Linear Programming Decoders in NLP:

Integer Programming, Message Passing, Dual Decomposition

André Martins

priberam   instituto de telecomunicações

EMNLP 2014: Tutorials, Doha, Qatar, October 25, 2014

Slides online at http://tiny.cc/lpdnlp.

Outline

Structured Prediction and NLP

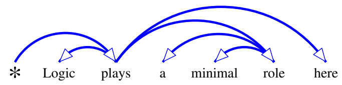
Structured prediction: a machine learning framework for predicting structured, constrained, and interdependent outputs

NLP deals with *structured* and *ambiguous* textual data (Smith, 2011):

- machine translation
- speech recognition
- syntactic parsing
- semantic parsing
- information extraction
- ...

Dependency Parsing

Map **sentences** to their **syntactic structure.**

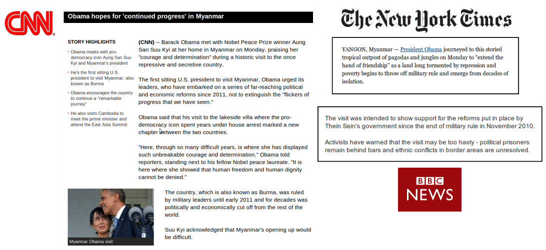


- A lexicalized syntactic formalism
- Grammar functions represented as lexical relationships (dependencies)

(Eisner, 1996; McDonald et al., 2005; Nivre et al., 2006; Koo et al., 2007)

Multi-Document Summarization

Map a set of related **documents** to a brief **summary.**



Multi-Document Summarization

Map a set of related **documents** to a brief **summary.**

Current State of Affairs

1. Greedy algorithms can deal with rich histories, but they are suboptimal and suffer from error propagation
2. Simple, tractable models permit exact decoding, but they make too stringent factorization assumptions

Current State of Affairs

1. Greedy algorithms can deal with rich histories, but they are suboptimal and suffer from error propagation
2. Simple, tractable models permit exact decoding, but they make too stringent factorization assumptions

We'd like *fast* predictors with *global* features and constraints, but how?

Related Recent Tutorials

- Dual Decomposition and Lagrangian Relaxation for Inference in NLP (Rush & Collins ACL 2011)
- Structured Predictions in NLP: Constrained Conditional Models and Integer Linear Programming (Srikumar, Goldwasser & Roth NAACL 2012)
- Variational Inference in Structured NLP Models (Burkett & Klein NAACL 2012)
- Structured Belief Propagation for NLP (Gromley & Eisner ACL 2014)

This Tutorial: Linear Programming Decoders

We'll provide a unified view over these approaches (ILPs, message-passing, dual decomposition)

We'll focus on MAP decoding, but touch briefly on marginal decoding

We'll illustrate with three applications:

1. Turbo Parsing
2. Compressive Summarization
3. Joint Coreference Resolution and Quote Attribution

(Companion software: AD^3 toolkit)

Outline

Structured Prediction

- Input set $\mathcal{X}$
- For each $x \in \mathcal{X}$: a **large** set of candidate outputs $\mathcal{Y}(x)$
- A compatibility function $F_{\boldsymbol{w}}(x, y)$ induced by a model $\boldsymbol{w}$ (Linear model: $F_{\boldsymbol{w}}(x, y) = \boldsymbol{w}^\top \boldsymbol{f}(x, y)$)

Structured Prediction

- Input set $\mathcal{X}$
- For each $x \in \mathcal{X}$: a **large** set of candidate outputs $\mathcal{Y}(x)$
- A compatibility function $F_{\boldsymbol{w}}(x, y)$ induced by a model $\boldsymbol{w}$ (Linear model: $F_{\boldsymbol{w}}(x, y) = \boldsymbol{w}^\top \boldsymbol{f}(x, y)$)
- **Training problem:** learn the model $\boldsymbol{w}$ from data $\{\langle x_i, y_i \rangle\}_{i=1}^M$
- **Decoding problem (our focus):**

$$\widehat{y} = \arg \max_{y \in \mathcal{Y}(x)} F_{\boldsymbol{w}}(x, y)$$

Structured Prediction

- Input set $\mathcal{X}$
- For each $x \in \mathcal{X}$: a **large** set of candidate outputs $\mathcal{Y}(x)$
- A compatibility function $F_{\boldsymbol{w}}(x, y)$ induced by a model $\boldsymbol{w}$ (Linear model: $F_{\boldsymbol{w}}(x, y) = \boldsymbol{w}^\top \boldsymbol{f}(x, y)$)
- **Training problem:** learn the model $\boldsymbol{w}$ from data $\{\langle x_i, y_i \rangle\}_{i=1}^M$
- **Decoding problem (our focus):**

$$\widehat{y} = \arg \max_{y \in \mathcal{Y}(x)} F_{\boldsymbol{w}}(x, y)$$

- **Key assumption:** $F_{\boldsymbol{w}}$ decomposes into (overlapping) *parts*

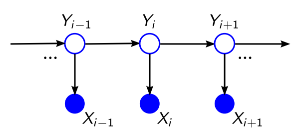
Three Important Questions

- Representation?
- Decoding/Inference?
- Learning the parameters?

Recap: Hidden Markov Models

$F_{\boldsymbol{w}}$ is a log-probability, factoring over emissions and transitions.

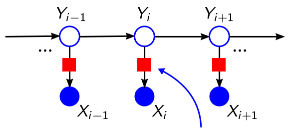
$$\mathbb{P}(x, y) = \prod_i \underbrace{\mathbb{P}(x_i | y_i)}_{\text{emissions}} \underbrace{\mathbb{P}(y_i | y_{i-1})}_{\text{transitions}}$$



Recap: Hidden Markov Models

$F_{\boldsymbol{w}}$ is a log-probability, factoring over emissions and transitions.

$$\mathbb{P}(x,y) = \prod_i \underbrace{\mathbb{P}(x_i|y_i)}_{\text{emissions}} \underbrace{\mathbb{P}(y_i|y_{i-1})}_{\text{transitions}}$$



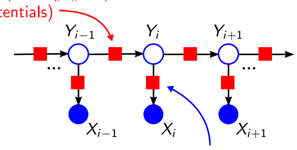
$\psi_i(y_i) := \mathbb{P}(x_i|y_i)$ (unary potentials)

Recap: Hidden Markov Models

$F_{\boldsymbol{w}}$ is a log-probability, factoring over emissions and transitions.

$$\mathbb{P}(x,y) = \prod_i \underbrace{\mathbb{P}(x_i|y_i)}_{\text{emissions}} \underbrace{\mathbb{P}(y_i|y_{i-1})}_{\text{transitions}}$$

$\psi_{i,i-1}(y_i, y_{i-1}) := \mathbb{P}(y_i|y_{i-1})$
(pairwise potentials)



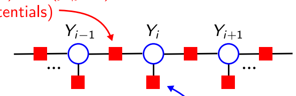
$\psi_i(y_i) := \mathbb{P}(x_i|y_i)$ (unary potentials)

Recap: Hidden Markov Models

$F_{\boldsymbol{w}}$ is a log-probability, factoring over emissions and transitions.

$$\mathbb{P}(x,y) = \prod_i \underbrace{\mathbb{P}(x_i|y_i)}_{\text{emissions}} \underbrace{\mathbb{P}(y_i|y_{i-1})}_{\text{transitions}}$$

$\psi_{i,i-1}(y_i, y_{i-1}) := \mathbb{P}(y_i|y_{i-1})$
(pairwise potentials)



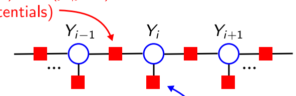
$\psi_i(y_i) := \mathbb{P}(x_i|y_i)$ (unary potentials)

Recap: Hidden Markov Models

$F_{\boldsymbol{w}}$ is a log-probability, factoring over emissions and transitions.

$$\mathbb{P}(x,y) = \prod_i \underbrace{\mathbb{P}(x_i|y_i)}_{\text{emissions}} \underbrace{\mathbb{P}(y_i|y_{i-1})}_{\text{transitions}} = \prod_i \psi_i(y_i) \prod_i \psi_{i,i-1}(y_i, y_{i-1})$$

$\psi_{i,i-1}(y_i, y_{i-1}) := \mathbb{P}(y_i|y_{i-1})$
(pairwise potentials)



$\psi_i(y_i) := \mathbb{P}(x_i|y_i)$ (unary potentials)

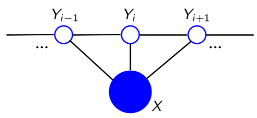
Recap: Hidden Markov Models

- Representation? **Directed sequence model.**
- Decoding/Inference? **Viterbi/forward-backward algorithms.**
- Learning the parameters? **Maximum likelihood (count and normalize).**

Recap: Conditional Random Fields

Same factorization, but globally normalized.

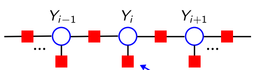
$$\mathbb{P}(y|x) = \frac{1}{Z(\boldsymbol{w},x)} \exp\left(\sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_i(x,y_i)}_{\text{nodes}} + \sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_{i,i-1}(x,y_i,y_{i-1})}_{\text{edges}}\right)$$



Recap: Conditional Random Fields

Same factorization, but globally normalized.

$$\mathbb{P}(y|x) = \frac{1}{Z(\boldsymbol{w},x)} \exp\left(\sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_i(x,y_i)}_{\text{nodes}} + \sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_{i,i-1}(x,y_i,y_{i-1})}_{\text{edges}}\right)$$



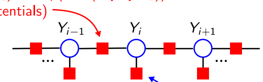
$\psi_i(y_i) := \exp(\mathbf{w}^\top \mathbf{f}(x, y_i))$ (unary potentials)

Recap: Conditional Random Fields

Same factorization, but globally normalized.

$$\mathbb{P}(y|x) = \frac{1}{Z(\boldsymbol{w},x)} \exp\left(\sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_i(x,y_i)}_{\text{nodes}} + \sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_{i,i-1}(x,y_i,y_{i-1})}_{\text{edges}}\right)$$

$\psi_{i,i-1}(y_i, y_{i-1}) := \exp(\mathbf{w}^\top \mathbf{f}(x, y_i, y_{i-1}))$
(pairwise potentials)



$\psi_i(y_i) := \exp(\mathbf{w}^\top \mathbf{f}(x, y_i))$ (unary potentials)

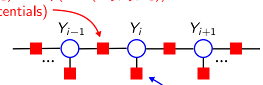
Recap: Conditional Random Fields

Same factorization, but globally normalized.

$$\mathbb{P}(y|x) = \frac{1}{Z(\boldsymbol{w},x)} \exp\left(\sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_i(x,y_i)}_{\text{nodes}} + \sum_i \underbrace{\boldsymbol{w}^\top \boldsymbol{f}_{i,i-1}(x,y_i,y_{i-1})}_{\text{edges}}\right)$$

$$\propto \prod_i \psi_i(y_i) \prod_i \psi_{i,i-1}(y_i, y_{i-1})$$

$\psi_{i,i-1}(y_i, y_{i-1}) := \exp(\mathbf{w}^\top \mathbf{f}(x, y_i, y_{i-1}))$
(pairwise potentials)



$\psi_i(y_i) := \exp(\mathbf{w}^\top \mathbf{f}(x, y_i))$ (unary potentials)

Recap: Conditional Random Fields

- Representation? **Undirected sequence model.**
- Decoding/Inference? **Viterbi/forward-backward algorithms.**
- Learning the parameters? **Maximum conditional likelihood (convex optimization).**

Graphical Models

HMMs and CRFs are two instances of *graphical models*.

In general, graphical models come in two flavours:

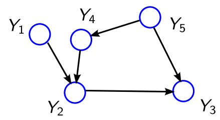
- Directed (Bayesian Networks)
- Undirected (Markov Networks)

Bayesian Networks

Useful to express **causality relations**.

Factors are **conditional probability tables**.

$$\mathbb{P}(y) = \mathbb{P}(y_1)\mathbb{P}(y_2|y_1,y_4)\mathbb{P}(y_3|y_2,y_5)\mathbb{P}(y_4|y_5)\mathbb{P}(y_5)$$



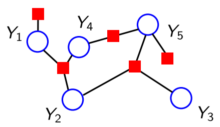
$$\boxed{\mathbb{P}(y) = \prod_i \mathbb{P}(y_i|\text{parents}(y_i))}$$

Bayesian Networks

Useful to express **causality relations**.

Factors are **conditional probability tables**.

$$\mathbb{P}(y) = \mathbb{P}(y_1)\mathbb{P}(y_2|y_1,y_4)\mathbb{P}(y_3|y_2,y_5)\mathbb{P}(y_4|y_5)\mathbb{P}(y_5)$$



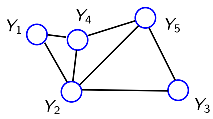
$$\boxed{\mathbb{P}(y) = \prod_i \mathbb{P}(y_i|\text{parents}(y_i))}$$

Markov Networks

Useful to express **correlations between variables**.

Factors correspond to **cliques of the graph**.

$$\mathbb{P}(y) = \tfrac{1}{Z}\psi_{124}(y_1,y_2,y_4)\psi_{235}(y_2,y_3,y_5)\psi_{245}(y_2,y_4,y_5)$$



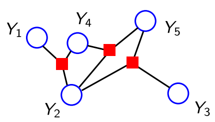
$$\boxed{\mathbb{P}(y) \propto \prod_{s \in \text{cliques}(G)} \psi_s(\boldsymbol{y}_s)}$$

Markov Networks

Useful to express **correlations between variables**.

Factors correspond to **cliques of the graph**.

$$\mathbb{P}(y) = \tfrac{1}{Z}\psi_{124}(y_1,y_2,y_4)\psi_{235}(y_2,y_3,y_5)\psi_{245}(y_2,y_4,y_5)$$



$$\boxed{\mathbb{P}(y) \propto \prod_{s \in \text{cliques}(G)} \psi_s(\boldsymbol{y}_s)}$$

Conditional Independence

Graphical models are a great tool for modeling conditional independence

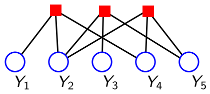
They link properties of the probability distribution with properties of the graph (reachability, D-separation, etc.)

Lots of literature about this: Pearl (1988); Lauritzen (1996); Koller and Friedman (2009)

An Intermediate Representation: Factor Graph

A bipartite graph with **variable nodes** and **factor nodes**

It makes *explicit* the factors of the distribution



$$\mathbb{P}(y) \propto \underbrace{\prod_i \psi_i(y_i)}_{\text{unary potentials}} \times \underbrace{\prod_s \psi_s(\boldsymbol{y}_s)}_{\text{higher-order potentials}}$$

With unary potentials only, all variables would be independent

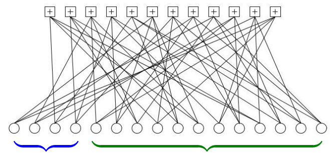
Higher-order potentials can model correlations, impose soft/hard constraints, etc.

Example: Low-Density Parity Check Codes

A message is transmitted through a noisy channel, corrupting some bits

Redundancy can help decoding the message, e.g. via additional parity check bits that can detect/correct errors (error-correcting codes)

High-level idea: increase redundancy to build more accurate decoders



(Adapted from MacKay 2003.)

Inference/Decoding

$$\mathbb{P}_\psi(y|x) = \frac{1}{Z(\psi, x)} \times \prod_i \psi_i(y_i) \times \prod_s \psi_s(\boldsymbol{y}_s)$$

Two decoding problems:

- **MAP decoding:** compute $\widehat{y} = \arg\max_y \mathbb{P}_\psi(y|x)$
- **Marginal decoding:** compute every $\mathbb{P}_\psi(y_i|x)$ and $\mathbb{P}_\psi(\boldsymbol{y}_s|x)$; and evaluate the partition function $Z(\psi, x)$

Sometimes easy, in general intractable (we'll elaborate more)

When is Decoding Easy?

