# Spectral Learning Algorithms for Natural Language Processing

Shay Cohen[1], Michael Collins[1], Dean Foster[2], Karl Stratos[1] and Lyle Ungar[2]

[1]Columbia University

[2]University of Pennsylvania

June 10, 2013

## Latent-variable Models

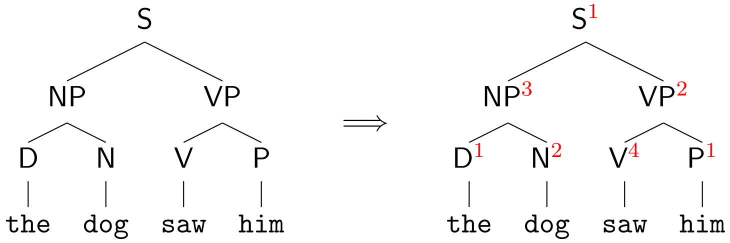
Latent-variable models are used in many areas of NLP, speech, etc.:

- Latent-variable PCFGs (Matsuzaki et al.; Petrov et al.)
- Hidden Markov Models
- Naive Bayes for clustering
- Lexical representations: Brown clustering, Saul and Pereira, etc.
- Alignments in statistical machine translation
- Topic modeling
- etc. etc.

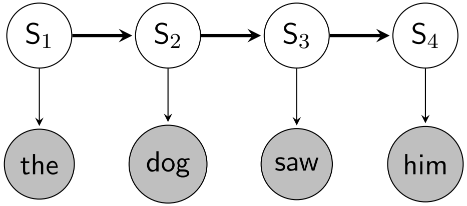
The Expectation-maximization (EM) algorithm is generally used for estimation in these models (Depmster et al., 1977)

Other relevant algorithms: cotraining, clustering methods

## Example 1: Latent-Variable PCFGs (Matsuzaki et al., 2005; Petrov et al., 2006)



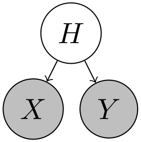
## Example 2: Hidden Markov Models



Parameterized by $\pi(s)$, $t(s|s')$ and $o(w|s)$

EM is used for learning the parameters

## Example 3: Naïve Bayes

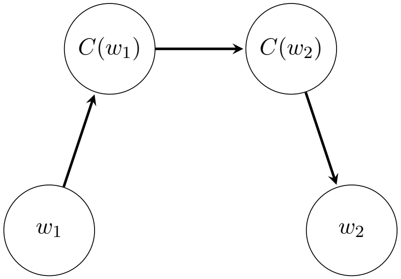


(the, dog)
(I, saw)
(ran, to)
(John, was)
⋮

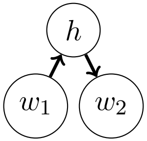
$$p(h, x, y) = p(h) \times p(x|h) \times p(y|h)$$

- EM can be used to estimate parameters

# Example 4: Brown Clustering and Related Models

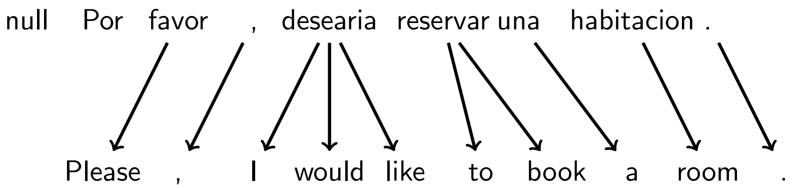


$$p(w_2|w_1) = p(C(w_2)|C(w_1)) \times p(w_2|C(w_2))$$ (Brown et al., 1992)



$$p(w_2|w_1) = \sum_h p(h|w_1) \times p(w_2|h)$$ (Saul and Pereira, 1997)

