta_{12}(1, 0) \\
  \theta_{12}(0, 1) \\
  \theta_{12}(1, 1)
\end{bmatrix}
\]

\[ \max_{b_1, b_2, b_{12}} \]

\[ \begin{align*}
\sum_y b_r(y_r) &\in \{0, 1\} \\
\sum_{y_p \setminus y_r} b_p(y_p) &= b_r(y_r)
\end{align*} \]
MAP LP Relaxation Task

\[
\begin{array}{cccccc}
    b_1(1) & b_1(2) & b_2(1) & b_2(2) & b_{12}(1, 1) & b_{12}(2, 1) \\
    b_{12}(1, 2) & b_{12}(2, 2) & & & & \\
\end{array}
\]

\[
\begin{array}{cccccc}
    \theta_1(1) & \theta_1(2) & \theta_2(1) & \theta_2(2) & \theta_{12}(1, 1) & \theta_{12}(2, 1) \\
    \theta_{12}(1, 2) & \theta_{12}(2, 2) & & & & \\
\end{array}
\]

\[
\begin{align*}
\text{max} & \quad \sum_{y_r} b_r(y_r) \quad \left[ b_{12}(1, 1), b_{12}(2, 1), b_{12}(1, 2), b_{12}(2, 2) \right] \\
\text{s.t.} & \quad \sum_{y_r} b_r(y_r) = 1 \\
& \quad \sum_{y_p \setminus y_r} b_p(y_p) = b_r(y_r)
\end{align*}
\]
MAP LP Relaxation Task

\[
\max_{b_r} \sum_{r, y_r} b_r(y_r) \theta_r(y_r)
\]

s.t.

\[
\begin{align*}
& b_r(y_r) \in \{0, 1\} \\
& \sum_{y_r} b_r(y_r) = 1 \\
& \sum_{y_p \setminus y_r} b_p(y_p) = b_r(y_r)
\end{align*}
\]

Can be solved by any standard LP solver but slow because of typically many variables and constraints. Can we do better?
MAP LP Relaxation Task

\[
\begin{align*}
\max_{b_r} & \quad \sum_{r,y_r} b_r(y_r) \theta_r(y_r) \\
\text{s.t.} & \quad b_r(y_r) \in \{0, 1\} \\
& \quad \sum_{y_r} b_r(y_r) = 1 \\
\text{Marginalization} & \end{align*}
\]
LP relaxation:

$$\max_{b_r} \sum_{r, y_r} b_r(y_r) \theta_r(y_r) \quad \text{s.t.} \quad b_r(y_r) \in \{0, 1\}$$

Local probability $b_r$

Marginalization
LP relaxation:

\[
\max_{b_r} \sum_{r, y_r} b_r(y_r) \theta_r(y_r) \quad \text{s.t.} \quad b_r(y_r) \in \{0, 1\}
\]

Local probability \( b_r \)

Marginalization

Can be solved by any standard LP solver but slow because of typically many variables and constraints. Can we do better?
LP relaxation:

\[
\max_{b_r} \sum_{r,y_r} b_r(y_r) \theta_r(y_r)
\]

s.t.

\[
b_r(y_r) \in \{0, 1\}
\]

Local probability \(b_r\)

Marginalization

Can be solved by any standard LP solver but \textbf{slow} because of typically many variables and constraints. Can we do better?
**Observation:** Graph structure in marginalization constraints.

![Graph](image)

Use dual to take advantage of structure in constraint set

- Set of parents of region $r$: $P(r)$
- Set of children of region $r$: $C(r)$

\[
\forall r, y_r, p \in P(r) \quad \sum_{y_p \setminus y_r} b_p(y_p) = b_r(y_r)
\]

- Lagrange multipliers for every constraint:

\[
\forall r, y_r, p \in P(r) \quad \lambda_{r \rightarrow p}(y_r)
\]
Re-parameterization of score $\theta_r(y_r)$:

$$\hat{\theta}_r(y_r) = \theta_r(y_r) + \sum_{p \in P(r)} \lambda_{r \to p}(y_r) - \sum_{c \in C(r)} \lambda_{c \to r}(y_c)$$

Properties of dual program:

$$\min_{\lambda} q(\lambda) = \min_{\lambda} \sum_{r} \max_{y_r} \hat{\theta}_r(y_r)$$

- **Dual upper-bounds primal** $\forall \lambda$
- Convex problem
- Unconstrained task
- Doing block coordinate descent in the dual results on message passing (Lagrange multipliers are your messages)
Block-coordinate descent solvers iterate the following steps:

- Take a block of Lagrange multipliers
- Optimize sub-problem of dual function w.r.t. this block while keeping all other variables fixed

**Advantage:** fast due to analytically computable sub-problems

Same type of algorithms also exist to compute approximate marginals
Theorem [Kolmogorov and Zabih, 2004]: If the energy function is a function of binary variables containing only unary and pairwise factors, the discrete energy minimization problem

$$\min_y \sum_{r \in R} E(y_r, x)$$

can be formulated as a graph cut problem if only off all pairwise energies are submodular

$$E_{i,j}(0,0) + E_{i,j}(1,1) \leq E_{i,j}(0,1) + E_{i,j}(1,0)$$
The ST-mincut problem

- The st-mincut is the st-cut with the minimum cost

[Source: P. Kohli]
Back to our energy minimization

Construct a graph such that

1. Any st-cut corresponds to an assignment of \( x \)
2. The cost of the cut is equal to the energy of \( x \) : \( E(x) \)

[Source: P. Kohli]
St-mincut and Energy Minimization

\[ E(x) = \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j) \]

For all ij, \( \theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1) \)

Equivalent (transformable)

\[ E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i(1-x_j) \]

[Source: P. Kohli]
How are they equivalent?

\[ A = \theta_{ij}(0,0) \quad B = \theta_{ij}(0,1) \quad C = \theta_{ij}(1,0) \quad D = \theta_{ij}(1,1) \]

\[
\begin{array}{ccc}
0 & x_j & 1 \\
0 & 0 & B \\
C & 0 & D \\
1 & 1 & 1
\end{array}
\]

\[
A = A + (\theta_{ij}(1,0) - \theta_{ij}(0,0)) x_i + (\theta_{ij}(1,0) - \theta_{ij}(0,0)) x_j
\]

\[
+ (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1)) (1-x_i) x_j
\]

\[ \theta_{ij}(x_i, x_j) = \theta_{ij}(0,0) \]

\[ B + C - A - D \geq 0 \text{ is true from the submodularity of } \theta_{ij} \]