- independent variables (trivial)
- sequence models (Viterbi, forward-backward)
- graphical models without cycles (variable elimination, belief propagation)
- graphical models with low treewidth (junction tree algorithm)

When is Decoding Easy?

- independent variables (trivial)
- sequence models (Viterbi, forward-backward)
- graphical models without cycles (variable elimination, belief propagation)
- graphical models with low treewidth (junction tree algorithm)

In general, for graphs with cycles, MAP decoding is NP-hard and marginal decoding is #P-hard

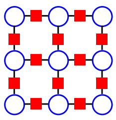
When is Decoding Easy?

- independent variables (trivial)
- sequence models (Viterbi, forward-backward)
- graphical models without cycles (variable elimination, belief propagation)
- graphical models with low treewidth (junction tree algorithm)

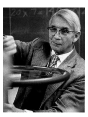
In general, for graphs with cycles, MAP decoding is NP-hard and marginal decoding is #P-hard

Note: tractability depends not only on the topology, but also on the *potentials*

Example: Ising and Potts Models



Ising/Potts grid

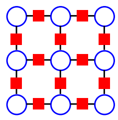


Ernst Ising, 1900–1998

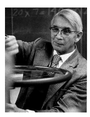
Ren Potts, 1925–2005

All factors are pairwise, variables are binary (Ising) or multi-class (Potts)

Example: Ising and Potts Models



Ising/Potts grid



Ernst Ising, 1900–1998

Ren Potts, 1925–2005

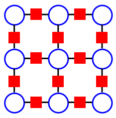
All factors are pairwise, variables are binary (Ising) or multi-class (Potts)

MAP decoding is tractable for *attractive* Ising models (i.e. Ising models with *supermodular* log-potentials):

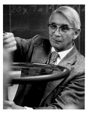
$$\log \psi_{ij}(1,1) + \log \psi_{ij}(0,0) \geq \log \psi_{ij}(0,1) + \log \psi_{ij}(1,0)$$

Good approximations for *attractive* Potts models

Example: Ising and Potts Models



Ising/Potts grid



Ernst Ising, 1900–1998

Ren Potts, 1925–2005

All factors are pairwise, variables are binary (Ising) or multi-class (Potts)

MAP decoding is tractable for *attractive* Ising models (i.e. Ising models with *supermodular* log-potentials):

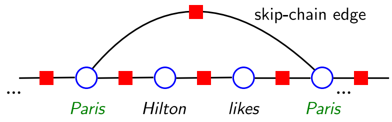
$$\log \psi_{ij}(1,1) + \log \psi_{ij}(0,0) \geq \log \psi_{ij}(0,1) + \log \psi_{ij}(1,0)$$

Good approximations for *attractive* Potts models

... but the general case is NP-hard and hard to approximate

Example: Skip-Chain CRFs

Skip-chain CRFs are useful to model long-range dependencies



Skip-chains introduce cycles, making decoding more expensive

We could write this information in the "state" and still decode with dynamic programming, but that would blow up the number of states

Beyond Graphical Models

Some NLP problems (e.g. parsing) require representations beyond graphical models

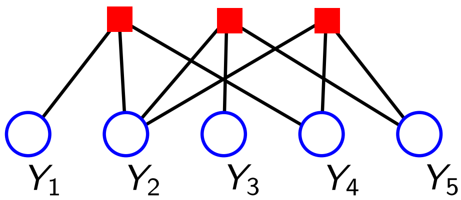
Dynamic programming algorithms (CKY, inside-outside) still work for those representations

Example: case-factor diagrams (McAllester et al., 2008)

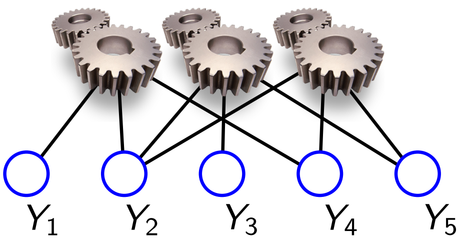
Other problems (e.g. matching, spanning trees) can be solved with combinatorial algorithms not related with dynamic programming

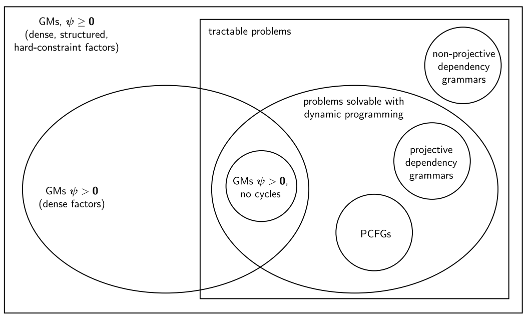
All these can still be represented as GMs by "generalizing" the notion of factor

Factors as Machines

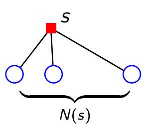


Factors as Machines





Three Kinds of Factors



Let $N(s)$ denote the set of variables that are *neighbors* of factor s.

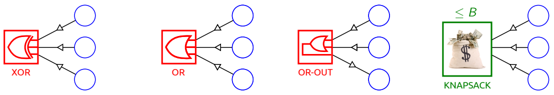
1. **Dense factors:** $O(\exp(|N(s)|))$ degrees of freedom
2. **Structured factors:** $\psi_s(\boldsymbol{y}_s)$ has internal structure
3. **Hard constraint factors:**

$$\psi_s(\boldsymbol{y}_s) := \begin{cases} 1, & \text{if } \boldsymbol{y}_s \in \mathcal{Y}_s \\ 0, & \text{otherwise.} \end{cases}$$

Examples of Structured Factors

- a factor for bipartite matching (Duchi et al., 2007)
- combining a sequential model (POS tagger) with a PCFG (Rush et al., 2010)
- combining CCG parsing and supertagging (Auli and Lopez, 2011)
- dependency parsing with head automata (Smith and Eisner, 2008; Koo et al., 2010)
- handling string-valued variables with factors that are finite state transducers (Dreyer and Eisner, 2009)
- inversion transduction grammar constraint (Burkett and Klein, 2012)

Examples of Hard Constraint Factors



Logic factors: can express arbitrary FOL constraints

- Applications: Markov logic networks (Richardson and Domingos, 2006), constrained conditional models (Roth and Yih, 2004)

Knapsack factors: can express budget constraints

- Applications: summarization, diversity problems,...

(Martins et al., 2011b; Almeida and Martins, 2013; Martins et al., 2014)

Approximate Decoding

What to do when exact decoding is intractable?

- Sampling methods (MCMC, etc.)
- Mean field algorithms
- LP relaxations
- Message-passing
- Dual decomposition

We'll highlight connections between several of these methods.

Approximate Decoding

What to do when exact decoding is intractable?

- Sampling methods (MCMC, etc.)
- Mean field algorithms
- LP relaxations
- Message-passing
- Dual decomposition

We'll highlight connections between several of these methods.

Global/Local Decoding

"Local" denotes independent problems within the scope of each factor

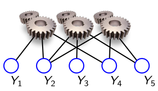
"Global" involves a global assignment of variables, consistent across factors

Key idea: "glue" the local evidence at the factors to obtain a global assignment

Our assumption: local decoding is easy, for every factor

We want to build a good (approximate) *global* decoder by invoking the *local* decoders.

What Kind of Local Decoding Do We Need?

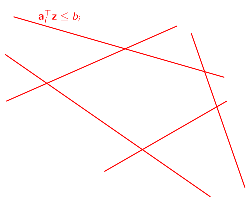


Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Outline

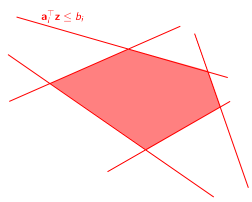
Linear Programming (Kantorovich, 1940; Dantzig, 1947)

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \textbf{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i, \; i = 1, \ldots, N. & \textbf{Linear constraints} \end{array}$$



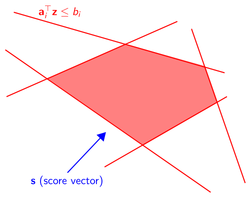
Linear Programming (Kantorovich, 1940; Dantzig, 1947)

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \textbf{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i, \; i = 1, \ldots, N. & \textbf{Linear constraints} \end{array}$$



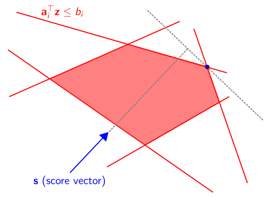
Linear Programming (Kantorovich, 1940; Dantzig, 1947)

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N. & \text{Linear constraints} \end{array}$$



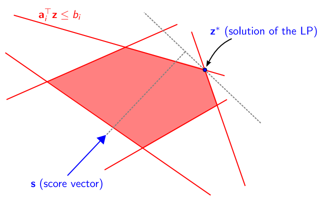
Linear Programming (Kantorovich, 1940; Dantzig, 1947)

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N. & \text{Linear constraints} \end{array}$$



Linear Programming (Kantorovich, 1940; Dantzig, 1947)

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N. & \text{Linear constraints} \end{array}$$

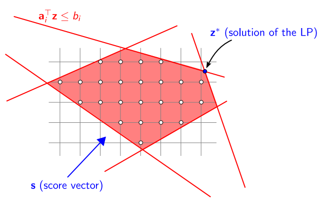


Linear Programming (Kantorovich, 1940; Dantzig, 1947)

- If feasible and bounded, the solution is always attained at a vertex
- Can be solved in **polynomial time** (Khachiyan, 1980)
- Lots of off-the-shelf solvers (CPLEX, Gurobi, GLPK, LP_Solve, etc.)

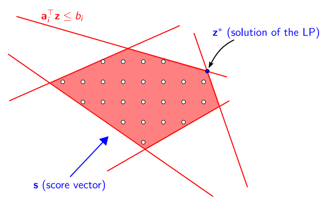
Integer Linear Programming

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N, & \text{Linear constraints} \\ & \boldsymbol{z} \text{ integer.} & \end{array}$$



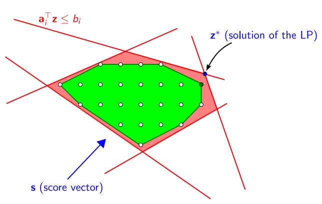
Integer Linear Programming

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N, & \text{Linear constraints} \\ & \boldsymbol{z} \text{ integer.} & \end{array}$$



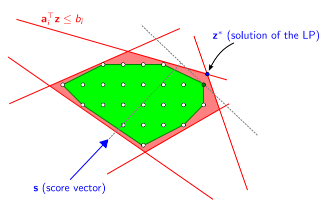
Integer Linear Programming

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N, & \text{Linear constraints} \\ & \boldsymbol{z} \text{ integer.} & \end{array}$$



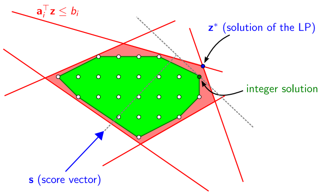
Integer Linear Programming

$$\begin{array}{lll} \max_{\boldsymbol{z}} & \boldsymbol{s}^\top \boldsymbol{z} & \text{Linear objective} \\ \text{s.t.} & \boldsymbol{a}_i^\top \boldsymbol{z} \le b_i,\ i = 1, \ldots, N, & \text{Linear constraints} \\ & \boldsymbol{z} \text{ integer.} & \end{array}$$



Integer Linear Programming

$$\max_{z} \; s^\top z \qquad \text{Linear objective}$$
$$\text{s.t.} \; {a_i}^\top z \leq b_i, \; i = 1, \ldots, N, \qquad \text{Linear constraints}$$
$$z \text{ integer.}$$



Integer Linear Programming

In general, NP-hard (Karp, 1972)

Existing solvers are effective for small instances, but don't scale

LP relaxation: drops the integer constraints

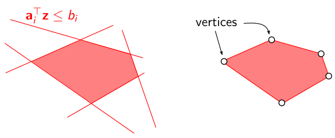
- Gives an upper bound of the solution of the ILP
- A common first step in exact algorithms (branch-and-bound, cutting plane, branch-and-cut)

Here's a very simple approximate algorithm:

1. Solve the LP relaxation
2. If the solution is integer, then it *is* the solution of the ILP
3. Otherwise, apply a rounding heuristic (problem-dependent)

Two Representations of Polytopes

Intersection of half-spaces (**H-representation**) or convex hull of a set of vertices (**V-representation**)

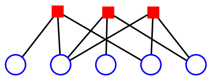


To call a solver, we need to specify a concise H-representation

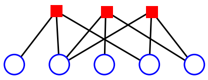
However, it may be difficult or impossible to obtain one if all we have is a V-representation

We next show how this relates to MAP decoding...

Structured Outputs as Bit-Vectors

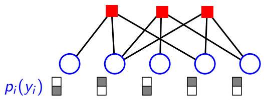


Structured Outputs as Bit-Vectors



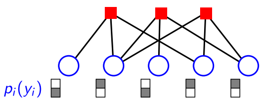
- One indicator $p_i(y_i)$ per each variable state

Structured Outputs as Bit-Vectors



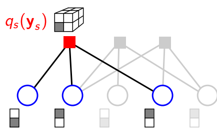
- One indicator $p_i(y_i)$ per each variable state

Structured Outputs as Bit-Vectors



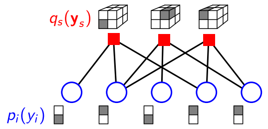
- One indicator $p_i(y_i)$ per each variable state
- One indicator $q_s(\boldsymbol{y}_s)$ per each factor configuration

Structured Outputs as Bit-Vectors



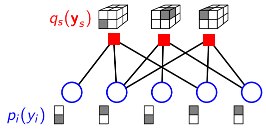
- One indicator $p_i(y_i)$ per each variable state
- One indicator $q_s(\boldsymbol{y}_s)$ per each factor configuration

Structured Outputs as Bit-Vectors



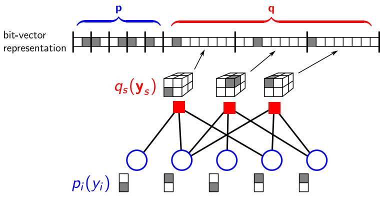
- One indicator $p_i(y_i)$ per each variable state
- One indicator $q_s(\mathbf{y}_s)$ per each factor configuration

Structured Outputs as Bit-Vectors



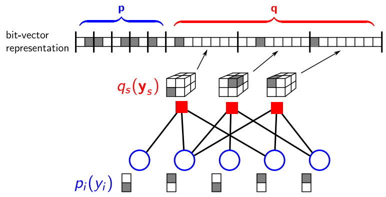
- One indicator $p_i(y_i)$ per each variable state
- One indicator $q_s(\mathbf{y}_s)$ per each factor configuration
- Overall: each global output $y \in \mathcal{Y}(x)$ is mapped to a bit-vector

Structured Outputs as Bit-Vectors



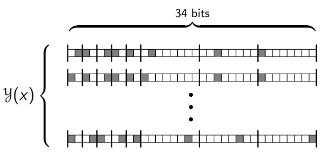
- One indicator $p_i(y_i)$ per each variable state
- One indicator $q_s(\mathbf{y}_s)$ per each factor configuration
- Overall: each global output $y \in \mathcal{Y}(x)$ is mapped to a bit-vector