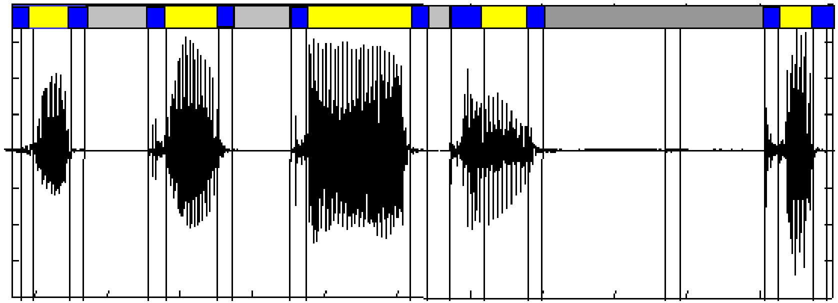
## Example 5: IBM Translation Models



Hidden variables are alignments

EM used to estimate parameters

## Example 6: HMMs for Speech



Phoneme boundaries are hidden variables

## Co-training (Blum and Mitchell, 1998)

Examples come in pairs

Each view is assumed to be sufficient for classification

E.g. Collins and Singer (1999):

. . . , says Mr. Cooper, a vice president of . . .

- View 1. Spelling features: "Mr.", "Cooper"
- View 2. Contextual features: appositive=president

## Spectral Methods

Basic idea: replace EM (or co-training) with methods based on matrix decompositions, in particular singular value decomposition (SVD)

SVD: given matrix A with m rows, n columns, approximate as

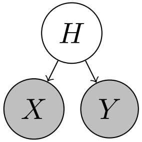
$$A_{jk} \approx \sum_{h=1}^{d} \sigma_h U_{jh} V_{jh}$$

where $\sigma_h$ are “singular values”

$U$ and $V$ are $m \times d$ and $n \times d$ matrices

Remarkably, can find the optimal rank-$d$ approximation efficiently

## Similarity of SVD to Naïve Bayes



$$P(X = x, Y = y) = \sum_{h=1}^{d} p(h)p(x|h)p(y|h)$$

$$A_{jk} \approx \sum_{h=1}^{d} \sigma_h U_{jh} V_{jh}$$

- SVD approximation minimizes squared loss, not log-loss
- $\sigma_h$ not interpretable as probabilities
- $U_{jh}$, $V_{jh}$ may be positive or negative, not probabilities

BUT we can still do **a lot** with SVD (and higher-order, tensor-based decompositions)

## CCA vs. Co-training

- Co-training assumption: 2 views, each sufficient for classification
- Several heuristic algorithms developed for this setting
- Canonical correlation analysis:
  - Take paired examples $x^{(i),1}, x^{(i),2}$
  - Transform to $z^{(i),1}, z^{(i),2}$
  - $z$'s are linear projections of the $x$'s
  - Projections are chosen to maximize correlation between $z^1$ and $z^2$
  - Solvable using SVD!
  - Strong guarantees in several settings

## One Example of CCA: Lexical Representations

- $x \in \mathbb{R}^d$ is a word

$$\texttt{dog} = (0, 0, \dots, 0, 1, 0, \dots, 0, 0) \in \mathbb{R}^{200{,}000}$$

- $y \in \mathbb{R}^{d'}$ is its context information

$$\texttt{dog-context} = (11, 0, \dots 0, 917, 3, 0, \dots 0) \in \mathbb{R}^{400{,}000}$$

- Use CCA on $x$ and $y$ to derive $\underline{x} \in \mathbb{R}^k$

$$\underline{\texttt{dog}} = (0.03, -1.2, \dots 1.5) \in \mathbb{R}^{100}$$

## Spectral Learning of HMMs and L-PCFGs

Simple algorithms: require SVD, then method of moments in low-dimensional space

Close connection to CCA

Guaranteed to learn (unlike EM) under assumptions on singular values in the SVD

## Spectral Methods in NLP

- Balle, Quattoni, Carreras, ECML 2011 (learning of finite-state transducers)
- Luque, Quattoni, Balle, Carreras, EACL 2012 (dependency parsing)
- Dhillon et al, 2012 (dependency parsing)
- Cohen et al 2012, 2013 (latent-variable PCFGs)