[Source: P. Kohli]
Graph Construction

$E(a_1, a_2)$

Source (0)

$\text{Sink (1)}$

$\text{Source (0)}$

$\text{Sink (1)}$

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 \]

Source (0)

\[ a_1 \]

Sink (1)

\[ a_2 \]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 \]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 \]

[Source: P. Kohli]
\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

Source (0) → a₁ → 1 → a₂ → 2 → Sink (1)

Cost of cut = 11

\[ a_1 = 1 \quad a_2 = 1 \]

\[ E(1,1) = 11 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

Source (0)

\[ a_1 \]

\[ a_2 \]

Sink (1)

\[ \text{st-mincut cost} = 8 \]

\[ a_1 = 1 \quad a_2 = 0 \]

\[ E(1,0) = 8 \]

[Source: P. Kohli]
How to compute the St-mincut?

Solve the dual \textit{maximum flow} problem.

Compute the maximum flow between Source and Sink s.t.

\begin{itemize}
  \item Edges: Flow < Capacity
  \item Nodes: Flow in = Flow out
\end{itemize}

\textbf{Min-cut/Max-flow Theorem}

In every network, the maximum flow equals the cost of the st-mincut.

Assuming non-negative capacity

[Source: P. Kohli]
How does the code look like

```c
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeID(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```

[Source: P. Kohli]
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeId(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeId(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q

    add_weights(nodeId(p), nodeId(q), cost(p,q));

end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeId(p));
// is the label of pixel p (0 or 1)

[Source: P. Kohli]
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeId(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeId(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeId(p), nodeId(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeId(p));
// is the label of pixel p (0 or 1)
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeId(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeId(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q

    add_weights(nodeId(p), nodeId(q), cost(p,q));

end

    g->compute_maxflow();

    label_p = g->is_connected_to_source(nodeId(p));

    // is the label of pixel p (0 or 1)
Example: Figure-Ground Segmentation

Binary labeling problem

(Original)  (Color model)  (Indep. Prediction)

Figure: from [Nowozin et al]
Example: Figure-Ground Segmentation

- Markov Random Field

$$E(y, x, w) = \sum_i \log p(y_i|x_i) + w \sum_{(i,j) \in E} C(x_i, x_j) I(y_i \neq y_j)$$

with $C(x_i, x_j) = \exp(\gamma ||x_i - x_j||^2)$, and $w \geq 0$.

![Figure: from [Nowozin et al]](image)

- Why do we need the condition $w \geq 0$?
Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

Figure: from [Nowozin et al]

\[ y \]
\[ y^0 \]
Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

Figure: from [Nowozin et al]

\[ N(y^0) \]
Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

Figure: from [Nowozin et al]
Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

Figure: from [Nowozin et al]
Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels $\alpha, \beta, \gamma$

  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0 \\
  V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)
  \]

- **Semi-metric** if it satisfies for any set of labels $\alpha, \beta, \gamma$

  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0
  \]
Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels \( \alpha, \beta, \gamma \)

  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0 \\
  V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)
  \]

- **Semi-metric** if it satisfies for any set of labels \( \alpha, \beta, \gamma \)

  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0
  \]
Examples for 1D label set

- **Truncated quadratic is a semi-metric**
  \[ V(\alpha, \beta) = \min(K, |\alpha - \beta|^2) \]
  with \( K \) a constant.

- **Truncated absolute distance is a metric**
  \[ V(\alpha, \beta) = \min(K, |\alpha - \beta|) \]
  with \( K \) a constant.
Examples for 1D label set

- Truncated quadratic is a semi-metric
  
  \[ V(\alpha, \beta) = \min(K, |\alpha - \beta|^2) \]

  with \( K \) a constant.

- Truncated absolute distance is a metric
  
  \[ V(\alpha, \beta) = \min(K, |\alpha - \beta|) \]

  with \( K \) a constant.

- Potts model is a metric
  
  \[ V(\alpha, \beta) = K \cdot T(\alpha \neq \beta) \]

  with \( T(\cdot) = 1 \) if the argument is true and 0 otherwise.
Examples for 1D label set

- Truncated quadratic is a semi-metric

\[ V(\alpha, \beta) = \min(K, |\alpha - \beta|^2) \]

with \( K \) a constant.

- Truncated absolute distance is a metric

\[ V(\alpha, \beta) = \min(K, |\alpha - \beta|) \]

with \( K \) a constant.

- Potts model is a metric

\[ V(\alpha, \beta) = K \cdot T(\alpha \neq \beta) \]

with \( T(\cdot) = 1 \) if the argument is true and 0 otherwise.
Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label $\alpha$.
- **Alpha - Beta Swaps**: Checks if a node with class $\alpha$ wants to switch to $\beta$.
- Binary problems that can be solved exactly for certain types of potentials.

Figure: Alpha-beta Swaps. Figure from [Nowozin et al]
Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label $\alpha$.
- **Alpha - Beta Swaps**: Checks if a node with class $\alpha$ wants to switch to $\beta$.
- Binary problems that can be solved exactly for certain types of potentials.

![Alpha-beta Swaps](image-url)  

**Figure**: Alpha-beta Swaps. Figure from [Nowozin et al]
Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label $\alpha$
- **Alpha - Beta Swaps**: Checks if a node with class $\alpha$ wants to switch to $\beta$.
- Binary problems that can be solved exactly for certain types of potentials

Figure: Alpha-beta Swaps. Figure from [Nowozin et al]
Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label $\alpha$
- **Alpha - Beta Swaps**: Checks if a node with class $\alpha$ wants to switch to $\beta$.
- Binary problems that can be solved exactly for certain types of potentials

**Figure**: Alpha-beta Swaps. Figure from [Nowozin et al]
Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label $\alpha$
- **Alpha - Beta Swaps**: Checks if a node with class $\alpha$ wants to switch to $\beta$.
- Binary problems that can be solved exactly for certain types of potentials

**Figure**: Alpha-beta Swaps. Figure from [Nowozin et al]
Binary Moves

- $\alpha - \beta$ moves works for semi-metrics