Structured Outputs as Bit-Vectors

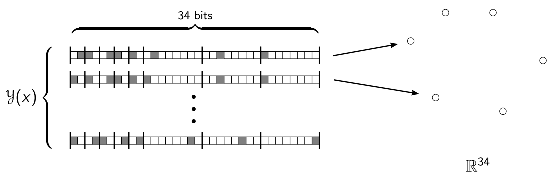


- One indicator $p_i(y_i)$ per each variable state
- One indicator $q_s(\mathbf{y}_s)$ per each factor configuration
- Overall: each global output $y \in \mathcal{Y}(x)$ is mapped to a bit-vector
- **Note: not all bit vectors are valid (they must be consistent)**

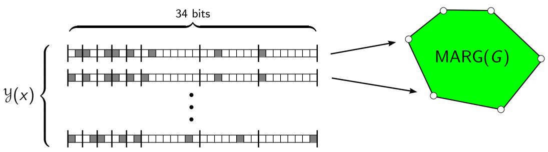
Marginal Polytope (Wainwright and Jordan, 2008)



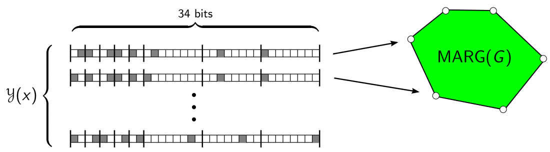
Marginal Polytope (Wainwright and Jordan, 2008)



Marginal Polytope (Wainwright and Jordan, 2008)

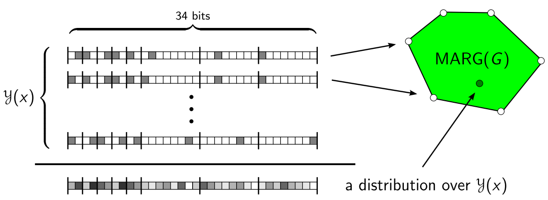


Marginal Polytope (Wainwright and Jordan, 2008)



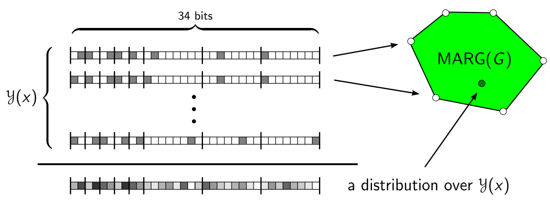
- Vertices of **MARG**(G) correspond to outputs $\mathcal{Y}(x)$
- Points of **MARG**(G) correspond to realizable marginals (more later)

Marginal Polytope (Wainwright and Jordan, 2008)



- Vertices of **MARG(G)** correspond to outputs $\mathcal{Y}(x)$
- Points of **MARG(G)** correspond to realizable marginals (more later)

Marginal Polytope (Wainwright and Jordan, 2008)



- Vertices of **MARG(G)** correspond to outputs $\mathcal{Y}(x)$
- Points of **MARG(G)** correspond to realizable marginals (more later)
- This is a V-representation, what about an H-representation?

H-Representation With Integer Constraints

In general, there's no concise H-representation for MARG(G)

... but we can represent its vertices if *integer constraints* are permitted:

$$\sum_{\boldsymbol{y}_s} q_s(\boldsymbol{y}_s) = 1, \quad q_s(\boldsymbol{y}_s) \geq \mathbf{0}, \quad \forall \boldsymbol{y}_s \in \mathcal{Y}_s \qquad \textbf{(normalization)}$$

$$p_i(y_i) = \sum_{\boldsymbol{y}_s \sim y_i} q_s(\boldsymbol{y}_s), \quad \forall i \in N(s) \qquad \textbf{(marginalization)}$$

$$\boldsymbol{q} \text{ is integer} \qquad \textbf{(integer constraints)}$$

This will open the door for formulating MAP decoding as an ILP.

MAP Decoding as an ILP

Recall the MAP decoding problem:

$$\begin{aligned}\widehat{y} &= \arg\max_{y \in \mathcal{Y}(x)} P_\psi(y|x) \\ &= \arg\max_{y \in \mathcal{Y}(x)} \frac{1}{Z(\psi, x)} \prod_i \psi_i(y_i) \prod_s \psi_s(\boldsymbol{y}_s) \\ &= \arg\max_{y \in \mathcal{Y}(x)} \sum_i \boldsymbol{\theta}_i(y_i) + \sum_s \boldsymbol{\theta}_s(\boldsymbol{y}_s),\end{aligned}$$

where $\boldsymbol{\theta}_i(y_i) := \log \psi_i(y_i)$ and $\boldsymbol{\theta}_s(\boldsymbol{y}_s) := \log \psi_s(\boldsymbol{y}_s)$

We can rewrite this as an ILP:

$$\text{maximize} \sum_i \sum_{y_i} \boldsymbol{\theta}_i(y_i) p_i(y_i) + \sum_s \sum_{\boldsymbol{y}_s} \boldsymbol{\theta}_s(\boldsymbol{y}_s) q_s(\boldsymbol{y}_s)$$
$$\text{subject to } (p, q) \in \text{MARG}(G)$$

Local Polytope

Obtained by relaxing the integer constraints

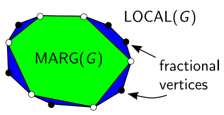
Regard $\boldsymbol{p}_i$ and $\boldsymbol{q}_s$ as probability distributions that must be **locally consistent**:

$$\sum_{\boldsymbol{y}_s} q_s(\boldsymbol{y}_s) = 1, \quad q_s(\boldsymbol{y}_s) \geq \mathbf{0}, \quad \forall \boldsymbol{y}_s \in \mathcal{Y}_s \qquad \textbf{(normalization)}$$

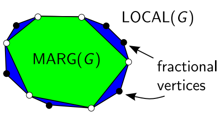
$$p_i(y_i) = \sum_{\boldsymbol{y}_s \sim y_i} q_s(\boldsymbol{y}_s), \quad \forall i \in N(s) \qquad \textbf{(marginalization)}$$

The feasible points are *pseudo-marginals* (not necessarily valid marginals)

Local and Marginal Polytopes



Local and Marginal Polytopes



- LOCAL(G) is an **outer bound** of MARG(G)
- It contains all the integer vertices of MARG(G), plus spurious **fractional vertices**
- **If the graph has no cycles, then LOCAL(G) = MARG(G)**

LP-MAP Decoding

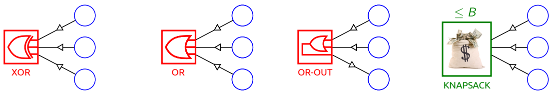
Solves a linear relaxation of MAP decoding, replacing MARG(G) by LOCAL(G):

$$\text{maximize} \sum_i \sum_{y_i} \boldsymbol{\theta}_i(y_i) p_i(y_i) + \sum_s \sum_{\boldsymbol{y}_s} \boldsymbol{\theta}_s(\boldsymbol{y}_s) q_s(\boldsymbol{y}_s)$$
$$\text{subject to } (p, q) \in \text{LOCAL}(G)$$

If the solution is integer, we solved the problem exactly; otherwise, apply a rounding heuristic

Runtime is polynomial, but the procedure is only approximate.

What About Hard Constraint Factors?



Logic and knapsack/budget constraints can also be expressed *linearly*

Logic/Budget Constraints

Assume $z_1, z_2, \ldots \in \{0, 1\}$ (binary variables)

Condition	Statement	Constraint
Implication	if z_1 then z_2	$z_1 \leq z_2$
Negation	z_1 iff $\neg z_2$	$z_1 = 1 - z_2$
OR	z_1 or z_2 or z_3	$z_1 + z_2 + z_3 \geq 1$
XOR	z_1 xor z_2 xor z_3	$z_1 + z_2 + z_3 = 1$
OR-OUT	$z_{12} = z_1 \vee z_2$	$z_{12} \geq z_1,\ z_{12} \geq z_2,$ $z_{12} \leq z_1 + z_2$
AND-OUT	$z_{12} = z_1 \wedge z_2$	$z_{12} \leq z_1,\ z_{12} \leq z_2,$ $z_{12} \geq z_1 + z_2 - 1$
Budget	at most B active units	$\sum_i z_i \leq B$
Knapsack	at most B total weight	$\sum_i w_i z_i \leq B$

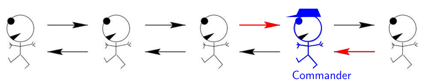
More complex expressions via composition and De Morgan's laws

Summing Up ILPs

- MAP decoding can be expressed as an Integer Linear Program (ILP)
- Logic constraints can be incorporated easily
- The ILP can be relaxed for approximate decoding (LP-MAP)
- Geometrically: an outer bound of the *marginal polytope*
- The relaxation is tight if the graph G does not have cycles
- **Disadvantage: an off-the-shelf LP solver won't exploit the *modularity* of the problem**
- **Algorithms that exploit the structure of the LP will be the topic of the remaining sections**

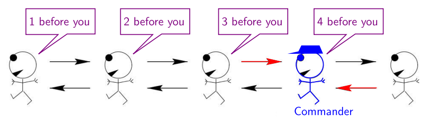
Outline

Motivating Example: Counting Soldiers



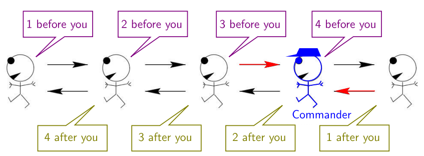
(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Motivating Example: Counting Soldiers



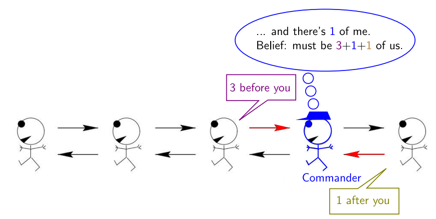
(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Motivating Example: Counting Soldiers



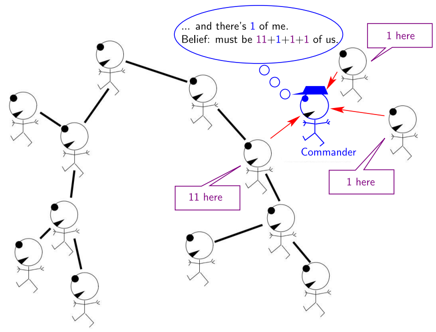
(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Motivating Example: Counting Soldiers



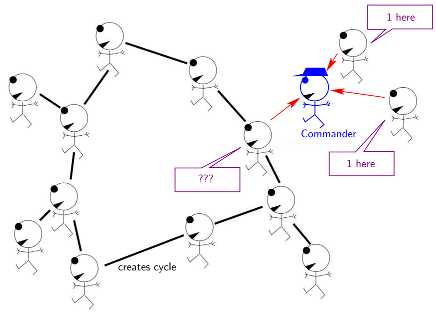
(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Motivating Example: Counting Soldiers



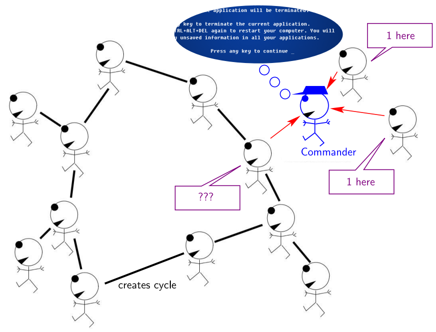
(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Motivating Example: Counting Soldiers



(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Motivating Example: Counting Soldiers



(Adapted from MacKay 2003 and Gormley & Eisner ACL'14 tutorial.)

Outline

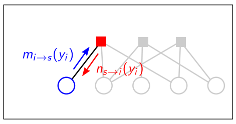
1. Structured Prediction and Factor Graphs
2. Integer Linear Programming
3. Message-Passing Algorithms
   - Sum-Product
   - Max-Product
4. Dual Decomposition
5. Applications
6. Conclusions

Sum-Product Belief Propagation

Recall that $\mathbb{P}_\psi(y|x) := \dfrac{1}{Z(\psi, x)} \times \prod_i \psi_i(y_i) \times \prod_s \psi_s(\boldsymbol{y}_s)$

Alternate between computing two kinds of messages:

- **Variable-to-factor:** $m_{i\to s}(y_i) = \psi_i(y_i) \prod_{s' \in N(i)\setminus\{s\}} n_{s'\to i}(y_i)$
- **Factor-to-variable:** $n_{s\to i}(y_i) = \sum_{\boldsymbol{y}_s \sim y_i} \psi_s(\boldsymbol{y}_s) \prod_{j\in N(s)\setminus\{i\}} m_{j\to s}(y_j)$



Beliefs

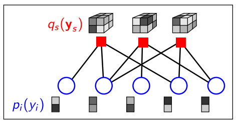
Given the messages, we compute local *beliefs*:

- **Variable beliefs:**

$$p_i(y_i) \propto \psi_i(y_i) \prod_{s\in N(i)} n_{s\to i}(y_i)$$

- **Factor beliefs:**

$$q_s(\boldsymbol{y}_s) \propto \psi_s(\boldsymbol{y}_s) \prod_{i\in N(s)} m_{i\to s}(y_i)$$



If the graph has no cycles, these beliefs converge to the true marginals

$$p_i(y_i) \to \mathbb{P}_\psi(y_i|x), \quad q_s(\boldsymbol{y}_s) \to \mathbb{P}_\psi(\boldsymbol{y}_s|x)$$

Otherwise: loopy BP (later)

Belief Propagation as Calibration

- **Variable-to-factor messages:**

$$m_{i\to s}(y_i) = \psi_i(y_i) \prod_{s'\in N(i)\setminus\{s\}} n_{s'\to i}(y_i) = \frac{p_i(y_i)}{n_{s\to i}(y_i)}$$

- **Factor-to-variable messages:**

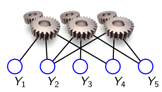
$$n_{s\to i}(y_i) = \sum_{\boldsymbol{y}_s\sim y_i} \psi_s(\boldsymbol{y}_s) \prod_{j\in N(s)\setminus\{i\}} m_{j\to s}(y_j) = \frac{\sum_{\boldsymbol{y}_s\sim y_i} q_s(\boldsymbol{y}_s)}{m_{i\to s}(y_i)}$$

- Calibration equations (attained at convergence):

$$p_i(y_i) = \sum_{\boldsymbol{y}_s\sim y_i} q_s(\boldsymbol{y}_s)$$

Punchline: to run sum-product BP, only need local marginals

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Loopy Belief Propagation

What if the graph has cycles?

Loopy Belief Propagation

What if the graph has cycles?