## Overview

## Matrices

$$A \in \mathbb{R}^{m \times n}$$

$$m \left\{ \overbrace{\begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix}}^{n} \right.$$

$$A = \begin{bmatrix} 3 & 1 & 4 \\ 0 & 2 & 5 \end{bmatrix}$$

"matrix of dimensions $m$ by $n$"

$$A \in \mathbb{R}^{2 \times 3}$$

## Vectors

$$u \in \mathbb{R}^n$$

$$n\left\{\begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix}\right.$$

$$u = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

"vector of dimension $n$"

$$u \in \mathbb{R}^3$$

## Matrix Transpose

- $A^\top \in \mathbb{R}^{n\times m}$ is the *transpose* of $A \in \mathbb{R}^{m\times n}$

$$A = \begin{bmatrix} 3 & 1 & 4 \\ 0 & 2 & 5 \end{bmatrix} \implies A^\top = \begin{bmatrix} 3 & 0 \\ 1 & 2 \\ 4 & 5 \end{bmatrix}$$

## Matrix Multiplication

Matrices $B \in \mathbb{R}^{m \times d}$ and $C \in \mathbb{R}^{d \times n}$

$$\underbrace{A}_{m \times n} = \underbrace{B}_{m \times d} \underbrace{C}_{d \times n}$$

## Overview

## Singular Value Decomposition (SVD)

$$\underbrace{A}_{m\times n} \overset{\text{SVD}}{=} \sum_{i=1}^{d} \underbrace{\sigma^i}_{scalar} \underbrace{\underbrace{u^i}_{m\times 1} \underbrace{(v^i)^\top}_{1\times n}}_{m\times n}$$

- $d = \min(m, n)$

## Singular Value Decomposition (SVD)

$$\underbrace{A}_{m\times n} \overset{\text{SVD}}{=} \sum_{i=1}^{d} \underbrace{\sigma^i}_{scalar} \underbrace{\underbrace{u^i}_{m\times 1} \underbrace{(v^i)^\top}_{1\times n}}_{m\times n}$$

- $d = \min(m, n)$
- $\sigma^1 \geq \ldots \geq \sigma^d \geq 0$

## Singular Value Decomposition (SVD)

$$\underbrace{A}_{m\times n} \overset{\text{SVD}}{=} \sum_{i=1}^{d} \underbrace{\sigma^i}_{scalar} \underbrace{\underbrace{u^i}_{m\times 1} \underbrace{(v^i)^\top}_{1\times n}}_{m\times n}$$

- $d = \min(m, n)$
- $\sigma^1 \geq \ldots \geq \sigma^d \geq 0$
- $u^1 \ldots u^d \in \mathbb{R}^m$ are orthonormal:

$$\left|\left|u^i\right|\right|_2 = 1 \qquad u^i \cdot u^j = 0 \ \ \forall i \neq j$$

## Singular Value Decomposition (SVD)

$$\underbrace{A}_{m\times n} \overset{\text{SVD}}{=} \sum_{i=1}^{d} \underbrace{\sigma^i}_{scalar} \underbrace{\underbrace{u^i}_{m\times 1} \underbrace{(v^i)^\top}_{1\times n}}_{m\times n}$$

- $d = \min(m, n)$
- $\sigma^1 \geq \ldots \geq \sigma^d \geq 0$
- $u^1 \ldots u^d \in \mathbb{R}^m$ are orthonormal:
$$\left|\left|u^i\right|\right|_2 = 1 \qquad u^i \cdot u^j = 0 \;\; \forall i \neq j$$
- $v^1 \ldots v^d \in \mathbb{R}^n$ are orthonormal:
$$\left|\left|v^i\right|\right|_2 = 1 \qquad v^i \cdot v^j = 0 \;\; \forall i \neq j$$