- $\alpha$ expansion works for $V$ being a metric

For certain $x^1$ and $x^2$, the move energy is sub-modular
The set of vertices includes the two terminals $\alpha$ and $\beta$, as well as image pixels $p$ in the sets $P_\alpha$ and $P_\beta$ (i.e., $f_p \in \{\alpha, \beta\}$).

Each pixel $p \in P_{\alpha\beta}$ is connected to the terminals $\alpha$ and $\beta$, called $t$-links.

Each set of pixels $p, q \in P_{\alpha\beta}$ which are neighbors is connected by an edge $e_{p,q}$
Learning in graphical models
Learning Tasks

- Estimation of the parameters $\mathbf{w}$

$$E_F(\mathbf{y}_F) = -\mathbf{w}^T \phi_F(\mathbf{y}_F)$$

- Learn the structure of the model
- Learn with hidden variables
Learning the parameters

- Log-loss learning
- Max margin learning
- One parameter extensions
- Pseudolikelihood
- Perturb and MAP approaches
- Contrastive Divergence
- ...
We are given a dataset of $\mathcal{S} = \{(x^i, y^i), \cdots, (x^N, y^N)\}$

We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$.
Supervised Learning

- We are given a dataset of $S = \{(x^i, y^i), \cdots, (x^N, y^N)\}$
- We also have the task loss that we want to minimize $\Delta: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- We want to find the weights by solving

$$\min_w \mathbb{E}_{(x,y) \sim \mathcal{D}} \{\Delta(y, f(x))\}$$

with $f(x) = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y)$

Typical supervised learning algorithms are convex.

Why is this problem difficult?
Supervised Learning

- We are given a dataset of $S = \{(x^i, y^i), \cdots, (x^N, y^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- We want to find the weights by solving
  \[
  \min_w \mathbb{E}_{(x,y) \sim \mathcal{D}} \{\Delta(y, f(x))\}
  \]
  with $f(x) = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y)$
- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting
  \[
  \min_w \sum_{(x,y) \in \mathcal{D}} \ell(w, x, y) + \frac{C}{p} \|w\|^p_p,
  \]
Supervised Learning

- We are given a dataset of \( S = \{ (x^i, y^i), \cdots, (x^N, y^N) \} \)
- We also have the task loss that we want to minimize \( \Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \)
- We want to find the weights by solving
  \[
  \min_{w} E_{(x,y) \sim D} \{ \Delta(y, f(x)) \}
  \]
  with \( f(x) = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) \)
- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting
  \[
  \min_{w} \sum_{(x,y) \in D} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
  \]
- Typical supervised learning algorithms are convex.
Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(x^i, y^i), \cdots, (x^N, y^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- We want to find the weights by solving
  \[
  \min_w \mathbb{E}_{(x, y) \sim \mathcal{D}} \{\Delta(y, f(x))\}
  \]
  with $f(x) = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y)$
- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting
  \[
  \min_w \sum_{(x, y) \in \mathcal{D}} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
  \]
- Typical supervised learning algorithms are convex.
- Why is this problem difficult?
Supervised Learning

- We are given a dataset of \( \mathcal{S} = \{(x^i, y^i), \cdots, (x^N, y^N)\} \)
- We also have the task loss that we want to minimize \( \Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R} \)
- We want to find the weights by solving
  \[
  \min_w \mathbb{E}_{(x,y) \sim \mathcal{D}} \{ \Delta(y, f(x)) \}
  \]
  with \( f(x) = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) \)
- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting
  \[
  \min_w \sum_{(x,y) \in \mathcal{D}} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
  \]
- Typical supervised learning algorithms are convex.
- Why is this problem difficult?
Max-margin Learning

- Regularized Risk Minimization

\[
\min_w \sum_{(x,y) \in S} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
\]

- In structured SVMs

\[
\ell_{hinge}(w, x, y) = \max_{\hat{y} \in \mathcal{Y}} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \}.
\]
Max-margin Learning

- Regularized Risk Minimization

\[
\min_w \sum_{(x,y) \in S} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
\]

- In structured SVMs

\[
\ell_{\text{hinge}}(w, x, y) = \max_{\hat{y} \in \mathcal{Y}} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \}
\]

- Optimize the unconstrained problem

\[
\min_w \sum_{(x,y) \in S} \max_{\hat{y} \in \mathcal{Y}} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \} + \frac{C}{p} \|w\|_p^p,
\]

- Convex but non-smooth.

- Use sub gradient methods
Max-margin Learning

- Regularized Risk Minimization
  \[
  \min_w \sum_{(x,y) \in S} \ell(w, x, y) + \frac{C}{p} \|w\|^p,
  \]

- In structured SVMs
  \[
  \ell_{\text{hinge}}(w, x, y) = \max_{\hat{y} \in Y} \left\{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \right\}
  \]

- Optimize the unconstrained problem
  \[
  \min_w \sum_{(x,y) \in S} \max_{\hat{y} \in Y} \left\{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \right\} + \frac{C}{p} \|w\|^p,
  \]

- Convex but non-smooth.
- Use sub gradient methods
Equivalent Formulation

- Optimize the unconstrained problem

\[
\min_w \sum_{(x,y) \in S} \max_{\hat{y} \in \hat{Y}} \left\{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \right\} + \frac{C}{p} \|w\|_p^p,
\]

- Write as constraints

\[
\min_w \sum_{(x,y) \in S} \xi_n^2 + \frac{C}{p} \|w\|_p^p,
\]

\[
\text{s.t. } \max_{\hat{y} \in \hat{Y}} \left\{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \right\} \leq \xi_n
\]
Equivalent Formulation

- Optimize the unconstrained problem

\[
\min_w \sum_{(x,y) \in S} \max_{\hat{y} \in \mathcal{Y}} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \} + \frac{C}{p} \|w\|^p_p,
\]

- Write as constraints

\[
\min_w \sum_{(x,y) \in S} \xi^2_n + \frac{C}{p} \|w\|^p_p,
\]

\[
\text{s.t. } \max_{\hat{y} \in \mathcal{Y}} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \} \leq \xi_n
\]

- Or equivalently

\[
\min_w \sum_{(x,y) \in S} \xi^2_n + \frac{C}{p} \|w\|^p_p,
\]

\[
\text{s.t. } \forall \hat{y} \quad \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \leq \xi_n
\]

- Use cutting plane methods as exp. many constraints
Equivalent Formulation

- Optimize the unconstrained problem

\[
\min_w \sum_{(x,y)\in S} \max_{\hat{y}\in Y} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \} + \frac{C}{p} \|w\|_p^p,
\]

- Write as constraints

\[
\min_w \sum_{(x,y)\in S} \xi_n^2 + \frac{C}{p} \|w\|_p^p,
