Probability and Structure in Natural Language Processing

Noah Smith, Carnegie Mellon University

2012 International Summer School in Language and Speech Technologies

Slides Online!

- http://tinyurl.com/psnlp2012
- (I'll post the slides after each lecture.)

Where We Left Off

- Graphical models … inference ... "max" inference and decoding with linear models.
- Five views of decoding:
  1. Probabilistic graphical models
  2. Polytopes and integer linear programming
  3. ?
  4. ?
  5. ?

3. Weighted Parsing

Grammars

- Grammars are often associated with natural language parsing, but they are extremely powerful for imposing constraints.
- We can add weights to them.
  - HMMs are a kind of weighted regular grammar (closely connected to WFSAs)
  - PCFGs are a kind of weighted CFG
  - Many, many more.
- Weighted parsing: find the maximum-weighted derivation for a string **x**.

Decoding as Weighted Parsing

- Every valid **y** is a grammatical derivation (parse) for **x**.
  - HMM: sequence of "grammatical" states is one allowed by the transition table.
- Augment parsing algorithms with weights and find the best parse.

The Viterbi algorithm is an instance of recognition by a weighted grammar!

BIO Tagging as a CFG

$$
\begin{array}{lllllllllll}
N_{\rightarrow} & \rightarrow & B\ R_B & R_B & \rightarrow & B\ R_B & R_I & \rightarrow & B\ R_B & R_O & \rightarrow & B\ R_B \\
N_{\rightarrow} & \rightarrow & O\ R_O & R_B & \rightarrow & O\ R_O & R_I & \rightarrow & O\ R_O & R_O & \rightarrow & O\ R_O \\
 & & & R_B & \rightarrow & I\ R_I & R_I & \rightarrow & I\ R_I & & & \\
 & & & R_B & \rightarrow & \epsilon & R_I & \rightarrow & \epsilon & R_O & \rightarrow & \epsilon
\end{array}
$$

$$
\forall x \in \Sigma, \quad B \rightarrow x \qquad I \rightarrow x \qquad O \rightarrow x
$$

- Weighted (or probabilistic) CKY is a dynamic programming algorithm very similar in structure to classical CKY.