We'll see that marginal decoding corresponds to optimizing a *free energy objective* over the *marginal polytope*

Sum-product "loopy" BP entails two approximations:

1. Replaces MARG(G) by LOCAL(G)
2. Optimizes a *Bethe free energy* approximation

Step #1: Dual Parametrization

For any ψ, there are marginals $\boldsymbol{p}, \boldsymbol{q}$ in MARG(G) that parametrize $\mathbb{P}_\psi$

E.g. if the graph has no cycles:

$$\begin{aligned}\mathbb{P}_\psi(y|x) &= \frac{1}{Z(\psi,x)}\prod_i \psi_i(y_i) \times \prod_s \psi_s(\boldsymbol{y}_s)\\ &= \prod_i p_i(y_i)^{1-|N(i)|} \times \prod_s q_s(\boldsymbol{y}_s) \qquad (*)\\ &:= \mathbb{P}_{\boldsymbol{p},\boldsymbol{q}}(y|x)\end{aligned}$$

Therefore: a distribution can be represented as a point in MARG(G)

$\boldsymbol{\theta} := \log(\psi)$ are called *canonical parameters*, and $(\boldsymbol{p}, \boldsymbol{q})$ *mean parameters*

(*) Derivation of Dual Parametrization

Assume a tree-shaped Bayes net (each variable i has a single parent π_i)

$$\begin{aligned}\mathbb{P}(y) &= \mathbb{P}(y_0)\prod_{i\neq 0}\mathbb{P}(y_i|y_{\pi_i})\\ &= \mathbb{P}(y_0)\prod_{i\neq 0}\frac{\mathbb{P}(y_i, y_{\pi_i})}{\mathbb{P}(y_{\pi_i})}\\ &= \frac{\mathbb{P}(y_0)\prod_s \mathbb{P}(\boldsymbol{y}_s)}{\prod_j \mathbb{P}(y_j)^{|i:j=\pi_i|}}\\ &= \frac{\mathbb{P}(y_0)\prod_s \mathbb{P}(\boldsymbol{y}_s)}{\mathbb{P}(y_0)^{|N(0)|}\prod_{j\neq 0}\mathbb{P}(y_j)^{|N(j)|-1}}\\ &= \frac{\prod_s \mathbb{P}(\boldsymbol{y}_s)}{\prod_j \mathbb{P}(y_j)^{|N(j)|-1}}\\ &= \prod_i p_i(y_i)^{1-|N(i)|} \times \prod_s q_s(\boldsymbol{y}_s).\end{aligned}$$

Step #2: Entropy and Log-Partition Function

Entropy of a probability distribution: $H(\mathbb{P}) = -\sum_y \mathbb{P}(y)\log\mathbb{P}(y)$

Definition: the Fenchel dual of a convex function $f : \mathbb{R}^D \to \mathbb{R}\cup\{+\infty\}$ is the convex function $f^\star : \mathbb{R}^D \to \mathbb{R}\cup\{+\infty\}$ defined pointwise as
$f^\star(\boldsymbol{v}) := \sup_{\boldsymbol{u}}\left(\boldsymbol{v}^\top\boldsymbol{u} - f(\boldsymbol{u})\right)$

Step #2: Entropy and Log-Partition Function

Entropy of a probability distribution: $H(\mathbb{P}) = -\sum_y \mathbb{P}(y)\log\mathbb{P}(y)$

Definition: the Fenchel dual of a convex function $f : \mathbb{R}^D \to \mathbb{R}\cup\{+\infty\}$ is the convex function $f^\star : \mathbb{R}^D \to \mathbb{R}\cup\{+\infty\}$ defined pointwise as
$f^\star(\boldsymbol{v}) := \sup_{\boldsymbol{u}}\left(\boldsymbol{v}^\top\boldsymbol{u} - f(\boldsymbol{u})\right)$

Theorem (I): the **log-partition function** $\log Z(\boldsymbol{\theta})$ and the **negative entropy** $-H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})$ are Fenchel dual:

$$\log Z(\boldsymbol{\theta}) = \max_{(\boldsymbol{p},\boldsymbol{q})\in\mathrm{MARG}(G)} \underbrace{\sum_i \boldsymbol{\theta}_i^\top\boldsymbol{p}_i + \sum_s \boldsymbol{\theta}_s^\top\boldsymbol{q}_s + H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})}_{\text{(negative) variational free energy}},$$

This underlies the well-known duality between maximum likelihood in log-linear models and maximum entropy.

Step #3: Loopy BP as Variational Inference

Theorem (II): The maximizers $(\boldsymbol{p}^*, \boldsymbol{q}^*)$ are the **true marginals** of $\mathbb{P}_{\boldsymbol{\theta}}$:

$$\boxed{(\boldsymbol{p}^*, \boldsymbol{q}^*) = \arg\max_{(\boldsymbol{p},\boldsymbol{q})\in\mathrm{MARG}(G)} \sum_i \boldsymbol{\theta}_i^\top\boldsymbol{p}_i + \sum_s \boldsymbol{\theta}_s^\top\boldsymbol{q}_s + H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})}$$

Step #3: Loopy BP as Variational Inference

Theorem (II): The maximizers $(\boldsymbol{p}^*, \boldsymbol{q}^*)$ are the **true marginals** of $\mathbb{P}_{\boldsymbol{\theta}}$:

$$\boxed{(\boldsymbol{p}^*, \boldsymbol{q}^*) = \arg\max_{(\boldsymbol{p},\boldsymbol{q})\in\mathrm{MARG}(G)} \sum_i \boldsymbol{\theta}_i^\top\boldsymbol{p}_i + \sum_s \boldsymbol{\theta}_s^\top\boldsymbol{q}_s + H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})}$$

Drawback: in general, MARG(G) and $H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})$ are both intractable

Step #3: Loopy BP as Variational Inference

Theorem (II): The maximizers $(\boldsymbol{p}^*, \boldsymbol{q}^*)$ are the **true marginals** of $\mathbb{P}_{\boldsymbol{\theta}}$:

$$\boxed{(\boldsymbol{p}^*, \boldsymbol{q}^*) = \arg\max_{(\boldsymbol{p},\boldsymbol{q})\in \mathrm{MARG}(G)} \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{p}_i + \sum_s \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})}$$

Drawback: in general, MARG(G) and $H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})$ are both intractable

Yedidia et al. (2001) showed that loopy BP entails two approximations:

1. Replace MARG(G) by LOCAL(G)
2. Approximate $H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})$ by the Bethe entropy $H_{\text{Bethe}}(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})$

Both are exact when the graph does not have cycles

Bethe Entropy Approximation

Derived by "pretending" the graph has no cycles
We have seen

$$\mathbb{P}_{\psi}(y|x) \approx \prod_i p_i(y_i)^{1-|N(i)|} \times \prod_s q_s(\boldsymbol{y}_s)$$

From which we can derive

$$\begin{aligned} H(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}}) &\approx H_{\text{Bethe}}(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}}) \\ &= \sum_i (1-|N(i)|) H_i(\boldsymbol{p}_i) + \sum_s H_s(\boldsymbol{q}_s) \end{aligned}$$

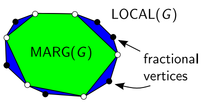
Hans Bethe, 1906–2005

A linear combination of local entropies:

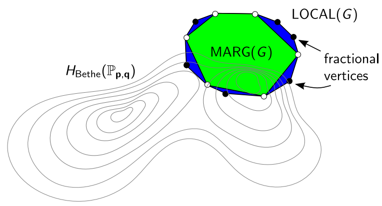
$$H_i(\boldsymbol{p}_i) = -\sum_{y_i} p_i(y_i)\log p_i(y_i), \quad H_s(\boldsymbol{q}_s) = -\sum_{\boldsymbol{y}_s} q_s(\boldsymbol{y}_s)\log q_s(\boldsymbol{y}_s)$$

Not concave in general!

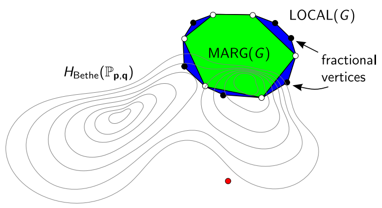
Geometric Illustration



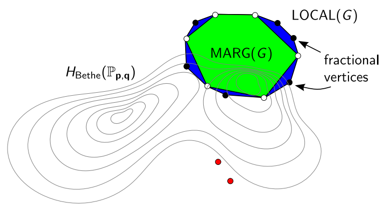
Geometric Illustration



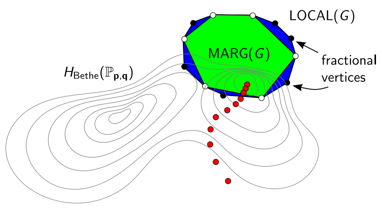
Geometric Illustration



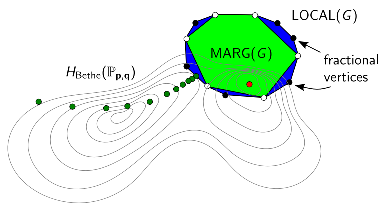
Geometric Illustration



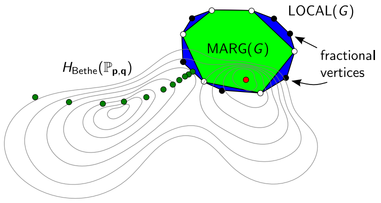
Geometric Illustration



Geometric Illustration



Geometric Illustration



If loopy BP converges, it reaches a stationary point of the approximate variational problem

$H_{\text{Bethe}}(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}})$ is non-concave in general $\Rightarrow$ local minima

Summary of Loopy BP

Advantages:

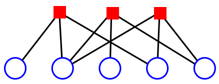
- Simple to implement
- Handles structured and logic factors (only need to compute local marginals)
- Often works well in practice (if cycles are not very influential)
- Often yields a reasonable approximation of $\log Z$ and H

Disadvantages:

- Doesn't give an upper/lower bound of $\log Z$ and H
- Entropy approximation is not concave (local minima)
- May not converge
- The final beliefs may not be realizable marginals

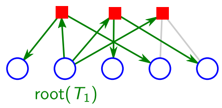
Tree Reweighted BP (Wainwright et al., 2005)

Key idea: cover the graph with a set of trees



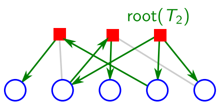
Tree Reweighted BP (Wainwright et al., 2005)

Key idea: cover the graph with a set of trees



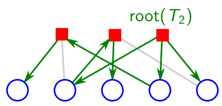
Tree Reweighted BP (Wainwright et al., 2005)

Key idea: cover the graph with a set of trees



Tree Reweighted BP (Wainwright et al., 2005)

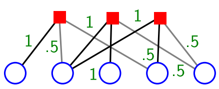
Key idea: cover the graph with a set of trees



Count the appearance probability $c_{is} > 0$ of each edge

Tree Reweighted BP (Wainwright et al., 2005)

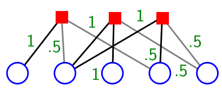
Key idea: cover the graph with a set of trees



Count the appearance probability $c_{is} > 0$ of each edge

Tree Reweighted BP (Wainwright et al., 2005)

Key idea: cover the graph with a set of trees



Count the appearance probability $c_{is} > 0$ of each edge

This results in a convex upper bound of $-H$ and $\log Z$:

$$H_{\text{TRBP}}(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}}) = \sum_i (1 - \textstyle\sum_{s \in N(i)} c_{is}) H_i(\boldsymbol{p}_i) + \sum_s H_s(\boldsymbol{q}_s)$$

(Note: if all $c_{is} = 1$ this would revert to the Bethe approximation)

TRBP Messages

- **Variable-to-factor messages:**

$$m_{i\to s}(y_i) = \frac{\psi_i(y_i)\prod_{s'\in N(i)} n_{s'\to i}^{c_{is'}}(y_i)}{n_{s\to i}(y_i)}$$

- **Factor-to-variable messages:**

$$n_{s\to i}(y_i) = \sum_{\boldsymbol{y}_s\sim y_i} \frac{\psi_s(\boldsymbol{y}_s)\prod_{j\in N(s)} m_{j\to s}^{c_{js}}(y_j)}{m_{i\to s}(y_i)}$$

- **Variable beliefs:**

$$p_i(y_i) \propto \psi_i(y_i) \prod_{s\in N(i)} n_{s\to i}^{c_{is}}(y_i)$$

- **Factor beliefs:**

$$q_s(\boldsymbol{y}_s) \propto \psi_s(\boldsymbol{y}_s) \prod_{i\in N(s)} m_{i\to s}^{c_{is}}(y_i)$$

Summary of TRBP

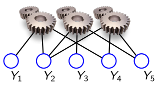
Advantages:

- Still simple to implement
- Entropy approximation is concave (no local minima)
- Gives an upper bound on $-H$ and $\log Z$
- Lots of knobs (the appearance probabilities)

Disadvantages:

- Lots of knobs (the appearance probabilities)
- Typically it's a very loose bound
- May not converge (but in practice always does, with dampening)
- The final beliefs may not be realizable marginals

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Norm-Product BP (Hazan and Shashua, 2010)

Subsumes loopy BP and TRBP

Relies on a convex approximation to the entropy using *counting numbers* $c_i \geq 0$ and $c_s > 0$ (in its simpler variant)

$$H_{\mathrm{NPBP}}(\mathbb{P}_{\boldsymbol{p},\boldsymbol{q}}) = \sum_i c_i H_i(\boldsymbol{p}_i) + \sum_s c_s H_s(\boldsymbol{q}_s)$$

Messages will become *norms*

Quick Reminder: ℓ_p-norms

$$\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p\right)^{1/p}$$

- $p=1$: sum of absolute values
- $p=2$: Euclidean norm
- $p=\infty$: $\|\boldsymbol{x}\|_\infty = max_i|x_i|$

NPBP Messages

- **Variable-to-factor messages:**

$$m_{i\to s}(y_i) = \frac{\left(\psi_i(y_i)\prod_{s'\in N(i)} n_{s'\to i}(y_i)\right)^{c_s/(c_i+\sum_{s'\in N(i)} c'_s)}}{n_{s\to i}(y_i)}$$

- **Factor-to-variable messages:**

$$n_{s\to i}(y_i) = \left(\sum_{\boldsymbol{y}_s\sim y_i}\left(\psi_s(\boldsymbol{y}_s)\prod_{j\in N(s)\setminus\{i\}} m_{j\to s}(y_j)\right)^{1/c_s}\right)^{c_s}$$

- **Variable beliefs:**

$$p_i(y_i) \propto \left(\psi_i(y_i)\prod_{s\in N(i)} n_{s\to i}(y_i)\right)^{1/(c_i+\sum_{s'\in N(i)} c'_s)}$$

- **Factor beliefs:**

$$q_s(\boldsymbol{y}_s) \propto \left(\psi_s(\boldsymbol{y}_s)\prod_{i\in N(s)} m_{i\to s}(y_i)\right)^{c_s}$$

Summary of NPBP

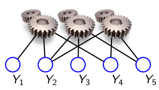
Advantages:

- Still simple to implement
- Entropy approximation is concave (no local minima)
- Always converges (primal-dual block ascent)
- Lots of knobs (the counting numbers)

Disadvantages:

- Lots of knobs (the counting numbers)
- Messages are not computed in parallel (otherwise, may not converge)
- The final beliefs may not be realizable marginals

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Outline

Zero-Limit Temperature

Define Z_ϵ where ϵ is a *temperature parameter*:

$$Z_\epsilon(\psi, x) = \left(\sum_{y \in \mathcal{Y}(x)} \prod_i \psi_i(y_i)^{1/\epsilon} \prod_s \psi_s(y_s)^{1/\epsilon} \right)^\epsilon$$

If $\epsilon = 1$, this becomes the partition function $Z(\psi, x)$

If $\epsilon \to 0$, this becomes the mode of $\mathbb{P}_\psi(y|x)$

Note that $Z_\epsilon(\psi, x) = Z(\psi^{1/\epsilon}, x)^\epsilon$ for any ϵ, i.e., **Z_ϵ can be computed by the same means as the partition function by scaling the potentials**

By choosing a small enough ϵ, any sum-product message-passing algorithm can be used to approximate the MAP (**smoothening**)

There is a trade-off between precision and numerical stability

Max-Product Belief Propagation

- For MAP decoding instead of marginal decoding
- Only change: factor-to-variable messages (max instead of $\sum$)

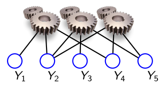
$$n_{s \to i}(y_i) = \max_{\mathbf{y}_s \sim y_i} \left(\psi_s(\mathbf{y}_s) \prod_{j \in N(s) \setminus \{i\}} m_{j \to s}(y_j) \right) = \frac{\max_{\mathbf{y}_s \sim y_i} q_s(\mathbf{y}_s)}{m_{i \to s}(y_i)}$$

- If the graph has no cycles, beliefs will converge to **max-marginals**:

$$p_i(y_i) \to \max_{y \sim y_i} \mathbb{P}_\psi(y|x), \quad q_s(\mathbf{y}_s) \to \max_{y \sim \mathbf{y}_s} \mathbb{P}_\psi(y|x)$$

- Decoding the best max-marginal at each variable node gives the MAP

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

TRW-S (Kolmogorov, 2006)

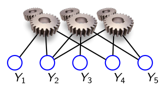
Same rationale as sum-product TRBP: cover the graph with spanning trees, and compute messages using edge appearance probabilities

Only differences:

- Replace $\sum$ with max
- Messages need to be computed *sequentially* for convergence

As max-product loopy BP, all is required is to compute *local max-marginals*

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Max-Product LP (Globerson and Jaakkola, 2008)

Derived by writing the dual of LP-MAP, and solving it with a **block coordinate descent algorithm**

The message updates need to be computed in a sequential schedule

Progress in the dual objective is monotonic

Drawback: since the dual is non-smooth, we may get stuck at a a suboptimal point

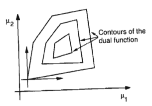


Figure 6.3.6. The basic difficulty with coordinate ascent for a nondifferentiable dual function. At some points it may be impossible to improve the dual function along any coordinate direction.