\]

\[
s.t. \max_{\hat{y}\in Y} \{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \} \leq \xi_n
\]

- Or equivalently

\[
\min_w \sum_{(x,y)\in S} \xi_n^2 + \frac{C}{p} \|w\|_p^p,
\]

\[
s.t. \forall \hat{y} \ell(y, \hat{y}) + w^T \Phi(x, \hat{y}) - w^T \Phi(x, y) \leq \xi_n
\]

- Use cutting plane methods as exp. many constraints
Log-loss Learning

- **Regularized Risk Minimization**

\[
\min_w \sum_{(x,y) \in S} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
\]

- **CRF loss:** The conditional distribution is

\[
p_{x,y}(\hat{y}; w) = \frac{1}{Z(x, y)} \exp (\Delta(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
\]

\[
Z(x, y) = \sum_{\hat{y} \in \mathcal{Y}} \exp (\Delta(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
\]

where \(\Delta(y, \hat{y})\) is a prior distribution and \(Z(x, y)\) the partition function, and

\[
\ell_{log}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}.
\]
Log-loss Learning

- Regularized Risk Minimization

\[
\min_w \sum_{(x,y) \in S} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p,
\]

- CRF loss: The conditional distribution is

\[
p_{x,y}(\hat{y}; w) = \frac{1}{Z(x, y)} \exp(\Delta(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
\]

\[
Z(x, y) = \sum_{\hat{y} \in Y} \exp(\Delta(y, \hat{y}) + w^\top \Phi(x, \hat{y}))
\]

where \(\Delta(y, \hat{y})\) is a prior distribution and \(Z(x, y)\) the partition function, and

\[
\ell_{\text{log}}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}.
\]

- Convex problem

- Problem: to do gradient descent I need to compute \(Z\)
Log-loss Learning

- Regularized Risk Minimization

\[ \min_w \sum_{(x,y) \in S} \ell(w, x, y) + \frac{C}{p} \|w\|_p^p, \]

- CRF loss: The conditional distribution is

\[ p_{x,y}(\hat{y}; w) = \frac{1}{Z(x, y)} \exp(\Delta(y, \hat{y}) + w^T \Phi(x, \hat{y})) \]

\[ Z(x, y) = \sum_{\hat{y} \in Y} \exp(\Delta(y, \hat{y}) + w^T \Phi(x, \hat{y})) \]

where \( \Delta(y, \hat{y}) \) is a prior distribution and \( Z(x, y) \) the partition function, and

\[ \ell_{\log}(w, x, y) = \ln \frac{1}{p_{x,y}(y; w)}. \]

- Convex problem

- Problem: to do gradient descent I need to compute \( Z \)
Relation between loss functions

- The CRF program is

\[
    \text{min}_w \left\{ \sum_{(x, y) \in S} \ln Z(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

where \((x, y) \in S\) ranges over training pairs and \(d = \sum_{(x, y) \in S} \Phi(x, y)\) is the vector of empirical means, and

\[
    Z(x, y) = \sum_{\hat{y} \in Y} \exp \left( \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right)
\]

- In structured SVMs

\[
    \text{min}_w \left\{ \sum_{(x, y) \in S} \max_{\hat{y} \in Y} \left\{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right\} - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]
Relation between loss functions

- The CRF program is

\[
(CRF) \quad \min_w \left\{ \sum_{(x,y) \in S} \ln Z(x,y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

where \((x,y) \in S\) ranges over training pairs and \(d = \sum_{(x,y) \in S} \Phi(x,y)\) is the vector of empirical means, and

\[
Z(x,y) = \sum_{\hat{y} \in Y} \exp \left( \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right)
\]

- In structured SVMs

\[
(\text{structured SVM}) \quad \min_w \left\{ \sum_{(x,y) \in S} \max_{\hat{y} \in Y} \left\{ \Delta(y, \hat{y}) + w^T \Phi(x, \hat{y}) \right\} - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]
A family of structure prediction problems

- One parameter extension of CRFs and structured SVMs [Hazan & Urtasun, NIPS 2010]

\[
\min_w \left\{ \sum_{(x,y) \in S} \ln Z_\epsilon(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\} ,
\]

\(d\) is the empirical means, and

\[
\ln Z_\epsilon(x, y) = \epsilon \ln \sum_{\hat{y} \in \mathcal{Y}} \exp \left( \frac{\Delta(y, \hat{y}) + w^T \Phi(x, \hat{y})}{\epsilon} \right)
\]

- CRF if \(\epsilon = 1\), Structured SVM if \(\epsilon = 0\) respectively.

- One can devise a single algorithm to solve both problems
A family of structure prediction problems

- One parameter extension of CRFs and structured SVMs [Hazan & Urtasun, NIPS 2010]

\[
\min_w \left\{ \sum_{(x,y) \in S} \ln Z_\epsilon(x, y) - d^T w + \frac{C}{p} \|w\|_p^p \right\},
\]

\(d\) is the empirical means, and

\[
\ln Z_\epsilon(x, y) = \epsilon \ln \sum_{\hat{y} \in \mathcal{Y}} \exp \left( \frac{\Delta(y, \hat{y}) + w^T \Phi(x, \hat{y})}{\epsilon} \right)
\]

- CRF if \(\epsilon = 1\), Structured SVM if \(\epsilon = 0\) respectively.

- One can devise a single algorithm to solve both problems
Structure Prediction for Scene Understanding II

Raquel Urtasun

University of Toronto

June 20, 2014
Structured Prediction in Practice
Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
What are my random variables?

How are they related? i.e., graph

How do I encode my prior knowledge about the problem?

\[
E(y_1, \cdots, y_n, x) = \sum_{r \in \mathcal{R}} w^T_r \phi_r(y_r, x)
\]
Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

\[ E(y_1, \cdots, y_n, x) = \sum_{r \in R} w_r^T \phi_r(y_r, x) \]

**Advise:** Forget about probabilities in your potentials, the partition function will take care of that!

\[ p(y|x) = \frac{1}{Z} \exp(-E(y, x)) \]
Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

\[ E(y_1, \ldots, y_n, x) = \sum_{r \in \mathcal{R}} w_r^T \phi_r(y_r, x) \]

**Advise:** Forget about probabilities in your potentials, the partition function will take care of that!

\[ p(y|x) = \frac{1}{Z} \exp(-E(y, x)) \]

- How can I do inference? Why is this complicated?

\[ \min_{y_1, \ldots, y_n} E(y_1, \ldots, y_n) \]
Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

\[ E(y_1, \ldots, y_n, x) = \sum_{r \in \mathcal{R}} w_r^T \phi_r(y_r, x) \]

- **Advise**: Forget about probabilities in your potentials, the partition function will take care of that!

\[ p(y|x) = \frac{1}{Z} \exp(-E(y, x)) \]

- How can I do inference? Why is this complicated?

\[ \min_{y_1, \ldots, y_n} E(y_1, \ldots, y_n) \]

- If you know how to do inference you will know how to do learning! Where does the complication come from?
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems

Why do I care about holistic (i.e., joint) models?
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?
  - Well understood inference algorithms, some of them exact!
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?
- Well understood inference algorithms, some of them exact!