4. Paths and Hyperpaths

Best Path

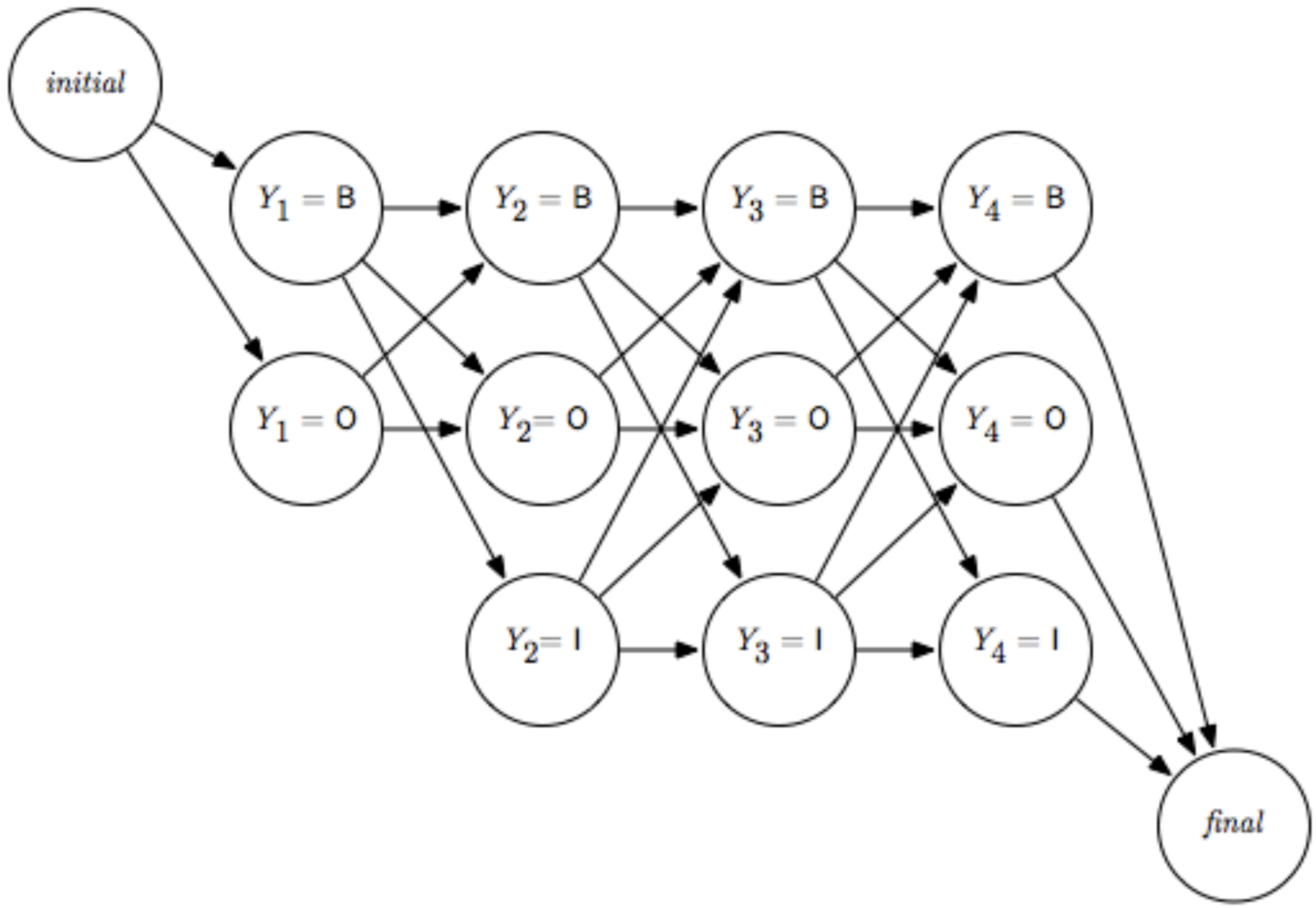
- General idea: take $\boldsymbol{x}$ and build a graph.
- Score of a path factors into the edges.

$$\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \sum_{e \in \text{Edges}} \mathbf{f}(e) \mathbf{1}\{e \text{ is crossed by } \boldsymbol{y}\text{'s path}\}$$

- Decoding is finding the *best* path.

The Viterbi algorithm is an instance of finding a best path!