(From Bertsekas et al. (1999))

MPLP Messages

- **Variable-to-factor messages:**

$$m_{i \to s}(y_i) = \psi_i(y_i) \prod_{s' \in N(i) \setminus \{s\}} n_{s' \to i}(y_i)$$

- **Factor-to-variable messages:**

$$n_{s \to i}(y_i) = \frac{\max_{\mathbf{y}_s \sim y_i} \left(\psi_s(\mathbf{y}_s)^{1/|N(s)|} \prod_{j \in N(s)} m_{j \to s}(y_j)^{1/|N(s)|} \right)}{m_{i \to s}(y_i)}$$

Summary of MPLP

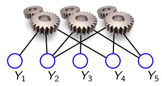
Advantages:

- Very simple to implement
- Handles structured and logic factors (only need to compute local max-marginals)
- Monotonically improves the dual
- No parameters to tune

Disadvantages:

- Can get stuck at a suboptimal solution (general problem with nonsmooth coordinate ascent)
- Messages are not computed in parallel (otherwise, may not converge)

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

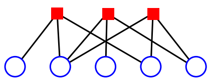
Summing Up Message-Passing

- BP algorithms and their variants can be used both for MAP and marginal decoding
- They need to compute local *marginals* (sum-product) or *max-marginals* (max-product)
- Always exact if the graph has no cycles; approximate otherwise
- They entail minimizing an energy approximation over the local polytope
- Some variants do convex approximations or compute upper bounds
- Two views of MAP decoding: (1) the near-zero temperature limit of marginal decoding; (2) a non-smooth optimization problem

Outline

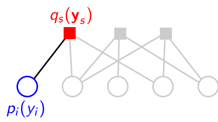
Dual Decomposition

- Old idea in optimization (Dantzig and Wolfe, 1960; Everett III, 1963)
- First proposed by Komodakis et al. (2007) in computer vision
- Introduced in NLP by Rush et al. (2010) for model combination
- Successful in syntax, semantics, MT (Koo and Collins, 2010; Auli and Lopez, 2011; Rush and Collins, 2011; Chang and Collins, 2011; Martins et al., 2011b; Rush and Collins, 2012; Almeida and Martins, 2013; Almeida et al., 2014; Martins and Almeida, 2014)



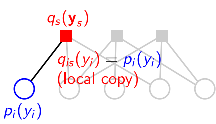
Dual Decomposition

- Old idea in optimization (Dantzig and Wolfe, 1960; Everett III, 1963)
- First proposed by Komodakis et al. (2007) in computer vision
- Introduced in NLP by Rush et al. (2010) for model combination
- Successful in syntax, semantics, MT (Koo and Collins, 2010; Auli and Lopez, 2011; Rush and Collins, 2011; Chang and Collins, 2011; Martins et al., 2011b; Rush and Collins, 2012; Almeida and Martins, 2013; Almeida et al., 2014; Martins and Almeida, 2014)



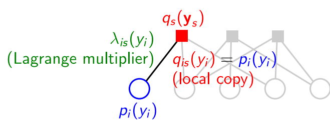
Dual Decomposition

- Old idea in optimization (Dantzig and Wolfe, 1960; Everett III, 1963)
- First proposed by Komodakis et al. (2007) in computer vision
- Introduced in NLP by Rush et al. (2010) for model combination
- Successful in syntax, semantics, MT (Koo and Collins, 2010; Auli and Lopez, 2011; Rush and Collins, 2011; Chang and Collins, 2011; Martins et al., 2011b; Rush and Collins, 2012; Almeida and Martins, 2013; Almeida et al., 2014; Martins and Almeida, 2014)



Dual Decomposition

- Old idea in optimization (Dantzig and Wolfe, 1960; Everett III, 1963)
- First proposed by Komodakis et al. (2007) in computer vision
- Introduced in NLP by Rush et al. (2010) for model combination
- Successful in syntax, semantics, MT (Koo and Collins, 2010; Auli and Lopez, 2011; Rush and Collins, 2011; Chang and Collins, 2011; Martins et al., 2011b; Rush and Collins, 2012; Almeida and Martins, 2013; Almeida et al., 2014; Martins and Almeida, 2014)



Recap: LP-MAP

Recall the LP-MAP problem:

$$\begin{array}{ll} \text{maximize} & \sum_i \boldsymbol{\theta}_i^\top \boldsymbol{p}_i + \sum_s \boldsymbol{\theta}_s^\top \boldsymbol{q}_s \\ \text{subject to} & \left\{ \begin{array}{l} \boldsymbol{q}_s \in \Delta^{|\mathcal{Y}_s|}, \ \forall s \\ \boldsymbol{p}_i = \mathbf{M}_{is} \boldsymbol{q}_s, \ \forall i, s. \end{array} \right. \quad \textbf{(local polytope)} \end{array}$$

Matrix $\mathbf{M}_{is} \in \{0,1\}^{|\mathcal{Y}_i| \times |\mathcal{Y}_s|}$ represents the constraints $p_i(y_i) = \sum_{\boldsymbol{y}_s \sim y_i} q_s(\boldsymbol{y}_s)$

We'll reformulate this problem by:

1. Introducing copy variables $\boldsymbol{q}_{is} = \boldsymbol{p}_i$
2. Defining $\boldsymbol{\theta}_{is} := \boldsymbol{\theta}_i / |N(i)|$

Reformulation of LP-MAP

The problem becomes:

$$\begin{array}{ll} \text{maximize} & \sum_s \left(\boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} \boldsymbol{\theta}_{is}^\top \boldsymbol{q}_{is} \right) \\ \text{subject to} & \left\{ \begin{array}{l} \boldsymbol{q}_s \in \Delta^{|\mathcal{Y}_s|}, \ \forall s \\ \boldsymbol{q}_{is} = \mathbf{M}_{is} \boldsymbol{q}_s, \ \forall i, s \\ \boldsymbol{q}_{is} = \boldsymbol{p}_i, \ \forall i, s. \end{array} \right. \quad \textbf{(local polytope)} \end{array}$$

By introducing Lagrange multipliers for the last constraints, we get the following **Lagrangian function**:

$$\mathcal{L}(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{\lambda}) = \sum_s \left(\boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} \boldsymbol{\theta}_{is}^\top \boldsymbol{q}_{is} \right) + \sum_{is} \boldsymbol{\lambda}_{is}^\top (\boldsymbol{p}_i - \boldsymbol{q}_{is})$$

Dual of LP-MAP

The dual problem is

$$\text{minimize} \ \sum_s g_s(\boldsymbol{\lambda}) \qquad \text{subject to } \boldsymbol{\lambda} \in \Lambda := \left\{ \boldsymbol{\lambda} \ \middle| \ \sum_{s \in N(i)} \boldsymbol{\lambda}_{is} = \mathbf{0} \right\}$$

where the $g_s(\boldsymbol{\lambda})$ are **local subproblems**,

$$\begin{aligned} g_s(\boldsymbol{\lambda}) &:= \max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \left(\boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} \right) \\ &= \max_{\boldsymbol{y}_s \in \mathcal{Y}_s} \left(\theta_s(\boldsymbol{y}_s) + \sum_{i \in N(s)} (\theta_{is}(y_i) + \lambda_{is}(y_i)) \right) \end{aligned}$$

and $\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s$ encodes the constraints $\left\{ \begin{array}{l} \boldsymbol{q}_s \in \Delta^{|\mathcal{Y}_s|} \\ \boldsymbol{q}_{is} = \mathbf{M}_{is} \boldsymbol{q}_s, \ \forall i \in N(s). \end{array} \right.$

A subgradient can be computed by solving these local subproblems

Projected Subgradient (Komodakis et al., 2007)

initialize penalties to zero
repeat

until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

Projected Subgradient (Komodakis et al., 2007)

initialize penalties to zero
repeat
  for each factor s **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is}$
  end for

until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

Projected Subgradient (Komodakis et al., 2007)

initialize penalties to zero
repeat
  for each factor s **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is}$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s \in N(i)} \boldsymbol{q}_{is}$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

Projected Subgradient (Komodakis et al., 2007)

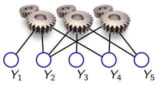
initialize penalties to zero
repeat
  for each factor s **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is}$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s \in N(i)} \boldsymbol{q}_{is}$
  $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

Projected Subgradient (Komodakis et al., 2007)

initialize penalties to zero
repeat
  for each factor s **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i \in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is}$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s \in N(i)} \boldsymbol{q}_{is}$
  $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

- Guaranteed to converge to an ϵ-accurate solution after at most $O(1/\epsilon^2)$ iterations
- **Problem:** too slow when there are many factors (Martins et al., 2011b)

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Accelerating Consensus

Two fundamental problems with the subgradient algorithm:

1. The dual objective $\sum_s g_s(\boldsymbol{\lambda})$ is non-smooth
2. Consensus is promoted only through updating $\boldsymbol{\lambda}$ (no memory about past updates)

Accelerating Consensus

Two fundamental problems with the subgradient algorithm:

1. The dual objective $\sum_s g_s(\boldsymbol{\lambda})$ is non-smooth
2. Consensus is promoted only through updating $\boldsymbol{\lambda}$ (no memory about past updates)

How can dual decomposition be accelerated?

Accelerating Consensus

Two fundamental problems with the subgradient algorithm:

1. The dual objective $\sum_s g_s(\boldsymbol{\lambda})$ is non-smooth
2. Consensus is promoted only through updating $\boldsymbol{\lambda}$ (no memory about past updates)

How can dual decomposition be accelerated?

- Jojic et al. (2010) smooth the objective and use gradient methods
- Martins et al. (2011a): augmented Lagrangian

Accelerating Consensus

Two fundamental problems with the subgradient algorithm:

1. The dual objective $\sum_s g_s(\boldsymbol{\lambda})$ is non-smooth
2. Consensus is promoted only through updating $\boldsymbol{\lambda}$ (no memory about past updates)

How can dual decomposition be accelerated?

- Jojic et al. (2010) smooth the objective and use gradient methods
- Martins et al. (2011a): augmented Lagrangian

Accelerated Gradient (Jojic et al., 2010)

Basic idea: make the dual objective smooth by adding an entropic perturbation with a near-zero ϵ temperature (also Johnson (2008))

The subproblems become local *marginal* computations instead of maximizations

With Nesterov's accelerated gradient method (Nesterov, 2005), the iteration bound goes from $O(1/\epsilon^2)$ to $O(1/\epsilon)$

Accelerated Gradient (Jojic et al., 2010)

Basic idea: make the dual objective smooth by adding an entropic perturbation with a near-zero ϵ temperature (also Johnson (2008))

The subproblems become local *marginal* computations instead of maximizations

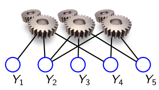
With Nesterov's accelerated gradient method (Nesterov, 2005), the iteration bound goes from $O(1/\epsilon^2)$ to $O(1/\epsilon)$

However: very sensitive to the temperature parameter

With low temperatures, may face numerical issues (in particular for some hard-constraint factors)

In practice, quite slow to take off (we'll see some plots later)

What Kind of Local Decoding Do We Need?



Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD^3 (Martins et al., 2011a)	QP/MAP

Accelerating Consensus

Two fundamental problems with the subgradient algorithm:

1. The dual objective $\sum_s g_s(\boldsymbol{\lambda})$ is non-smooth
2. Consensus is promoted only through updating $\boldsymbol{\lambda}$ (no memory about past updates)

How can dual decomposition be accelerated?

- Jojic et al. (2010) smooth the objective and use gradient methods ✓
- Martins et al. (2011a): augmented Lagrangian

Accelerating Consensus

Two fundamental problems with the subgradient algorithm:

1. The dual objective $\sum_s g_s(\boldsymbol{\lambda})$ is non-smooth
2. Consensus is promoted only through updating $\boldsymbol{\lambda}$ (no memory about past updates)

How can dual decomposition be accelerated?

- Jojic et al. (2010) smooth the objective and use gradient methods ✓
- Martins et al. (2011a): augmented Lagrangian

Alternating Directions Dual Decomposition (AD^3)

Based on the **alternating direction method of multipliers (ADMM):**

- an old method in optimization inspired by augmented Lagrangians (Gabay and Mercier, 1976; Glowinski and Marroco, 1975)
- a natural fit to consensus problems
- a natural "upgrade" of the subgradient algorithm (Boyd et al., 2011)

Augmented Lagrangian and ADMM

Basic idea: augment the Lagrangian function with a **quadratic penalty**

$$\mathcal{L}_\eta(\boldsymbol{p},\boldsymbol{q},\boldsymbol{\lambda}) = \sum_s \left(\boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \textstyle\sum_{i\in N(s)} \boldsymbol{\theta}_{is}^\top \boldsymbol{q}_{is}\right) + \sum_{is} \boldsymbol{\lambda}_{is}^\top (\boldsymbol{p}_i - \boldsymbol{q}_{is}) - \sum_{is} \|\boldsymbol{q}_{is} - \boldsymbol{p}_i\|^2$$

Method of multipliers (super-linear convergence):

1. Maximize $\mathcal{L}_\eta(\boldsymbol{p},\boldsymbol{q},\boldsymbol{\lambda})$ jointly w.r.t. $\boldsymbol{p}$ and $\boldsymbol{q}$ (challenging)
2. Multiplier update: $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$

Alternating direction method of multipliers: replace step 1 by separate maximizations (first w.r.t. $\boldsymbol{q}$, then $\boldsymbol{p}$)

From Subgradient to AD^3 (Martins et al., 2011a)

initialize penalties to zero
repeat
  for **each** factor $s = 1, \ldots, S$ **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \Omega_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is}$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s\in N(i)} \boldsymbol{q}_{is}$
  $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

From Subgradient to AD^3 (Martins et al., 2011a)

initialize penalties to zero
repeat
  for **each** factor $s = 1, \ldots, S$ **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \Omega_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2} \sum_{i\in N(s)} \|\boldsymbol{q}_{is} - \boldsymbol{p}_i\|^2$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s\in N(i)} \boldsymbol{q}_{is}$
  $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

From Subgradient to AD^3 (Martins et al., 2011a)

initialize penalties to zero
repeat
  for **each** factor $s = 1, \ldots, S$ **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \Omega_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2} \sum_{i\in N(s)} \|\boldsymbol{q}_{is} - \boldsymbol{p}_i\|^2$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s\in N(i)} \boldsymbol{q}_{is}$
  $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

- **faster consensus:** regularize $\boldsymbol{q}$-step towards average votes in $\boldsymbol{p}$