- Good learning algorithms exist as well
Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? → there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?
- Well understood inference algorithms, some of them exact!
- Good learning algorithms exist as well
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don’t think this is a negative point though
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don’t think this is a negative point though
- Is a log-linear model expressive enough?
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don’t think this is a negative point though
- Is a log-linear model expressive enough?
- Where does the structure come from?
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don’t think this is a negative point though
- Is a log-linear model expressive enough?
- Where does the structure come from?
- Can I learn everything from unlabeled data? How deep are you?
What’s not so good?

- Use as a keyword, approaches that don’t think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don’t think this is a negative point though
- Is a log-linear model expressive enough?
- Where does the structure come from?
- Can I learn everything from unlabeled data? How deep are you?
First task: 3D indoor scene understanding
Task: Estimate the 3D layout from a single image

- What’s the metric? how do I know if I did well?
- How would you parameterize this problem? (i.e., what are your random variables?)
3D layout for Indoors

**Task:** Estimate the 3D layout from a single image

- What’s the metric? how do I know if I did well?
- How would you parameterize this problem? (i.e., what are your random variables?)
- What prior knowledge would you like to encode?
3D layout for Indoors

**Task:** Estimate the 3D layout from a single image

- What’s the metric? how do I know if I did well?
- How would you parameterize this problem? (i.e., what are your random variables?)
- What prior knowledge would you like to encode?
Isn’t this a segmentation task where each pixel can be labeled as a wall?
Isn’t this a segmentation task where each pixel can be labeled as a wall?

Let’s start with the most simple parameterization: split the image into super pixels, and for each define

$$y_i \in \{1, \cdots, 5\}$$

the label the super pixel is associated with
Isn’t this a segmentation task where each pixel can be labeled as a wall?

Let’s start with the most simple parameterization: split the image into super pixels, and for each define

$$y_i \in \{1, \ldots, 5\}$$

the label the super pixel is associated with.

Define the energy as

$$E(y_1, \ldots, y_n, x) = \sum_{r \in \mathcal{R}} w_r^T \phi_r(y_r, x)$$

What are the $\phi_r(y_r, x)$?
Isn’t this a segmentation task where each pixel can be labeled as a wall?

Let’s start with the most simple parameterization: split the image into super pixels, and for each define

\[ y_i \in \{1, \cdots, 5\} \]

the label the super pixel is associated with.

Define the energy as

\[ E(y_1, \cdots, y_n, x) = \sum_{r \in \mathcal{R}} w_r^T \phi_r(y_r, x) \]

What are the \( \phi_r(y_r, x) \)?
Geometric Features as Unaries

- Orientation maps [Leet et al. 09], geometric context [Hoiem et al. 05]

How do I construct my unaries $\phi_i(x, y_i)$?

- What are my pairwise potentials $\phi_{ij}(x, y_i, y_j)$?
Geometric Features as Unaries

- Orientation maps [Leet et al. 09], geometric context [Hoiem et al. 05]

How do I construct my unaries $\phi_i(x, y_i)$?

- What are my pairwise potentials $\phi_{ij}(x, y_i, y_j)$?

- What's the problem with smoothness potentials?
Geometric Features as Unaries

- Orientation maps [Leet el al 09], geometric context [Hoiem et al. 05]

How do I construct my unaries $\phi_i(x, y_i)$?

What are my pairwise potentials $\phi_{ij}(x, y_i, y_j)$?

What’s the problem with smoothness potentials?

Are we missing something? What extra knowledge do we have?
Geometric Features as Unaries

- Orientation maps [Leet et al. 09], geometric context [Hoiem et al. 05]

How do I construct my unaries $\phi_i(x, y_i)$?

- What are my pairwise potentials $\phi_{ij}(x, y_i, y_j)$?

- What’s the problem with smoothness potentials?

- Are we missing something? What extra knowledge do we have?
Labels are not appearing at random in the image.

We can encode that the world is Manhattan by expressing *ordering* constraints.
Labels are not appearing at random in the image

We can encode that the world is Manhattan by expressing **ordering** constraints

What would that be?
Manhattan World for Segmentation

- Labels are not appearing at random in the image

- We can encode that the world is Manhattan by expressing **ordering** constraints

- What would that be?

- What’s the order of the potentials?
Manhattan World for Segmentation

- Labels are not appearing at random in the image

![Image of Manhattan world segmentation]

- We can encode that the world is Manhattan by expressing **ordering** constraints
- What would that be?
- What’s the order of the potentials?
- Can we do inference easily?
Labels are not appearing at random in the image

We can encode that the world is Manhattan by expressing *ordering* constraints

What would that be?

What’s the order of the potentials?

Can we do inference easily?

Which algorithm will you use? would it take a long time? would it be optimal?
Labels are not appearing at random in the image

We can encode that the world is Manhattan by expressing ordering constraints

What would that be?

What’s the order of the potentials?

Can we do inference easily?

Which algorithm will you use? would it take a long time? would it be optimal?
Let’s assume that I can compute vanishing points

How should I express the problem? how many degrees of freedom do I have?
Let’s assume that I can compute vanishing points.

How should I express the problem? how many degrees of freedom do I have?
Let’s assume that I can compute vanishing points.

How should I express the problem? how many degrees of freedom do I have?

We parameterize a layout with 4 variables $y_i \in \mathcal{Y}, i \in \{1, \ldots, 4\}$ [Lee et al. 09]

What have I lost with respect to before?
Let’s assume that I can compute vanishing points.

How should I express the problem? how many degrees of freedom do I have?

We parameterize a layout with 4 variables $y_i \in \mathcal{Y}, i \in \{1, \ldots, 4\}$ [Lee et al. 09]

What have I lost with respect to before?

What have I won?
Let’s assume that I can compute vanishing points.