"Lattice" View of Viterbi

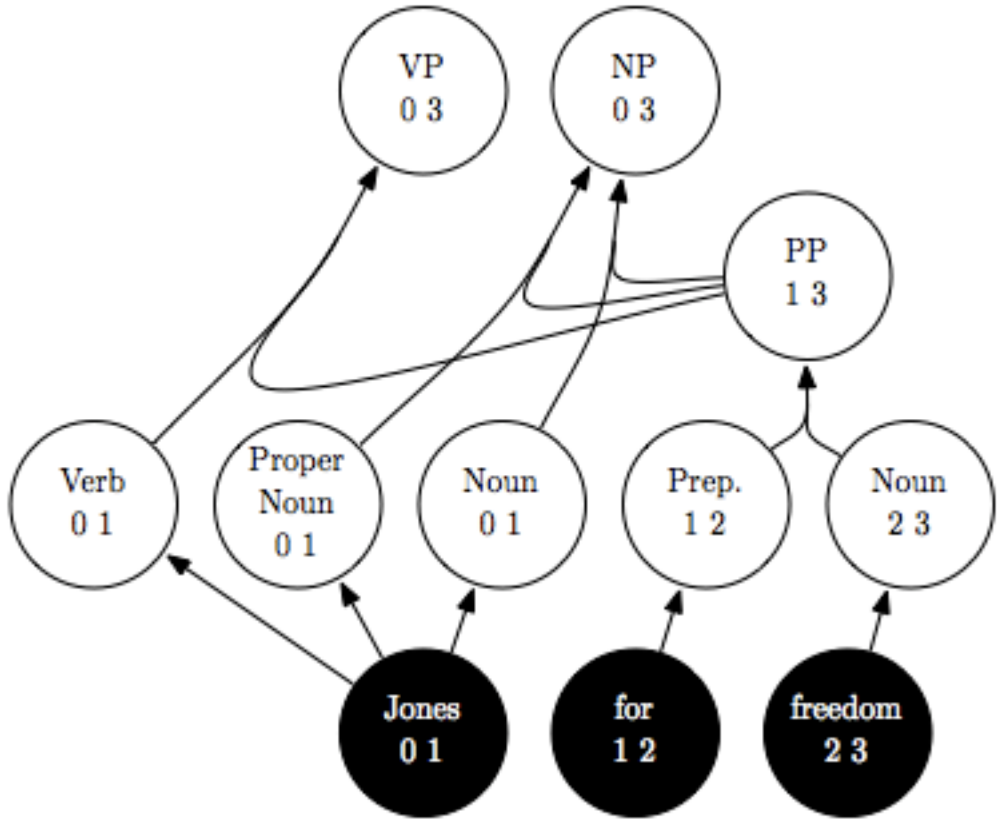


Minimum Cost Hyperpath

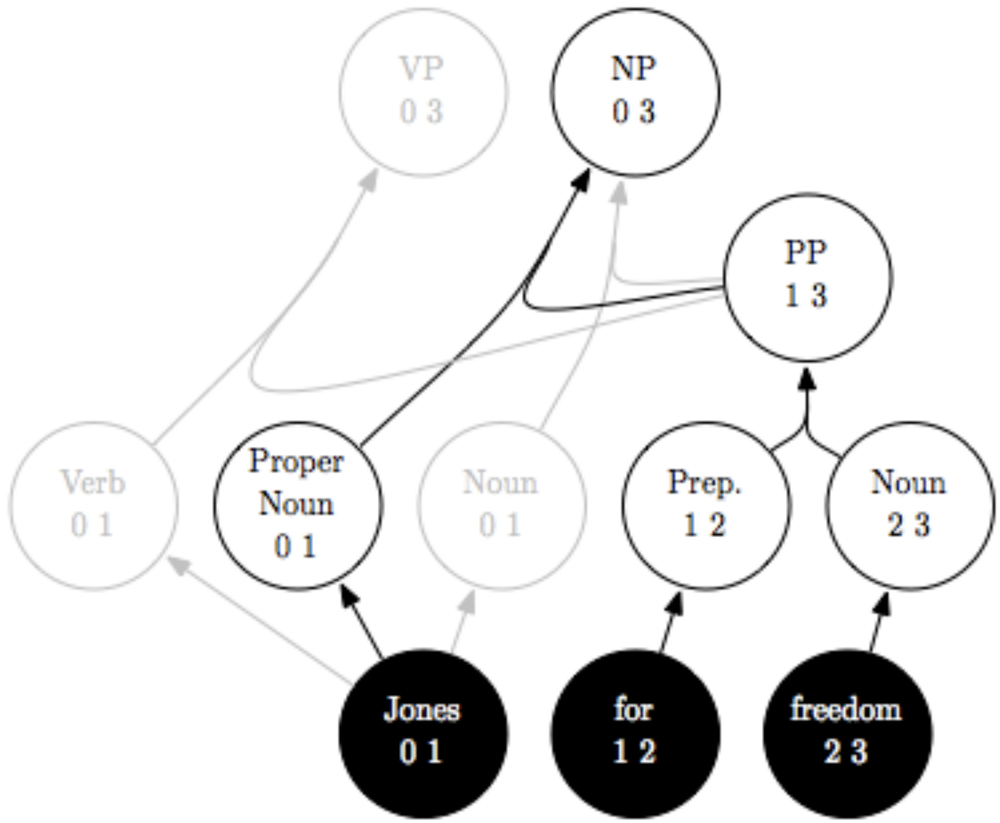
- General idea: take ***x*** and build a hypergraph.
- Score of a hyperpath factors into the hyperedges.
- Decoding is finding the best *hyperpath*.