From Subgradient to AD^3 (Martins et al., 2011a)

initialize penalties to zero
repeat
  for **each** factor $s = 1, \ldots, S$ **do**
    $\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \Omega_s} \boldsymbol{\theta}_s^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is} + \boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2} \sum_{i\in N(s)} \|\boldsymbol{q}_{is} - \boldsymbol{p}_i\|^2$
  end for
  $\boldsymbol{p}_i \leftarrow \frac{1}{|N(i)|} \sum_{s\in N(i)} \boldsymbol{q}_{is}$
  $\boldsymbol{\lambda}_{is} \leftarrow \boldsymbol{\lambda}_{is} - \eta(\boldsymbol{q}_{is} - \boldsymbol{p}_i)$
until consensus (all $\boldsymbol{q}_{is} = \boldsymbol{p}_i$) or maximum number of iterations reached

- **faster consensus:** regularize $\boldsymbol{q}$-step towards average votes in $\boldsymbol{p}$
- **better stopping conditions:** keeps track of primal and dual residuals

Theoretical Guarantees of AD^3

Convergent in primal and dual (Glowinski and Le Tallec, 1989)

Iteration bound: $O(1/\epsilon)$ (cf. $O(1/\epsilon^2)$ for projected subgradient)

Inexact AD^3 subproblems: still convergent if residuals are summable (Eckstein and Bertsekas, 1992)

Always dual feasible: can compute upper bounds and embed in branch-and-bound toward *exact* decoding (Das et al., 2012)

Theoretical Guarantees of AD^3

Convergent in primal and dual (Glowinski and Le Tallec, 1989)

Iteration bound: $O(1/\epsilon)$ (cf. $O(1/\epsilon^2)$ for projected subgradient)

Inexact AD^3 subproblems: still convergent if residuals are summable (Eckstein and Bertsekas, 1992)

Always dual feasible: can compute upper bounds and embed in branch-and-bound toward *exact* decoding (Das et al., 2012)

But: AD^3 local subproblems are *quadratic* (more involved than in projected subgradient)

Theoretical Guarantees of AD^3

Convergent in primal and dual (Glowinski and Le Tallec, 1989)

Iteration bound: $O(1/\epsilon)$ (cf. $O(1/\epsilon^2)$ for projected subgradient)

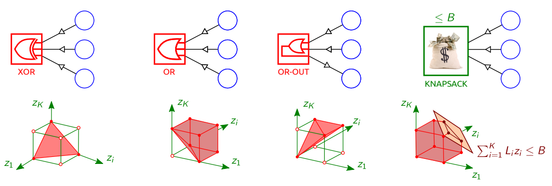
Inexact AD^3 subproblems: still convergent if residuals are summable (Eckstein and Bertsekas, 1992)

Always dual feasible: can compute upper bounds and embed in branch-and-bound toward *exact* decoding (Das et al., 2012)

But: AD^3 local subproblems are *quadratic* (more involved than in projected subgradient)

Still—very easy and efficient for logic and knapsack factors!

Projecting onto Hard Constraint Polytopes



- Martins et al. (2014): logic factors can be solved in $O(K)$ time
- **Almeida and Martins (2013): same for *knapsack* factors!**

Structured Factors

What about structured factors?

Structured Factors

What about structured factors?

Projected subgradient handles these quite well

- combinatorial machinery (Viterbi, Chu-Liu-Edmonds, Fulkerson-Ford, Floyd-Warshall,...)

We cannot solve the AD^3 subproblems with that machinery...

Structured Factors

What about structured factors?

Projected subgradient handles these quite well

- combinatorial machinery (Viterbi, Chu-Liu-Edmonds, Fulkerson-Ford, Floyd-Warshall,...)

We cannot solve the AD^3 subproblems with that machinery...

Or can we?

Structured Factors

What about structured factors?

Projected subgradient handles these quite well

- combinatorial machinery (Viterbi, Chu-Liu-Edmonds, Fulkerson-Ford, Floyd-Warshall,...)

We cannot solve the AD^3 subproblems with that machinery...

Or can we?

Active set method: seek the support of the solution by adding/removing components; very suitable for warm-starting (Nocedal and Wright, 1999)

An Active Set Method for the AD³ Subproblem

$$\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \left(\boldsymbol{\theta}_s{}^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is}+\boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2}\sum_{i\in N(s)} \|\boldsymbol{q}_{is}-\boldsymbol{p}_i\|^2 \right)$$

An Active Set Method for the AD³ Subproblem

$$\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \left(\boldsymbol{\theta}_s{}^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is}+\boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2}\sum_{i\in N(s)} \|\boldsymbol{q}_{is}-\boldsymbol{p}_i\|^2 \right)$$

Too many possible assignments: $O(\exp(|N(s)|))$

An Active Set Method for the AD³ Subproblem

$$\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \left(\boldsymbol{\theta}_s{}^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is}+\boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2}\sum_{i\in N(s)} \|\boldsymbol{q}_{is}-\boldsymbol{p}_i\|^2 \right)$$

Too many possible assignments: $O(\exp(|N(s)|))$

Key result: solution spanned by only $O(|N(s)|)$ **assignments**

An Active Set Method for the AD³ Subproblem

$$\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \left(\boldsymbol{\theta}_s{}^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is}+\boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2}\sum_{i\in N(s)} \|\boldsymbol{q}_{is}-\boldsymbol{p}_i\|^2 \right)$$

Too many possible assignments: $O(\exp(|N(s)|))$

Key result: solution spanned by only $O(|N(s)|)$ **assignments**

Active set methods: seek the support of the solution by adding/removing components; very suitable for warm-starting (Nocedal and Wright, 1999)

Only requirement: a local-max oracle (as in projected subgradient)

An Active Set Method for the AD³ Subproblem

$$\bar{\boldsymbol{q}}_s \leftarrow \arg\max_{\bar{\boldsymbol{q}}_s \in \mathcal{Q}_s} \left(\boldsymbol{\theta}_s{}^\top \boldsymbol{q}_s + \sum_{i\in N(s)} (\boldsymbol{\theta}_{is}+\boldsymbol{\lambda}_{is})^\top \boldsymbol{q}_{is} - \frac{\eta}{2}\sum_{i\in N(s)} \|\boldsymbol{q}_{is}-\boldsymbol{p}_i\|^2 \right)$$

Too many possible assignments: $O(\exp(|N(s)|))$

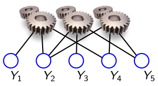
Key result: solution spanned by only $O(|N(s)|)$ **assignments**

Active set methods: seek the support of the solution by adding/removing components; very suitable for warm-starting (Nocedal and Wright, 1999)

Only requirement: a local-max oracle (as in projected subgradient)

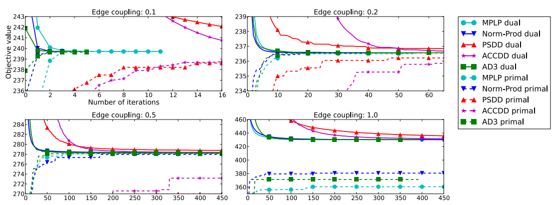
More info: Martins et al. (2014)

What Kind of Local Decoding Do We Need?



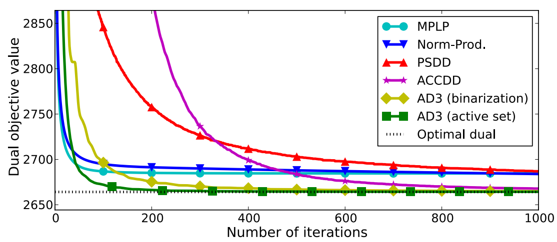
Algorithm	Local Operation
Sum-Prod. BP (Pearl, 1988)	marginals
TRBP (Wainwright et al., 2005)	marginals
Norm-Product BP (Hazan and Shashua, 2010)	marginals
Max-Prod. BP (Pearl, 1988)	max-marginals
TRW-S (Kolmogorov, 2006)	max-marginals
MPLP (Globerson and Jaakkola, 2008)	max-marginals
PSDD (Komodakis et al., 2007)	MAP
Accelerated DD (Jojic et al., 2010)	marginals
AD³ (Martins et al., 2011a)	QP/MAP

Example: Ising Grid (30 × 30)



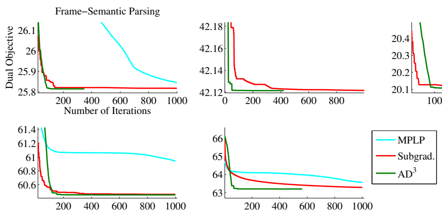
- A. Martins, M. Figueiredo, P. Aguiar, N. Smith, E. Xing (2014).
  AD³: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models.
  Journal of Machine Learning Research (to appear).

Example: Potts Grid (20 × 20, 8 states)



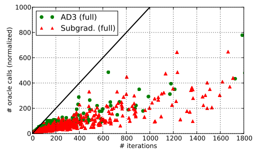
- A. Martins, M. Figueiredo, P. Aguiar, N. Smith, E. Xing (2014).
  AD³: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models.
  Journal of Machine Learning Research (to appear).

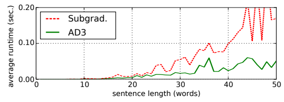
Example: Frame-Semantic Parsing



- Embedded in a branch-and-bound procedure for *exact* decoding
- D. Das, A. Martins, N. Smith.
  "An Exact DD Algorithm for Shallow Semantic Parsing with Constraints."
  *SEM Workshop, 2012.

Runtime of AD^3 vs Subgradient (Parsing)





- **Caching** and **warm-starting** the subproblems reduces drastically the number of oracle calls—**huge speed-ups!!**
- **AD^3 faster to achieve consensus** (due to the quadratic penalty)

Summing Up Dual Decomposition

- Dual decomposition is a general optimization technique that splits the dual into several subproblems (one per factor) that must agree on overlaps
- This can be used to solve LP-MAP
- We discussed three variants: subgradient (PSDD), accelerated gradient (ADD), and alternating directions (AD^3)
- The algorithms are convergent and retrieve the true MAP if the graph has no cycles; they also give certificates when the solution of LP-MAP equals the MAP
- For PSDD and AD^3 only local maximizations are necessary; ADD requires computing marginals

Outline

Applications

We'll discuss three applications:

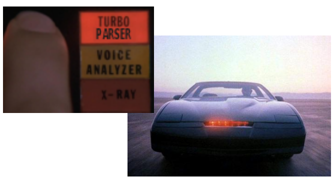
- Turbo Parsing
- Compressive Summarization
- Joint Coreference Resolution and Quotation Attribution

Applications

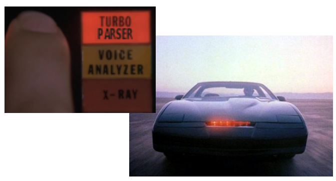
We'll discuss three applications:

- Turbo Parsing
- Compressive Summarization
- Joint Coreference Resolution and Quotation Attribution

What is a Turbo Parser?

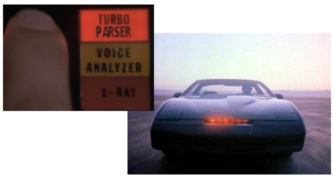


What is a Turbo Parser?



- **A parser that runs inference in factor graphs, ignoring global effects caused by loops (Martins et al., 2010)**
- name inspired from *turbo* decoders (Berrou et al., 1993)

What is a Turbo Parser?



- **A parser that runs inference in factor graphs, ignoring global effects caused by loops (Martins et al., 2010)**
- name inspired from *turbo* decoders (Berrou et al., 1993)
- **This talk:** we speed up turbo parsers via AD^3 w/ active set

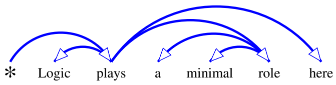
Recent Paper



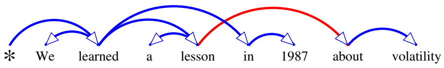
- André F. T. Martins, Miguel B. Almeida, Noah A. Smith. "Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers." ACL 2013 Short Paper.

An Important Distinction

- A projective tree:

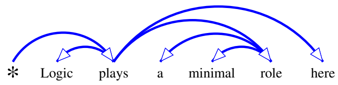


- A non-projective tree:

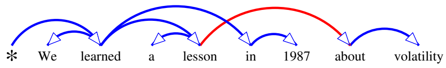


An Important Distinction

- A projective tree:



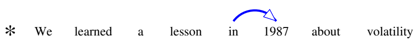
- A non-projective tree:



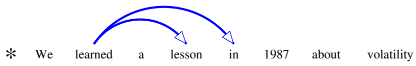
This talk: we allow non-projective trees.

Suitable for languages with flexible word order (Dutch, German, Czech,...)

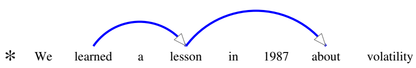
First-Order Scores for Arcs



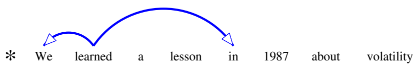
Second-Order Scores for Consecutive Siblings



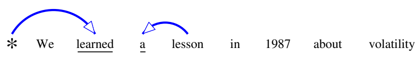
Second-Order Scores for Grandparents



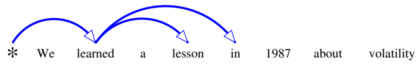
Scores for Arbitrary Siblings



Scores for Head Bigrams

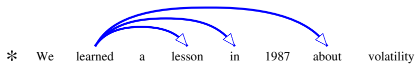


Third-Order Scores for Grand-siblings



Used by Koo and Collins (2010) for *projective* parsing.

Third-Order Scores for Tri-siblings



Used by Koo and Collins (2010) for *projective* parsing.

Decoding

How to deal with all these parts?

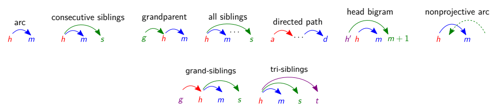
- Dynamic programming only available for the *projective* case...

Decoding

How to deal with all these parts?

- Dynamic programming only available for the *projective* case...
- Beyond arc-factored models, non-projective parsing is **NP-hard** (McDonald and Satta, 2007)
- **Need to embrace approximations!**

Approximate Dependency Parsers



	parser	AF	CS	G	AS	DP	HB	NPA	GS	TS
McDonald et al. (2006)	projective + greedy	✓	✓							
Smith et al. (2008)	loopy BP	✓	✓	✓	✓					
Martins et al. (2010)	LP solver	✓		✓	✓			✓		
Koo et al. (2010)	dual decomp.	✓	✓							
Martins et al. (2011)	AD^3	✓	✓	✓	✓	✓	✓	✓		
This work	AD^3 & active set	✓	✓	✓	✓		✓		✓	✓

Factor Graph Representation

- Variables nodes for **dependency arcs**, linked to a tree constraint
- Head automata for **consecutive siblings and grandparents** (as in Smith and Eisner (2008); Koo et al. (2010))
- Pairwise factors for **arbitrary siblings** (as Martins et al. (2011b))

Factor Graph Representation

- Variables nodes for **dependency arcs**, linked to a tree constraint
- Head automata for **consecutive siblings and grandparents** (as in Smith and Eisner (2008); Koo et al. (2010))
- Pairwise factors for **arbitrary siblings** (as Martins et al. (2011b))
- Third-order head automata for **grand-siblings and tri-siblings**
- Sequence model for **head bigrams**

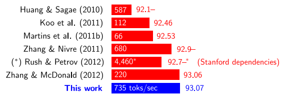
Factor Graph Representation

- Variables nodes for **dependency arcs**, linked to a tree constraint
- Head automata for **consecutive siblings and grandparents** (as in Smith and Eisner (2008); Koo et al. (2010))
- Pairwise factors for **arbitrary siblings** (as Martins et al. (2011b))
- Third-order head automata for **grand-siblings and tri-siblings**
- Sequence model for **head bigrams**

We solve an LP relaxation with AD^3.