## SVD in Matrix Form

$$\underbrace{A}_{m\times n} \overset{\text{SVD}}{=} \underbrace{U}_{m\times d}\,\underbrace{\Sigma}_{d\times d}\,\underbrace{V^\top}_{d\times n}$$

$$U = \begin{bmatrix} | & & | \\ u^1 & \dots & u^d \\ | & & | \end{bmatrix} \in \mathbb{R}^{m\times d} \qquad \Sigma = \begin{bmatrix} \sigma^1 & & 0 \\ & \ddots & \\ 0 & & \sigma^d \end{bmatrix} \in \mathbb{R}^{d\times d}$$

$$V = \begin{bmatrix} | & & | \\ v^1 & \dots & v^d \\ | & & | \end{bmatrix} \in \mathbb{R}^{n\times d}$$

## Matrix Rank

$$A \in \mathbb{R}^{m \times n}$$

$$\text{rank}(A) \leq \min(m, n)$$

- $\text{rank}(A) :=$ number of linearly independent columns in $A$

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{bmatrix}$$

rank 2

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

rank 3
(full-rank)

## Matrix Rank: Alternative Definition

- rank$(A)$ := number of positive singular values of $A$

$$\begin{bmatrix}1 & 1 & 2\\1 & 2 & 2\\1 & 1 & 2\end{bmatrix} \qquad \begin{bmatrix}1 & 1 & 2\\1 & 2 & 2\\1 & 1 & 3\end{bmatrix}$$

$$\Sigma = \begin{bmatrix}4.53 & 0 & 0\\0 & 0.7 & 0\\0 & 0 & 0\end{bmatrix} \qquad \Sigma = \begin{bmatrix}5 & 0 & 0\\0 & 0.98 & 0\\0 & 0 & 0.2\end{bmatrix}$$

rank 2

rank 3
(full-rank)

## SVD and Low-Rank Matrix Approximation

- Suppose we want to find $B^*$ such that

$$B^* = \arg\min_{B:\, \mathsf{rank}(B)=r} \sum_{jk} (A_{jk} - B_{jk})^2$$

- Solution:

$$B^* = \sum_{i=1}^{r} \sigma^i u^i (v^i)^\top$$

## SVD in Practice

- Black box, e.g., in Matlab
  - Input: matrix $A$, output: scalars $\sigma^1 \dots \sigma^d$, vectors $u^1 \dots u^d$ and $v^1 \dots v^d$
  - Efficient implementations
  - Approximate, randomized approaches also available
- Can be used to solve a variety of optimization problems
  - For instance, Canonical Correlation Analysis (CCA)

## Overview

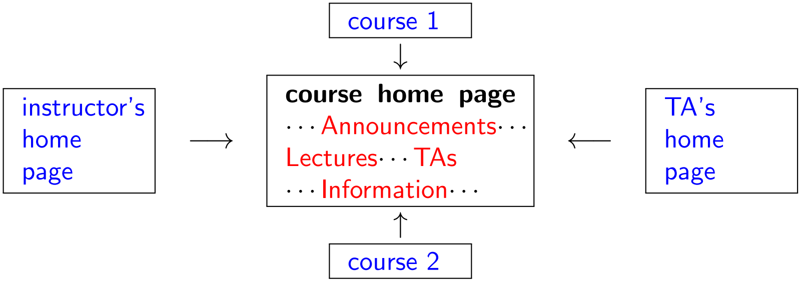
## Canonical Correlation Analysis (CCA)

- Data consists of paired samples: $(x^{(i)}, y^{(i)})$ for $i = 1 \dots n$
- As in co-training, $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \mathbb{R}^{d'}$ are two "views" of a sample point

| View 1 | View 2 |
|---|---|
| $x^{(1)} = (1, 0, 0, 0)$ | $y^{(1)} = (1, 0, 0, 1, 0, 1, 0)$ |
| $x^{(2)} = (0, 0, 1, 0)$ | $y^{(2)} = (0, 1, 0, 0, 0, 0, 1)$ |
| $\vdots$ | $\vdots$ |
| $x^{(100000)} = (0, 1, 0, 0)$ | $y^{(100000)} = (0, 0, 1, 0, 1, 1, 1)$ |