- This connection was elucidated by Klein and Manning (2002).

Parsing as a Hypergraph

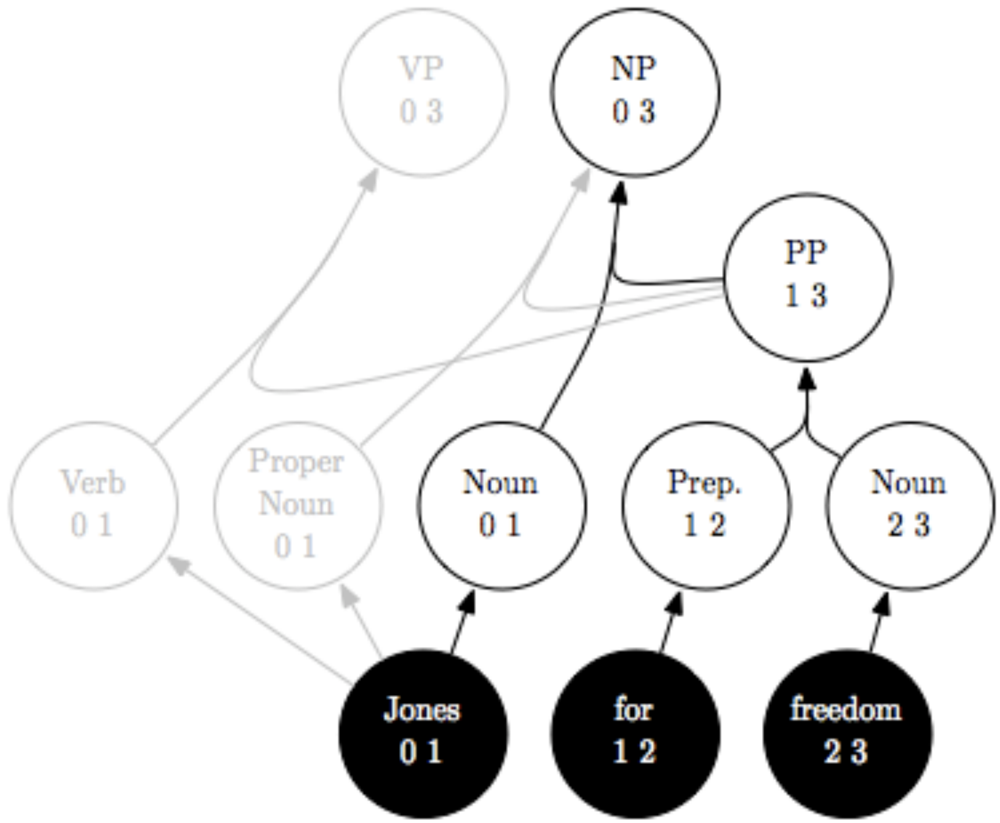


Parsing as a Hypergraph



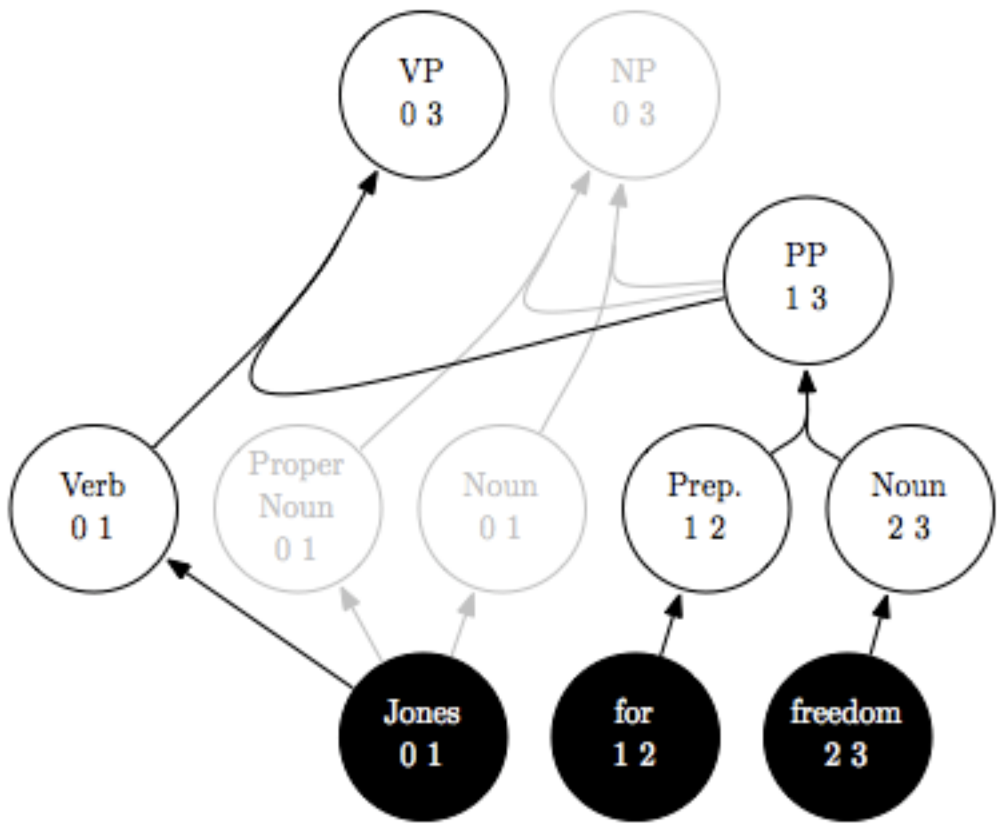
cf. "Dean for democracy"

Parsing as a Hypergraph



Forced to work on his thesis, sunshine streaming in the window, Mike experienced a ...

Parsing as a Hypergraph



Forced to work on his thesis, sunshine streaming in the window, Mike began to ...

Why Hypergraphs?

- Useful, compact encoding of the hypothesis space.
  - Build hypothesis space using local features, maybe do some filtering.
  - Pass it off to another module for more fine-grained scoring with richer or more expensive features.

5. Weighted Logic Programming

Logic Programming

- Start with a set of axioms and a set of inference rules.

$$\forall A, C, \quad \text{ancestor}(A, C) \Leftarrow \text{parent}(A, C)$$