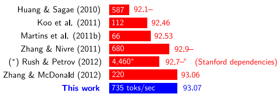
Parsing Accuracies/Runtimes

- Projective dataset (English PTB):

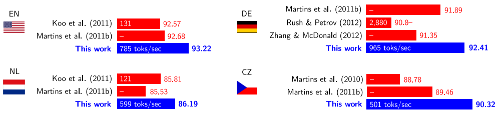


Parsing Accuracies/Runtimes

- Projective dataset (English PTB):



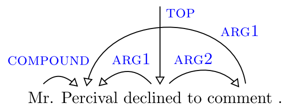
- Non-projective datasets (CoNLL-2006 and CoNLL-2008):



We get SOTA accuracies for the largest non-projective datasets!

Extensions

Same idea applied to **broad-coverage semantic parsing**



Best results in the SemEval 2014 shared task:

- André F. T. Martins and Mariana S. C. Almeida.
  "Priberam: A Turbo Semantic Parser with Second Order Features."
  SemEval 2014.

Applications

We'll discuss three applications:

- Turbo Parsing
- Compressive Summarization
- Joint Coreference Resolution and Quotation Attribution

Applications

We'll discuss three applications:

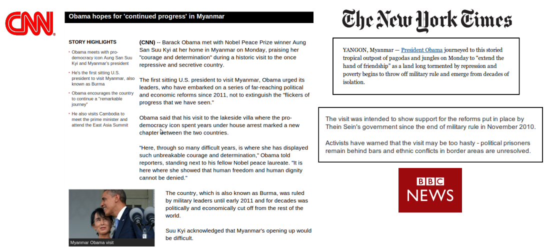
- Turbo Parsing
- Compressive Summarization
- Joint Coreference Resolution and Quotation Attribution

Recent Paper

- Miguel B. Almeida and André F. T. Martins.
  "Fast and Robust Compressive Summarization with Dual Decomposition and Multi-Task Learning."
  ACL 2013 Long Paper.

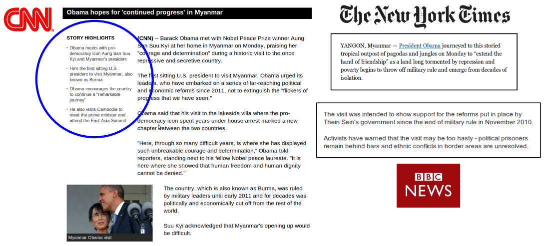
Multi-Document Summarization

Map a set of related **documents** to a brief **summary.**



Multi-Document Summarization

Map a set of related **documents** to a brief **summary.**



What Makes a Good Summary?

STORY HIGHLIGHTS

- Obama meets with pro-democracy icon Aung San Suu Kyi and Myanmar's president
- He's the first sitting U.S. president to visit Myanmar, also known as Burma
- Obama encourages the country to continue a "remarkable journey"
- He also visits Cambodia to meet the prime minister and attend the East Asia Summit

What Makes a Good Summary?

STORY HIGHLIGHTS

- Obama meets with pro-democracy icon Aung San Suu Kyi and Myanmar's president
- He's the first sitting U.S. president to visit Myanmar, also known as Burma
- Obama encourages the country to continue a "remarkable journey"
- He also visits Cambodia to meet the prime minister and attend the East Asia Summit

1. conciseness

What Makes a Good Summary?

STORY HIGHLIGHTS

- Obama meets with pro-democracy icon Aung San Suu Kyi and Myanmar's president
- He's the first sitting U.S. president to visit Myanmar, also known as Burma
- Obama encourages the country to continue a "remarkable journey"
- He also visits Cambodia to meet the prime minister and attend the East Asia Summit

1. conciseness
2. informativeness

What Makes a Good Summary?

STORY HIGHLIGHTS

- Obama meets with pro-democracy icon Aung San Suu Kyi and Myanmar's president
- He's the first sitting U.S. president to visit Myanmar, also known as Burma
- Obama encourages the country to continue a "remarkable journey"
- He also visits Cambodia to meet the prime minister and attend the East Asia Summit

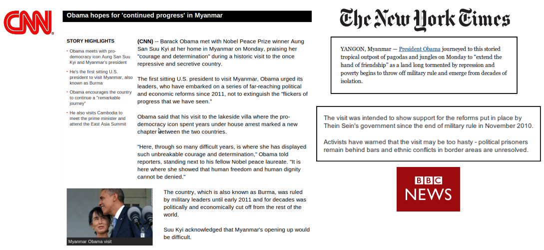
1. conciseness
2. informativeness
3. grammaticality

Extractive Summarization

Just ***extract*** the most salient sentences.

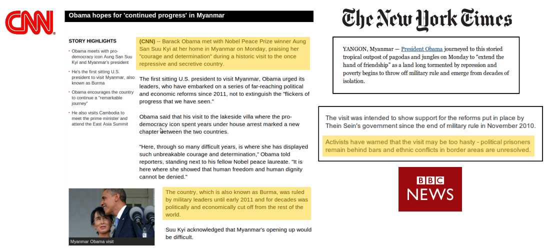
Extractive Summarization

Just ***extract*** the most salient sentences.



Extractive Summarization

Just ***extract*** the most salient sentences.

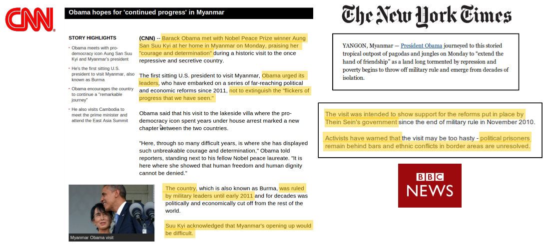


What We Do: Compressive Summarization

Jointly **extract** and **compress** sentences.

What We Do: Compressive Summarization

Jointly **extract** and **compress** sentences.



Compressive Summarization as Global Optimization

- Indicator variables for every word of the nth sentence, $\boldsymbol{z}_n := \langle z_{n,\ell} \rangle_{\ell=1}^{L_n}$

Compressive Summarization as Global Optimization

- Indicator variables for every word of the nth sentence, $\boldsymbol{z}_n := \langle z_{n,\ell} \rangle_{\ell=1}^{L_n}$
- Summary length must not exceed the **budget** (B words)
- Quality function rewards *global* **informativeness** (through $g(\boldsymbol{z})$)...
- ... but also *local* **grammaticality** (through $h_n(\boldsymbol{z}_n)$):

$$\begin{array}{ll} \text{maximize} & g(\boldsymbol{z}) + \sum_{n=1}^{N} h_n(\boldsymbol{z}_n) \\ \text{s.t.} & \sum_{n=1}^{N} \sum_{\ell=1}^{L_n} z_{n,\ell} \leq B. \end{array}$$

Grammaticality: Sentence Compression Model

Inspired by Knight and Marcu (2000)'s word deletion model

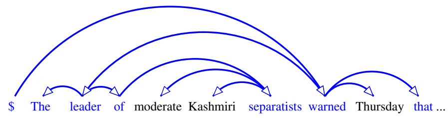
Other relevant work: McDonald (2006); Clarke and Lapata (2008); Martins and Smith (2009); Berg-Kirkpatrick et al. (2011)

Grammaticality: Sentence Compression Model

Inspired by Knight and Marcu (2000)'s word deletion model

Other relevant work: McDonald (2006); Clarke and Lapata (2008); Martins and Smith (2009); Berg-Kirkpatrick et al. (2011)

Our model factors over dependency arcs:

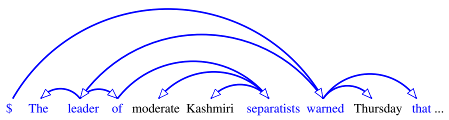


Grammaticality: Sentence Compression Model

Inspired by Knight and Marcu (2000)'s word deletion model

Other relevant work: McDonald (2006); Clarke and Lapata (2008); Martins and Smith (2009); Berg-Kirkpatrick et al. (2011)

Our model factors over dependency arcs:



A structured factor, locally decodable with dynamic programming.

Informativeness: Coverage Model

Inspired by *extractive* max-coverage models (Filatova and Hatzivassiloglou, 2004; Yih et al., 2007; Gillick et al., 2008; Lin and Bilmes, 2010)

Informativeness: Coverage Model

Inspired by *extractive* max-coverage models (Filatova and Hatzivassiloglou, 2004; Yih et al., 2007; Gillick et al., 2008; Lin and Bilmes, 2010)

- Extract a list of **concept types** $\langle c_m \rangle_{m=1}^M$ from the original documents
- Define concept **relevance scores** $\langle \sigma_m \rangle_{m=1}^M$

Informativeness: Coverage Model

Inspired by *extractive* max-coverage models (Filatova and Hatzivassiloglou, 2004; Yih et al., 2007; Gillick et al., 2008; Lin and Bilmes, 2010)

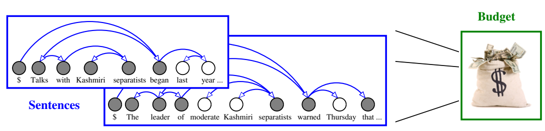
- Extract a list of **concept types** $\langle c_m \rangle_{m=1}^M$ from the original documents
- Define concept **relevance scores** $\langle \sigma_m \rangle_{m=1}^M$
- Define $g(z)$ as the global concept coverage:

$$g(z) := \sum_{m=1}^{M} \sigma_m u_m, \quad \text{where } u_m := \bigvee_{\langle n,\ell_s,\ell_e\rangle \in \mathcal{T}_m} \left(\bigwedge_{\ell=\ell_s}^{\ell_e} z_{n,\ell} \right)$$

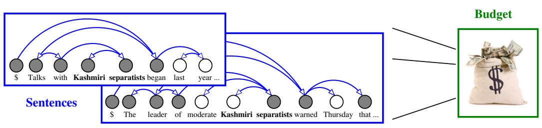
- Intuitively: a **concept type** occurs if some **concept token** occurs.

Graphical Model for Our Compressive Summarizer

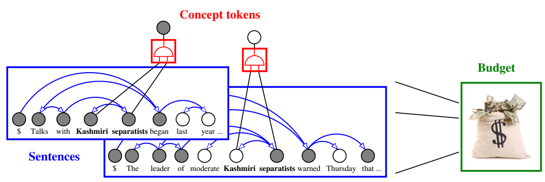
Graphical Model for Our Compressive Summarizer



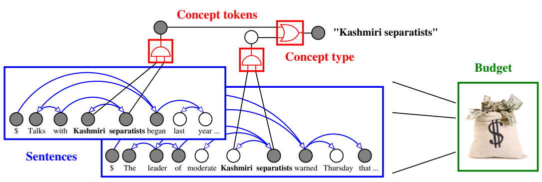
Graphical Model for Our Compressive Summarizer



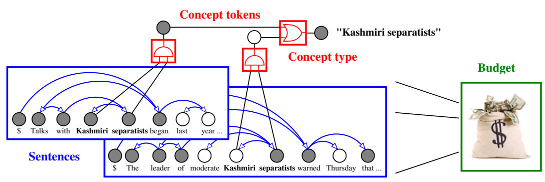
Graphical Model for Our Compressive Summarizer



Graphical Model for Our Compressive Summarizer

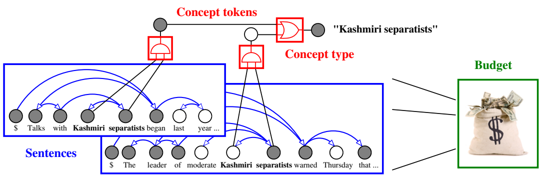


Graphical Model for Our Compressive Summarizer



1. We use dual decomposition (AD^3) for solving a linear relaxation
2. We apply a fast rounding procedure to obtain a valid summary

Graphical Model for Our Compressive Summarizer



1. We use dual decomposition (AD^3) for solving a linear relaxation
2. We apply a fast rounding procedure to obtain a valid summary

Multi-task learning: user-generated data (Simple English Wikipedia) along with manual abstracts and compressive summaries

Results on TAC-2008 Dataset

- Quality scores:

	ROUGE-2	Pyramid	LQ (1–5)
ICSI-1 (Gillick et al., 2008)	11.03	–	–
Berg-Kirkpatrick et al. (2011)	11.71	–	–
Woodsend and Lapata (2012)	11.37	–	–
Extractive, ILP	11.16	36.0	**4.6**
Single-task, AD^3	11.88	41.0	3.8
Multi-task, AD^3	**12.30**	**42.6**	4.2

Results on TAC-2008 Dataset

- Quality scores:

	ROUGE-2	Pyramid	LQ (1–5)
ICSI-1 (Gillick et al., 2008)	11.03	–	–
Berg-Kirkpatrick et al. (2011)	11.71	–	–
Woodsend and Lapata (2012)	11.37	–	–
Extractive, ILP	11.16	36.0	**4.6**
Single-task, AD^3	11.88	41.0	3.8
Multi-task, AD^3	**12.30**	**42.6**	4.2

- Averaged runtimes per summarization problem (10 documents):

Solver	Runtime (sec.)	ROUGE-2
ILP Exact, GLPK	10.394	12.40
LP-Relax., GLPK	2.265	12.38
AD^3 (1,000 its.)	**0.406**	**12.30**
Extractive (ILP)	0.265	11.16

Applications

We'll discuss three applications:

- Turbo Parsing
- Compressive Summarization
- Joint Coreference Resolution and Quotation Attribution

Applications

We'll discuss three applications:

- Turbo Parsing
- Compressive Summarization
- Joint Coreference Resolution and Quotation Attribution

Recent Paper



- Mariana S. C. Almeida, Miguel B. Almeida and André F. T. Martins. "A Joint Model for Quotation Attribution and Coreference Resolution." EACL 2014.

Why Jointly?

Coreference resolution and **quotation attribution** may benefit from being treated as a joint task.

Why Jointly?

Coreference resolution and **quotation attribution** may benefit from being treated as a joint task.

A speaker doesn't refer to himself as *he*:

> *Rivals carp at "the principle of Pilson," as NBC's Arthur Watson once put it – "he's always expounding that rights are too high, then he's going crazy." But the 49-year-old Mr. Pilson is hardly a man to ignore the numbers.*

Why Jointly?

Coreference resolution and **quotation attribution** may benefit from being treated as a joint task.

A speaker doesn't refer to himself as *he*:

> *Rivals carp at "the principle of Pilson," as NBC's Arthur Watson once put it – "he's always expounding that rights are too high, then he's going crazy." But the 49-year-old Mr. Pilson is hardly a man to ignore the numbers.*

Why Jointly?

Coreference resolution and **quotation attribution** may benefit from being treated as a joint task.

A speaker doesn't refer to himself as *he*:

> *Rivals carp at "the principle of Pilson," as NBC's Arthur Watson once put it – "he's always expounding that rights are too high, then he's going crazy." But the 49-year-old Mr. Pilson is hardly a man to ignore the numbers.*

Two consecutive quotes are often from co-referent speakers:

> *English novelist Dorothy L. Sayers described ringing as a "passion that finds its satisfaction in mathematical completeness and mechanical perfection."*
> *Ringers, she added, are "filled with the solemn intoxication that comes of intricate ritual faultlessly performed."*

Why Jointly?

Coreference resolution and **quotation attribution** may benefit from being treated as a joint task.