How should I express the problem? how many degrees of freedom do I have?

We parameterize a layout with 4 variables $y_i \in \mathcal{Y}, i \in \{1, \ldots, 4\}$ [Lee et al. 09]

What have I lost with respect to before?

What have I won?
Let’s define the energy. Which potentials will you use?

\[ E(y_1, \ldots, y_4) = \sum_r w_r^T \phi(y_r, x) \]
Energy of the problem

- Let’s define the energy. Which potentials will you use?

\[ E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(y_r, x) \]

- Let’s start with the geometric features

We will like to maximize the yellow pixels in the left wall, green in the frontal wall, etc.
Energy of the problem

- Let’s define the energy. Which potentials will you use?

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T \phi(y_r, x) \]

- Let’s start with the geometric features

- We will like to maximize the yellow pixels in the left wall, green in the frontal wall, etc

- We will also like to minimize the other colors in those walls, e.g., all but yellow in left wall
Energy of the problem

Let’s define the energy. Which potentials will you use?

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T \phi(y_r, x) \]

Let’s start with the geometric features

- We will like to maximize the yellow pixels in the left wall, green in the frontal wall, etc.
- We will also like to minimize the other colors in those walls, e.g., all but yellow in left wall.
How do I express this in my potentials?

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T \phi(y_r, x) \]

How many \( y_i \)'s do I need to define them?
How do I express this in my potentials?

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T \phi(y_r, x) \]

How many \( y_i \)'s do I need to define them?

Do I need other potentials?
More on energy

How do I express this in my potentials?

\[ E(y_1, \ldots, y_4) = \sum_r w_r^T \phi(y_r, x) \]

How many \(y_i\)'s do I need to define them?

Do I need other potentials?

Why did I need more potentials than just geometric features before?
How do I express this in my potentials?

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T \phi(y_r, x) \]

How many \( y_i \)'s do I need to define them?

Do I need other potentials?

Why did I need more potentials than just geometric features before?
Is inference easy in this model? Why?

What can we do?
Is inference easy in this model? Why?

What can we do?

Multi-label problem, message passing seems the best option
Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
  - Problem: High order potentials $\rightarrow$ very very slow!
Is inference easy in this model? Why?

What can we do?

Multi-label problem, message passing seems the best option

Problem: High order potentials $\rightarrow$ very very slow!

Let’s think about it for a second, maybe we can do something
Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials $\rightarrow$ very very slow!
- Let’s think about it for a second, maybe we can do something
- Remember we want to compute sum of features in faces, and search over all possible faces
Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials → very very slow!
- Let’s think about it for a second, maybe we can do something
- Remember we want to compute sum of features in faces, and search over all possible faces
- Let’s first take a detour
Is inference easy in this model? Why?

What can we do?

Multi-label problem, message passing seems the best option

Problem: High order potentials $\rightarrow$ very very slow!

Let’s think about it for a second, maybe we can do something

Remember we want to compute sum of features in faces, and search over all possible faces

Let’s first take a detour
Integral Images

- We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle.
- How can we do this efficiently?
- Compute the sum area table, also called integral image.

\[
s(i, j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k, l)
\]

This can be efficiently computed using a recursive (raster-scan) algorithm:

\[
s(i, j) = s(i - 1, j) + s(i, j - 1) - s(i - 1, j - 1) + f(i, j)
\]
We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle.

How can we do this efficiently?

Compute the **sum area table**, also called **integral image**

\[
s(i, j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k, l)
\]

This can be efficiently computed using a recursive (raster-scan) algorithm

\[
s(i, j) = s(i - 1, j) + s(i, j - 1) - s(i - 1, j - 1) + f(i, j)
\]

Then compute the sum on the rectangle by accessing 4 numbers

\[
S([i_0, i_1] \times [j_0, j_1]) = s(i_1, j_1) - s(i_1, j_0 - 1) - s(i_0 - 1, j_1) + s(i_0 - 1, j_0 - 1)
\]
We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle. How can we do this efficiently? Compute the **sum area table**, also called **integral image**

\[ s(i, j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k, l) \]

This can be efficiently computed using a recursive (raster-scan) algorithm

\[ s(i, j) = s(i - 1, j) + s(i, j - 1) - s(i - 1, j - 1) + f(i, j) \]

Then compute the sum on the rectangle by accessing 4 numbers

\[ S([i_0, i_1] \times [j_0, j_1]) = s(i_1, j_1) - s(i_1, j_0 - 1) - s(i_0 - 1, j_1) + s(i_0 - 1, j_0 - 1) \]

Can we do something similar in our case?
We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle.

How can we do this efficiently?

Compute the sum area table, also called integral image

\[
s(i, j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k, l)
\]

This can be efficiently computed using a recursive (raster-scan) algorithm

\[
s(i, j) = s(i - 1, j) + s(i, j - 1) - s(i - 1, j - 1) + f(i, j)
\]

Then compute the sum on the rectangle by accessing 4 numbers

\[
S([i_0, i_1] \times [j_0, j_1]) = s(i_1, j_1) - s(i_1, j_0 - 1) - s(i_0 - 1, j_1) + s(i_0 - 1, j_0 - 1)
\]

Can we do something similar in our case?
Generalization to 3D

- Faces are generalizations of rectangles
- We need to extend the concept of integral images to 3D
- This is called integral geometry [Schwing et al. 12a]
- How does this work?

\[
\phi_{\{left\_w\}}(y_i, y_j, y_k, x) = H_1(y_i, y_j, x) - H_2(y_j, y_k, x)
\]
Faces are generalizations of rectangles

We need to extend the concept of integral images to 3D

This is called **integral geometry** [Schwing et al. 12a]

How does this work?

\[
\phi_{\{\text{floor}\}}(y_i, y_j, y_k, x) = H_1(y_i, y_j, x) - H_2(y_j, y_k, x)
\]
What are the implications?

- We can now write the problem in terms of potentials of order at most 2

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T(y_r, x) \]