$$\forall A, C, \quad \text{ancestor}(A, C) \Leftarrow \bigvee_{B} \text{ancestor}(A, B) \wedge \text{parent}(B, C)$$

- The goal is to prove a specific theorem, *goal*.
- Many approaches, but we assume a *deductive* approach.
  - Start with axioms, iteratively produce more theorems.

$$
\begin{array}{rl}
& \mathsf{label\text{-}bigram}(\text{“B”}, \text{“B”}) \\
& \mathsf{label\text{-}bigram}(\text{“B”}, \text{“I”}) \\
& \mathsf{label\text{-}bigram}(\text{“B”}, \text{“O”}) \\
& \mathsf{label\text{-}bigram}(\text{“I”}, \text{“B”}) \\
& \mathsf{label\text{-}bigram}(\text{“I”}, \text{“I”}) \\
& \mathsf{label\text{-}bigram}(\text{“I”}, \text{“O”}) \\
& \mathsf{label\text{-}bigram}(\text{“O”}, \text{“B”}) \\
& \mathsf{label\text{-}bigram}(\text{“O”}, \text{“O”}) \\
\forall x \in \Sigma, & \mathsf{labeled\text{-}word}(x, \text{“B”}) \\
\forall x \in \Sigma, & \mathsf{labeled\text{-}word}(x, \text{“I”}) \\
\forall x \in \Sigma, & \mathsf{labeled\text{-}word}(x, \text{“O”})
\end{array}
$$