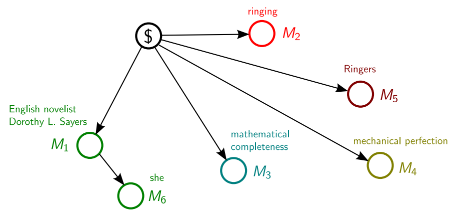
A speaker doesn't refer to himself as *he*:

> *Rivals carp at "the principle of Pilson," as NBC's Arthur Watson once put it – "he's always expounding that rights are too high, then he's going crazy." But the 49-year-old Mr. Pilson is hardly a man to ignore the numbers.*

Two consecutive quotes are often from co-referent speakers:

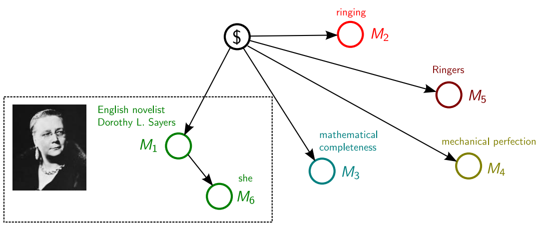
> *English novelist Dorothy L. Sayers described ringing as a "passion that finds its satisfaction in mathematical completeness and mechanical perfection."*
> *Ringers, she added, are "filled with the solemn intoxication that comes of intricate ritual faultlessly performed."*

Coreference Tree (Denis and Baldridge, 2008; Fernandes et al., 2012; Durrett and Klein, 2013)



- Clusters of co-referent mentions (**entities**) correspond to subtrees coming out from the root node.

Coreference Tree (Denis and Baldridge, 2008; Fernandes et al., 2012; Durrett and Klein, 2013)

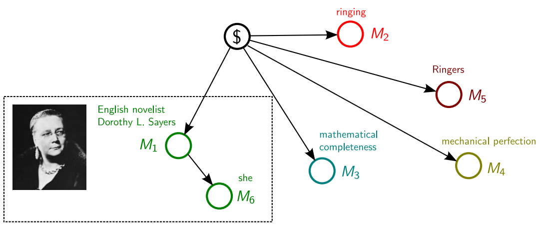


- Clusters of co-referent mentions (**entities**) correspond to subtrees coming out from the root node.

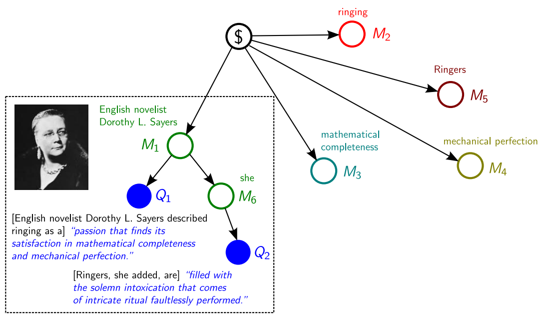
From Coreference to *Quotation-Coreference* Trees (Almeida et al., 2014)

- Include mention nodes **and quotation nodes**
- Quotation nodes have to be leaves
- Subtrees coming out from the root induce entity clusters along with their quotes: **entity-based quotation attribution**
- All arcs are left-to-right in the document (quotations are moved to the rightmost positions)

From Coreference to *Quotation-Coreference* Trees (Almeida et al., 2014)



From Coreference to *Quotation-Coreference* Trees (Almeida et al., 2014)



Beyond Arc Scores

The simplest coreference models (e.g., the SURFACE model of Durrett and Klein (2013)) are **arc-factored**

- Exact decoding can be performed in a greedy manner

Beyond Arc Scores

The simplest coreference models (e.g., the SURFACE model of Durrett and Klein (2013)) are **arc-factored**

- Exact decoding can be performed in a greedy manner

However: in our approach, an arc factored model would be equivalent to do coreference resolution and quotation attribution *independently*...

Beyond Arc Scores

The simplest coreference models (e.g., the SURFACE model of Durrett and Klein (2013)) are **arc-factored**

- Exact decoding can be performed in a greedy manner

However: in our approach, an arc factored model would be equivalent to do coreference resolution and quotation attribution *independently*...

To do things *jointly*, we add extra scores for:

- **A speaker being mentioned inside a quotation**
- **Consecutive quotes having the same speakers**

Beyond Arc Scores

The simplest coreference models (e.g., the SURFACE model of Durrett and Klein (2013)) are **arc-factored**

- Exact decoding can be performed in a greedy manner

However: in our approach, an arc factored model would be equivalent to do coreference resolution and quotation attribution *independently*...

To do things *jointly*, we add extra scores for:

- **A speaker being mentioned inside a quotation**
- **Consecutive quotes having the same speakers**

These scores require knowing if pairs of nodes are in the same subtree.

Logic Program

- Each node except the root has exactly one parent:

$$\sum_{i=0}^{j-1} a_{i\to j} = 1, \quad \forall j \neq 0$$

- Paths propagate through arcs:

$$\pi_{i\to^* i} = 1, \quad \forall i, \quad \pi_{i\to^* k} = \bigvee_{i<j\leq k} (a_{i\to j} \wedge \pi_{j\to^* k}), \quad \forall i,k$$

- Nodes k and ℓ are in the same subtree if they have a common ancestor which is not the root:

$$p_{k,\ell} = \bigvee_{i\neq 0} (\pi_{i\to^* k} \wedge \pi_{i\to^* \ell}), \quad \forall k, l.$$

Experiments

Datasets:

- WSJ portion of the Ontonotes (597 documents); same splits as CoNLL 2011 shared task
- Quotation annotations of the PARC dataset (Pareti, 2012; O'Keefe et al., 2012)

Coreference evaluation metrics: average between MUC, B^3, $CEAF_e$

Quotation evaluation metrics (new in this paper):

- **Representative speaker match (RSM):** # matches to representative (non-pronominal) mention of the gold speaker's entity
- **Entity cluster F_1 (ECF$_1$):** F_1 score between the predicted and gold speaker entity (sets of mentions)

Results

Coreference Resolution:

	MUC F_1	BCUB F_1	CEAFE F_1	Avg.
Durrett and Klein (2013) (SURFACE)	**58.87**	62.74	45.46	55.7
QUOTE/COREF INDEPENDENT	57.89	62.50	45.48	55.3
JOINT SYSTEM	58.78	**63.79**	**45.50**	**56.0**

Quotation attribution:

	RSM	ECF$_1$
QUOTEONLY	49.4%	41.2%
QUOTEAFTERCOREF	64.6%	70.0%
QUOTE/COREF INDEPENDENT	74.7%	73.7%
JOINT SYSTEM	**76.6%**	**74.1%**

Outline

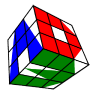
Conclusions

- Many structured problems in NLP are NP-hard or expensive (constrained models, diversity, combination of structured models)
- Often they can be approximately decoded via Linear Programming (e.g., by relaxing an ILP)
- The structure inherent to these problems can be represented with a factor graph
- Message-passing and dual decomposition algorithms can solve these LPs efficiently, exploiting the structure of the graph
- Conceptually: approximate *global* decoding by invoking only *local* decoders (local maximizations, marginals, max-marginals, QPs, ...)
- AD^3 is faster than the subgradient algorithm both in theory and in practice, and requires the same local decoders
- SOTA results in several applications (turbo parsing, summarization, joint coref and quotation attribution)

Thank you!

The parser and AD^3 are freely available at:

http://www.ark.cs.cmu.edu/TurboParser
http://www.ark.cs.cmu.edu/AD3



Acknowledgments

- Fundação para a Ciência e Tecnologia, grants PEst-OE/EEI/LA0008/2011 and PTDC/EEI-SII/2312/2012.
- Fundação para a Ciência e Tecnologia and Information and Communication Technologies Institute (Portugal/USA), through the CMU-Portugal Program.
- Priberam: QREN/POR Lisboa (Portugal), EU/FEDER programme, Discooperio project, contract 2011/18501.
- Priberam: QREN/POR Lisboa (Portugal), EU/FEDER programme, Intelligo project, contract 2012/24803.

References I

Almeida, M. B. and Martins, A. F. T. (2013). Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Almeida, M. S. C., Almeida, M. B., and Martins, A. F. T. (2014). A joint model for quotation attribution and coreference resolution. In *Proc. of the Annual Meeting of the European Chapter of the Association for Computational Linguistics*.

Auli, M. and Lopez, A. (2011). A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proc. of Annual Meeting of the Association for Computational Linguistics*.

Berg-Kirkpatrick, T., Gillick, D., and Klein, D. (2011). Jointly learning to extract and compress. In *Proc. of Annual Meeting of the Association for Computational Linguistics*.

Berrou, C., Glavieux, A., and Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding. In *Proc. of International Conference on Communications*, volume 93, pages 1064–1070.

Bertsekas, D., Hager, W., and Mangasarian, O. (1999). *Nonlinear programming*. Athena Scientific.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers.

Burkett, D. and Klein, D. (2012). Fast inference in phrase extraction models with belief propagation. In *Proc. of the North American Chapter of the Association for Computational Linguistics*, pages 29–38. Association for Computational Linguistics.

Chang, Y.-W. and Collins, M. (2011). Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proc. of Empirical Methods for Natural Language Processing*.

Clarke, J. and Lapata, M. (2008). Global Inference for Sentence Compression An Integer Linear Programming Approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

Dantzig, G. B. (1947). Maximization of a linear function of variables subject to linear inequalities. *Published in T.C. Koopmans (ed.): Activity Analysis of Production and Allocation, New York-London 1951*, pages 339–347.

Das, D., Martins, A. F. T., and Smith, N. A. (2012). An Exact Dual Decomposition Algorithm for Shallow Semantic Parsing with Constraints. In *Proc. of First Joint Conference on Lexical and Computational Semantics (*SEM)*.

Denis, P. and Baldridge, J. (2008). Specialized models and ranking for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 660–669. Association for Computational Linguistics.

References II

Dreyer, M. and Eisner, J. (2009). Graphical models over multiple strings. In *Proc. of Empirical Methods in Natural Language Processing*, pages 101–110. Association for Computational Linguistics.

Duchi, J., Tarlow, D., Elidan, G., and Koller, D. (2007). Using combinatorial optimization within max-product belief propagation. *Advances in Neural Information Processing Systems*, 19.

Durrett, G. and Klein, D. (2013). Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Eckstein, J. and Bertsekas, D. (1992). On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proc. of International Conference on Computational Linguistics*, pages 340–345.

Everett III, H. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417.

Fernandes, E. R., dos Santos, C. N., and Milidiú, R. L. (2012). Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 41–48. Association for Computational Linguistics.

Filatova, E. and Hatzivassiloglou, V. (2004). A formal model for information selection in multi-sentence text extraction. In *Proc. of International Conference on Computational Linguistics*.

Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40.

Gillick, D., Favre, B., and Hakkani-Tur, D. (2008). The icsi summarization system at tac 2008. In *Proc. of Text Understanding Conference*.

Globerson, A. and Jaakkola, T. (2008). Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *Neural Information Processing Systems*, 20.

Glowinski, R. and Le Tallec, P. (1989). *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. Society for Industrial Mathematics.

Glowinski, R. and Marroco, A. (1975). Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*, 9:41–76.

References III

Hazan, T. and Shashua, A. (2010). Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316.

Johnson, J. (2008). Equivalence of Entropy Regularization and Relative-Entropy Proximal Method. Unpublished manuscript.

Jojic, V., Gould, S., and Koller, D. (2010). Accelerated dual decomposition for MAP inference. In *International Conference of Machine Learning*.

Kantorovich, L. V. (1940). A new method of solving of some classes of extremal problems. In *Dokl. Akad. Nauk SSSR*, volume 28, pages 211–214.

Karp, R. M. (1972). *Reducibility among combinatorial problems*. Springer.

Khachiyan, L. G. (1980). Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72.

Knight, K. and Marcu, D. (2000). Statistics-based summarization—step one: Sentence compression. In *AAAI/IAAI*.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1568–1583.

Komodakis, N., Paragios, N., and Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of International Conference on Computer Vision*.

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proc. of Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Koo, T., Globerson, A., Carreras, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Empirical Methods for Natural Language Processing*.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proc. of Empirical Methods for Natural Language Processing*.

Lauritzen, S. (1996). *Graphical Models*. Clarendon Press, Oxford.

Lin, H. and Bilmes, J. (2010). Multi-document summarization via budgeted maximization of submodular functions. In *Proc. of Annual Meeting of the North American chapter of the Association for Computational Linguistics*.

MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*, volume 7. Cambridge University Press.

References IV

Martins, A. F. T. and Almeida, M. S. C. (2014). Priberam: A turbo semantic parser with second order features. In *Proc. of the International Workshop on Semantic Evaluations (SemEval); task 8: Broad-Coverage Semantic Dependency Parsing*.

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2011a). An Augmented Lagrangian Approach to Constrained MAP Inference. In *Proc. of International Conference on Machine Learning*.

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2014). AD^3: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models. *Journal of Machine Learning Research (to appear)*.

Martins, A. F. T. and Smith, N. A. (2009). Summarization with a Joint Model for Sentence Extraction and Compression. In *North American Chapter of the Association for Computational Linguistics: Workshop on Integer Linear Programming for NLP*.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011b). Dual Decomposition with Many Overlapping Components. In *Proc. of Empirical Methods for Natural Language Processing*.

Martins, A. F. T., Smith, N. A., Xing, E. P., Figueiredo, M. A. T., and Aguiar, P. M. Q. (2010). Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In *Proc. of Empirical Methods for Natural Language Processing*.

McAllester, D., Collins, M., and Pereira, F. (2008). Case-factor diagrams for structured probabilistic modeling. *Journal of Computer and System Sciences*, 74(1):84–96.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic constraints. In *Proc. of Annual Meeting of the European Chapter of the Association for Computational Linguistics*.

McDonald, R. and Satta, G. (2007). On the complexity of non-projective data-driven dependency parsing. In *Proc. of International Conference on Parsing Technologies*.

McDonald, R. T., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proc. of Empirical Methods for Natural Language Processing*.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., and Marinov, S. (2006). Labeled pseudo-projective dependency parsing with support vector machines. In *Procs. of International Conference on Natural Language Learning*.

Nocedal, J. and Wright, S. (1999). *Numerical optimization*. Springer verlag.

References V

O'Keefe, T., Pareti, S., Curran, J. R., Koprinska, I., and Honnibal, M. (2012). A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799. Association for Computational Linguistics.

Pareti, S. (2012). A database of attribution relations. In *LREC*, pages 3213–3217.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1):107–136.

Roth, D. and Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. In *International Conference on Natural Language Learning*.

Rush, A. and Collins, M. (2012). A Tutorial on Dual Decomposition and Lagrangian Relaxation for Inference in Natural Language Processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Rush, A., Sontag, D., Collins, M., and Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. In *Proc. of Empirical Methods for Natural Language Processing*.

Rush, A. M. and Collins, M. (2011). Exact decoding of syntactic translation models through lagrangian relaxation. In *Proc. of Annual Meeting on Association for Computational Linguistics*.

Smith, D. and Eisner, J. (2008). Dependency parsing by belief propagation. In *Proc. of Empirical Methods for Natural Language Processing*.

Smith, N. A. (2011). *Linguistic Structure Prediction*, volume 13 of *Synthesis Lectures on Human Language Technologies*. Morgan and Claypool.

Wainwright, M. and Jordan, M. (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers.

Wainwright, M. J., Jaakkola, T., and Willsky, A. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335.

Woodsend, K. and Lapata, M. (2012). Multiple aspect summarization using integer linear programming. In *Proc. of Empirical Methods in Natural Language Processing*.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Generalized belief propagation. In *Neural Information Processing Systems*.

Yih, W.-t., Goodman, J., Vanderwende, L., and Suzuki, H. (2007). Multi-document summarization by maximizing informative content-words. In *Proc. of International Joint Conference on Artifical Intelligence*.