## Example of Paired Data: Webpage Classification (Blum and Mitchell, 98)

- Determine if a webpage is an course home page



- View 1. Words on the page: "Announcements", "Lectures"
- View 2. Identities of pages pointing to the page: instructror's home page, related course home pages
- Each view is sufficient for the classification!

## Example of Paired Data: Named Entity Recognition (Collins and Singer, 99)

- Identify an entity's type as either Organization, Person, or Location

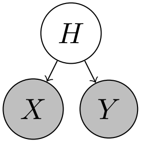
  $\boxed{\text{. . . , says Mr. Cooper, a vice president of . . .}}$

- View 1. Spelling features: "Mr.", "Cooper"

- View 2. Contextual features: appositive=president

- Each view is sufficient to determine the entity's type!

## Example of Paired Data: Bigram Model



(the, dog)
(I, saw)
(ran, to)
(John, was)
⋮

$$p(h, x, y) = p(h) \times p(x|h) \times p(y|h)$$

- EM can be used to estimate the parameters of the model
- Alternatively, CCA can be used to derive vectors which can be used in a predictor

$$\text{the} \Longrightarrow \begin{bmatrix} 0.3 \\ \vdots \\ 1.1 \end{bmatrix} \qquad \text{dog} \Longrightarrow \begin{bmatrix} -1.5 \\ \vdots \\ -0.4 \end{bmatrix}$$

## Projection Matrices

- Project samples to lower dimensional space

$$x \in \mathbb{R}^d \Longrightarrow x' \in \mathbb{R}^p$$

  - If $p$ is small, we can learn with far fewer samples!

## Projection Matrices

- Project samples to lower dimensional space

$$x \in \mathbb{R}^d \Longrightarrow x' \in \mathbb{R}^p$$

  - If $p$ is small, we can learn with far fewer samples!

- CCA finds projection matrices $A \in \mathbb{R}^{d \times p}$, $B \in \mathbb{R}^{d' \times p}$

- The new data points are $a^{(i)} \in \mathbb{R}^p$, $b^{(i)} \in \mathbb{R}^p$ where

$$\underbrace{a^{(i)}}_{p \times 1} = \underbrace{A^\top}_{p \times d} \underbrace{x^{(i)}}_{d \times 1} \qquad \underbrace{b^{(i)}}_{p \times 1} = \underbrace{B^\top}_{p \times d'} \underbrace{y^{(i)}}_{d' \times 1}$$

## Mechanics of CCA: Step 1

- Compute $\hat{C}_{XY} \in \mathbb{R}^{d \times d'}$, $\hat{C}_{XX} \in \mathbb{R}^{d \times d}$, and $\hat{C}_{YY} \in \mathbb{R}^{d' \times d'}$

$$[\hat{C}_{XY}]_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \bar{x}_j)(y_k^{(i)} - \bar{y}_k)$$

where $\bar{x} = \sum_i x^{(i)}/n$ and $\bar{y} = \sum_i y^{(i)}/n$

## Mechanics of CCA: Step 1

- Compute $\hat{C}_{XY} \in \mathbb{R}^{d \times d'}$, $\hat{C}_{XX} \in \mathbb{R}^{d \times d}$, and $\hat{C}_{YY} \in \mathbb{R}^{d' \times d'}$

$$[\hat{C}_{XY}]_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \bar{x}_j)(y_k^{(i)} - \bar{y}_k)$$
$$[\hat{C}_{XX}]_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \bar{x}_j)(x_k^{(i)} - \bar{x}_k)$$