$$
\begin{array}{rcl}
\forall \ell \in \Lambda, \;\; \mathsf{v}(\ell, 1) & = & \mathsf{labeled\text{-}word}(x_1, \ell) \\
\forall \ell \in \Lambda, \;\; \mathsf{v}(\ell, i) & = & \bigvee_{\ell' \in \Lambda} \mathsf{v}(\ell', i-1) \wedge \mathsf{label\text{-}bigram}(\ell', \ell) \wedge \mathsf{labeled\text{-}word}(x_i, \ell) \\
\mathsf{goal} & = & \bigvee_{\ell \in \Lambda} \mathsf{v}(\ell, n)
\end{array}
$$

Weighted Logic Programming

- Twist: axioms have weights.
- Want the proof of *goal* with the best score:

$$\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) = \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \sum_{a \in \text{Axioms}} \mathbf{f}(a) \mathit{freq}(a; \boldsymbol{y})$$

- Note that axioms can be used more than once in a proof (**y**).

Whence WLP?

- Shieber, Schabes, and Pereira (1995): many parsing algorithms can be understood in the same deductive logic framework.
- Goodman (1999): add weights, get many useful NLP algorithms.
- Eisner, Goldlust, and Smith (2004, 2005): semiring-generic algorithms, Dyna.

Dynamic Programming

- Most views (exception is polytopes) can be understood as DP algorithms.
  - The low-level *procedures* we use are often DP.
  - Even DP is too high-level to know the best way to implement.
- DP does *not* imply polynomial time and space!
  - Most common approximations when the desired state space is too big: beam search, cube pruning, agendas with early stopping, ...
  - Other views suggest others.

Summary

- Decoding is the general problem of choosing a complex structure.
  - Linguistic analysis, machine translation, speech recognition, …
  - Statistical models are usually involved (not necessarily probabilistic).
- No perfect general view, but much can be gained through a combination of views.

Lecture 4: Supervised Learning

Quick Recap

- Graphical models
- Inference
- Decoding for models of structure

- Finally, we get to *learning*.
  - Today, assume a collection of N pairs (**x**, **y**); supervised learning with complete data.

Loss

- Let h be a hypothesis (an instantiated, predictive model).
- loss(**x**, **y**; h) = a measure of how badly h performs on input **x** if **y** is the correct output.

- How to decide what "loss" should be?
  1. computational expense
  2. knowledge of actual costs of errors
  3. formal foundations enabling theoretical guarantees

Risk

- There is some true distribution p* over input, output pairs (**X**, **Y**).
- Under that distribution, what do we expect h's loss to be?

$$\mathbb{E}_{p^*(\boldsymbol{X},\boldsymbol{Y})}[loss(\boldsymbol{X},\boldsymbol{Y};h)]$$

- We don't have p*, but we have the empirical distribution, giving **empirical risk**:

$$\mathbb{E}_{\tilde{p}(\boldsymbol{X},\boldsymbol{Y})}[loss(\boldsymbol{X},\boldsymbol{Y};h)] = \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i,\boldsymbol{y}_i;h)$$

Empirical Risk Minimization

- Provides a criterion to decide on h:

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h)$$

- Background preferences over h can be included in **regularized** empirical risk minimization:

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h) + R(h)$$

Parametric Assumptions

- Typically we do not move in "h-space," but rather in the space of continuously-parameterized predictors.