and \( r \) only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily.
What are the implications?

- We can now write the problem in terms of potentials of order at most 2

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T(y_r, x) \]

and \( r \) only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily

- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more
What are the implications?

- We can now write the problem in terms of potentials of order at most 2
  \[ E(y_1, \cdots, y_4) = \sum_r w_r^T(y_r, x) \]
  and \( r \) only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily

- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more

- Some of this \( r \) share the same weights, as they come from the integral geometry.
What are the implications?

- We can now write the problem in terms of potentials of order at most 2

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T(y_r, x) \]

and \( r \) only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily

- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more

- Some of this \( r \) share the same weights, as they come from the integral geometry.

- If they are not shared then they do not represent the same problem
What are the implications?

- We can now write the problem in terms of potentials of order at most 2

\[ E(y_1, \cdots, y_4) = \sum_r w_r^T(y_r, x) \]

and \( r \) only contains sets of 2 random variables.

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily.

- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more.

- Some of this \( r \) share the same weights, as they come from the integral geometry.

- If they are not shared then they do not represent the same problem.

- This speeds up the message passing inference by a few orders of magnitude.
What are the implications?

- We can now write the problem in terms of potentials of order at most 2
  \[ E(y_1, \cdots, y_4) = \sum_r w_r^T (y_r, x) \]
  and \( r \) only contains sets of 2 random variables
- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily
- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more
- Some of this \( r \) share the same weights, as they come from the integral geometry.
- If they are not shared then they do not represent the same problem
- This speed ups the message passing inference by a few orders of magnitude
Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal solution
Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal
- What other algorithms do you know that give the optimal solution?
Can we compute the optimal solution?

The graph of the previous problem loops

Message passing will not give the optimal

What other algorithms do you know that give the optimal solution?

Let’s look at branch and bound
Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal
- What other algorithms do you know that give the optimal solution?
- Let’s look at branch and bound
We have to define:

1. A parameterization that defines **sets of hypothesis**.
2. A **scoring function** $f$
3. **Tight bounds** on the scoring function that can be computed very **efficiently**
Parameterization of the Problem

- Layout with 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, \ldots, 4\}$ [Lee et al. 09]
- How do we define $\mathcal{Y}$?
- Is this problem continuous or discrete?

We parameterize the sets by **intervals** of minimum and maximum angles

$$\{[y_{1\text{min}}, y_{1\text{max}}], \ldots, [y_{4\text{min}}, y_{4\text{max}}]\}$$

- Why intervals?
- We have defined already the scoring function. What about the bounds?
Properties of the Bounds

Derive bounds $\bar{f}$ for the original scoring function $w^T \phi(y, x)$ that satisfy:

1. The bound of the interval $\hat{Y}$ has to upper-bound the true cost of each hypothesis $y \in \hat{Y}$,
   \[ \forall y \in \hat{Y}, \quad \bar{f}(\hat{Y}) \geq w^T \phi(y, x). \]

2. The bound has to be exact for every single hypothesis,
   \[ \forall y \in Y, \quad \bar{f}(y) = w^T \phi(y, x). \]

Can we define this for our problem?
Intuitions from 2D

Let’s look at the 2D case again

- We want to compute the bounding box that maximizes a scoring function
- Let’s try to do this with branch and bound
- We define an interval as the max and min of the x and y axis of the rectangle

The scoring function sums features in the rectangle defined by the BBox

\[ E(y_1, \ldots, y_4) = \sum_{i \in BBox(y)} f_i(x) \]
Let’s look at the 2D case again

- We want to compute the bounding box that maximizes a scoring function
- Let’s try to do this with branch and bound
- We define an interval as the max and min of the $x$ and $y$ axis of the rectangle

The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \ldots, y_4) = \sum_{i \in \text{BBox}(y)} f_i(x)$$
Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox

\[ E(y_1, \ldots, y_4) = \sum_{i \in BBox(y)} f_i(x) \]

- Some features are positive and some are negative
Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f_i(x) \]

- Some features are positive and some are negative

- **Trick**: Divide the space into negative and positive features

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox^+(y)} f_i^+(x) + \sum_{i \in BBox^-(y)} f_i^-(x) \]
Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox
  \[ E(y_1, \ldots, y_4) = \sum_{i \in BBox(y)} f_i(x) \]

- Some features are positive and some are negative

- **Trick**: Divide the space into negative and positive features
  \[ E(y_1, \ldots, y_4) = \sum_{i \in BBox^+(y)} f_i^+(x) + \sum_{i \in BBox^-(y)} f_i^-(x) \]

- Bound the positive and negative independently
  \[ bound(E(\tilde{Y})) = \tilde{f}^+(\tilde{Y}, x) + \tilde{f}^-(\tilde{Y}, x) \]
The scoring function sums features in the rectangle defined by the BBox

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f_i(x) \]

Some features are positive and some are negative

**Trick:** Divide the space into negative and positive features

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f_i^+(x) + \sum_{i \in BBox(y)} f_i^-(x) \]

Bound the positive and negative independently

\[ \text{bound}(E(\bar{Y})) = \bar{f}^+(\bar{Y}, x) + \bar{f}^-(\bar{Y}, x) \]
Bounding the functions

- Energy was defined as

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f^+_i(x) + \sum_{i \in BBox(y)} f^-_i(x) \]

- Bound the positive and negative independently

\[ \text{bound}(E(\bar{y})) = \bar{f}^+(\bar{y}, x) + \bar{f}^-(\bar{y}, x) \]

- These bounds are very simple? What are they?
Bounding the functions

- Energy was defined as

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f^+(x) + \sum_{i \in BBox(y)} f^-(x) \]

- Bound the positive and negative independently

\[ \text{bound}(E(\bar{y})) = \bar{f}^+(\bar{y}, x) + \bar{f}^-(\bar{y}, x) \]

- These bounds are very simple? What are they?

- How can we compute them very fast?
Bounding the functions

- Energy was defined as

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f^+_i(x) + \sum_{i \in BBox(y)} f^-_i(x) \]

- Bound the positive and negative independently

\[ \text{bound}(E(\vec{Y})) = \tilde{f}^+(\vec{Y}, x) + \tilde{f}^-(\vec{Y}, x) \]

- These bounds are very simple? What are they?
- How can we compute them very fast?
- What's the complexity of computing them?
Bounding the functions

- Energy was defined as
  \[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f_i^+(x) + \sum_{i \in BBox(y)} f_i^-(x) \]

- Bound the positive and negative independently
  \[ bound(E(\tilde{Y})) = \tilde{f}^+(\tilde{Y}, x) + \tilde{f}^-(\tilde{Y}, x) \]

- These bounds are very simple? What are they?
- How can we compute them very fast?
- What’s the complexity of computing them?
- How many integral images do we need?
Bounding the functions

- Energy was defined as

\[ E(y_1, \cdots, y_4) = \sum_{i \in BBox(y)} f_i^+(x) + \sum_{i \in BBox(y)} f_i^-(x) \]

- Bound the positive and negative independently

\[ bound(E(\bar{Y})) = \bar{f}^+(\bar{Y}, x) + \bar{f}^-(\bar{Y}, x) \]

- These bounds are very simple? What are they?
- How can we compute them very fast?
- What’s the complexity of computing them?
- How many integral images do we need?
Algorithm for 2D BBox [Lampert et al. 06]

Algorithm 1 Efficient Subwindow Search

Require: image \( x \)

Require: quality bounding function \( \hat{f} \) (see Sect.III)

Ensure: \( (t_{opt}, b_{opt}, l_{opt}, r_{opt}) = \arg\max_{y \in \mathcal{Y}} f(y) \)

initialize \( P \) as empty priority queue
set \( [T, B, L, R] = [1, n] \times [1, n] \times [1, m] \times [1, m] \)

repeat
    split \( [T, B, L, R] \to [T_1, B_1, L_1, R_1] \cup [T_2, B_2, L_2, R_2] \)
    push \( ([T_1, B_1, L_1, R_1]; \hat{f}([T_1, B_1, L_1, R_1])) \) onto \( P \)
    push \( ([T_2, B_2, L_2, R_2]; \hat{f}([T_2, B_2, L_2, R_2])) \) onto \( P \)
    retrieve top state \( [T, B, L, R] \) from \( P \)
until \( [T, B, L, R] \) consists of only one rectangle
set \( (t_{opt}, b_{opt}, l_{opt}, r_{opt}) = [T, B, L, R] \)

- How do we split?

- When do we terminate?
Let’s go back to our problem.

We parameterize the sets by **intervals** of minimum and maximum angles:

\[
\{[y_1^{\text{min}}, y_1^{\text{max}}], \ldots, [y_4^{\text{min}}, y_4^{\text{max}}]\}
\]

The scoring function sums features over the faces:

\[
E(y_1, \ldots, y_4) = \sum_r w_r^T \phi(y_r, x) = \sum_{\alpha} f_{\alpha}(y, x)
\]

with \(\alpha = \{\text{floor}, \text{left}_w, \text{right}_w, \text{ceiling}, \text{front}_w\}\)

What about the bounds?
Bounds for 3D layout

- The scoring function sums features over the faces
  \[ E(y_1, \cdots, y_4) = \sum_r w_r^T \phi(y_r, x) = \sum_\alpha f_\alpha(y, x) \]

  with \( \alpha = \{\text{floor}, \text{left}_w, \text{right}_w, \text{ceiling}, \text{front}_w\} \)

- Let’s bound each ”face” \( \alpha \) separately

- Recall where the features come from

  - Some features are positive, some are negative. Why? How do I know which ones are positive/negative?
Deriving bounds

- Inference can be then done by
  \[ E(y_1, \cdots, y_4) = \sum_{\alpha} f_\alpha^+(x, y) + f_\alpha^-(x, y), \]

- We can bound each of this terms separately
  \[ \text{bound}(E(\hat{Y}, x)) = \sum_{\alpha \in \mathcal{F}} \hat{f}_\alpha^+(\hat{Y}, x) + \hat{f}_\alpha^-(\hat{Y}, x) \]

- We construct bounds by computing the max positive and min negative contribution of the score within the set \( \hat{Y} \) for each face \( \alpha \in \mathcal{F} \).
  \[ \hat{f}_{\text{front-wall}}(\hat{Y}) = f_{\text{front-wall}}^+(x, y_{up}) + f_{\text{front-wall}}^-(x, y_{low}), \]

(Front Wall) (Minimal left wall) (Maximal left wall)
Efficient bounds

- How can we compute the bounds efficiently?
Efficient bounds

- How can we compute the bounds efficiently?

- What’s the complexity?
Efficient bounds

- How can we compute the bounds efficiently?

- What’s the complexity?

- How many evaluations?
Efficient bounds

- How can we compute the bounds efficiently?

- What’s the complexity?

- How many evaluations?
Table: Pixel classification error in the layout dataset of [Hedau et al. 09].

<table>
<thead>
<tr>
<th></th>
<th>OM</th>
<th>GC</th>
<th>OM + GC</th>
<th>Other</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Hoiem07]</td>
<td>-</td>
<td>28.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Hedau09] (a)</td>
<td>-</td>
<td>26.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Hedau09] (b)</td>
<td>-</td>
<td>21.2</td>
<td>-</td>
<td>-</td>
<td>10-30 min</td>
</tr>
<tr>
<td>[Wang10]</td>
<td>22.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[Lee10]</td>
<td>24.7</td>
<td>22.7</td>
<td>18.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[delPero11]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16.3</td>
<td>12 min</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>18.6</strong></td>
<td><strong>15.4</strong></td>
<td><strong>13.6</strong></td>
<td>-</td>
<td><strong>0.007s</strong></td>
</tr>
</tbody>
</table>

Table: Pixel classification error in the bedroom data set [Hedau et al. 10].

<table>
<thead>
<tr>
<th></th>
<th>[delPero11]</th>
<th>[Hoiem07]</th>
<th>[Hedau09] (a)</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o box</td>
<td>29.59</td>
<td>23.04</td>
<td>22.94</td>
<td><strong>16.46</strong></td>
</tr>
</tbody>
</table>

- Takes on average **0.007s** for exact solution over $50^4$ possibilities!
- It’s 6 orders of magnitude faster than the state-of-the-art!
Qualitative Results
Conclusion:

- We have studied structured prediction including learning and inference
- We have investigated how to think to solve a real-world problem

Relations to previous two talks:

- RBMs are graphical models
- Your potentials $\phi_r(y_r)$ can be "deep"

Open questions:

- Latent variable models: non-convex learning
- Learn the structure of the graph
- Go beyond log-linear models
- MAP inference: high order potentials
- Continuous Markov random fields

If you are interested in doing research at University of Toronto, talk to me!