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h) + R(h)$$

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

Three Kinds of Loss Functions

- Error
  - Could be zero-one, or task-specific.
  - Mean squared error makes sense for *continuous* predictions and is used in *regression*.
- Log loss
  - Probabilistic interpretation ("likelihood")
- Hinge loss
  - Geometric interpretation ("margin")

Log Loss (First Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y})$$

- Maximum likelihood estimation: R(**w**) is 0 for models in the family, +∞ for other models.
- Maximum *a posteriori* (MAP) estimation: R(**w**) is –log p(**w**)
- Often called **generative** modeling.

Log Loss (First Version)

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y})$$

Examples:

- N-gram language models
- Supervised HMM taggers
- Charniak, Collins, and Stanford parsers

Log Loss (First Version)

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y})$$

Computationally …

- Convex and differentiable.
- Closed form for directed, multinomial-based models $p_{\mathbf{w}}$.
  - Count and normalize!
- In other cases, requires posterior inference, which can be expensive depending on the model's structure.
- Linear decoding (for some parameterizations).

Log Loss (First Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y})$$

Error …

- No notion of error.
- Learner wins by moving as much probability mass as possible to training examples.

Log Loss (First Version)

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y})$$

Guarantees…

- Consistency: if the true model is in the right family, enough data will lead you to it.

Log Loss (First Version)

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad -\log p_{\mathbf{w}}(\boldsymbol{x}, \boldsymbol{y})$$

Different parameterizations …

- Multinomials (BN-like): $-\sum_{\boldsymbol{e}} freq(\boldsymbol{e}; \boldsymbol{x}, \boldsymbol{y}) \underbrace{\log p_{\boldsymbol{e}}}_{w_{\boldsymbol{e}}}$
- Global log-linear (MN-like): $-\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) + \log \sum_{\boldsymbol{x}', \boldsymbol{y}'} \exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}', \boldsymbol{y}')$
- Locally normalized log-linear:

$$-\sum_{\boldsymbol{e}} freq(\boldsymbol{e}; \boldsymbol{x}, \boldsymbol{y}) \left(\mathbf{w}^\top \mathbf{g}(\boldsymbol{e}) - \log \sum_{\boldsymbol{e}' \in \mathcal{C}(\boldsymbol{e})} \exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{e}') \right)$$

Reflections on Generative Models

- Most early solutions are generative.
- Most unsupervised approaches are generative.
- Some people only believe in generative models.
- Sometimes estimators are not as easy as they seem ("deficiency").
- Start here if there's a sensible generative story.
  - You can always use a "better" loss function with the same linear model later on.

Zero-One Loss

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad \mathbf{1}\{h_{\mathbf{w}}(\boldsymbol{x}) \neq \boldsymbol{y}\}$$

Zero-One Loss

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad \mathbf{1}\{h_{\mathbf{w}}(\boldsymbol{x}) \neq \boldsymbol{y}\}$$

Computationally:

- Piecewise constant. ☹

Error: ☺

Guarantees: none

Error as Loss

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} loss(\boldsymbol{x}_i, \boldsymbol{y}_i; h_{\mathbf{w}}) + R(\mathbf{w})$$

$$loss(\boldsymbol{x}, \boldsymbol{y}; h_{\mathbf{w}}) \quad = \quad error(h_{\mathbf{w}}(\boldsymbol{x}); \boldsymbol{y})$$

Generalizes zero-one, same difficulties.

Example: Bleu-score maximization in machine translation, with “MERT” line search.

Comparison

	Generative (Log Loss)	Error as Loss
Computation	Convex optimization.	Optimizing a piecewise constant function.
Error-awareness	None	☺
Guarantees	Consistency.	None.

Discriminative Learning

- Various loss functions between log loss and error.
- Three commonly used in NLP:
  - Conditional log loss ("max ent," CRFs)
  - Hinge loss (structural SVMs)
  - Perceptron's loss
- We' ll discuss each, compare, and unify.