where $\bar{x} = \sum_i x^{(i)}/n$ and $\bar{y} = \sum_i y^{(i)}/n$

## Mechanics of CCA: Step 1

- Compute $\hat{C}_{XY} \in \mathbb{R}^{d \times d'}$, $\hat{C}_{XX} \in \mathbb{R}^{d \times d}$, and $\hat{C}_{YY} \in \mathbb{R}^{d' \times d'}$

$$[\hat{C}_{XY}]_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \bar{x}_j)(y_k^{(i)} - \bar{y}_k)$$

$$[\hat{C}_{XX}]_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \bar{x}_j)(x_k^{(i)} - \bar{x}_k)$$

$$[\hat{C}_{YY}]_{jk} = \frac{1}{n} \sum_{i=1}^{n} (y_j^{(i)} - \bar{y}_j)(y_k^{(i)} - \bar{y}_k)$$

where $\bar{x} = \sum_i x^{(i)}/n$ and $\bar{y} = \sum_i y^{(i)}/n$

## Mechanics of CCA: Step 2

- Do SVD on $\hat{C}_{XX}^{-1/2}\hat{C}_{XY}\hat{C}_{YY}^{-1/2} \in \mathbb{R}^{d \times d'}$

$$\hat{C}_{XX}^{-1/2}\hat{C}_{XY}\hat{C}_{YY}^{-1/2} \overset{\text{SVD}}{=} U\Sigma V^\top$$

Let $U_p \in \mathbb{R}^{d \times p}$ be the top $p$ left singular vectors. Let $V_p \in \mathbb{R}^{d' \times p}$ be the top $p$ right singular vectors.

## Mechanics of CCA: Step 3

- Define projection matrices $A \in \mathbb{R}^{d \times p}$ and $B \in \mathbb{R}^{d' \times p}$

$$A = \hat{C}_{XX}^{-1/2} U_p \qquad B = \hat{C}_{YY}^{-1/2} V_p$$

- Use $A$ and $B$ to project each $(x^{(i)}, y^{(i)})$ for $i = 1 \ldots n$:

$$x^{(i)} \in \mathbb{R}^d \Longrightarrow A^\top x^{(i)} \in \mathbb{R}^p$$
$$y^{(i)} \in \mathbb{R}^{d'} \Longrightarrow B^\top y^{(i)} \in \mathbb{R}^p$$

## Input and Output of CCA

$$x^{(i)} = (0, 0, 0, 1, 0, 0, 0, 0, 0, \ldots, 0) \in \mathbb{R}^{50{,}000}$$

$$\downarrow$$

$$a^{(i)} = (-0.3 \ldots 0.1) \in \mathbb{R}^{100}$$

$$y^{(i)} = (497, 0, 1, 12, 0, 0, 0, 7, 0, 0, 0, 0, \ldots, 0, 58, 0) \in \mathbb{R}^{120{,}000}$$

$$\downarrow$$

$$b^{(i)} = (-0.7 \ldots - 0.2) \in \mathbb{R}^{100}$$

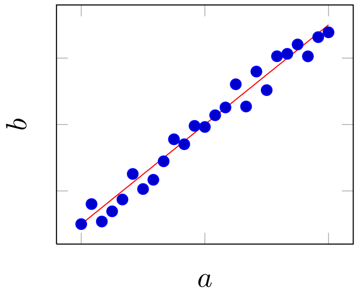
## Overview

## Justification of CCA: Correlation Coefficients

- Sample correlation coefficient for $a_1 \ldots a_n \in \mathbb{R}$ and $b_1 \ldots b_n \in \mathbb{R}$ is

$$\text{Corr}(\{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n) = \frac{\sum_{i=1}^n (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^n (a_i - \bar{a})^2}\sqrt{\sum_{i=1}^n (b_i - \bar{b})^2}}$$

where $\bar{a} = \sum_i a_i/n$, $\bar{b} = \sum_i b_i/n$



Correlation $\approx 1$

## Simple Case: $p = 1$

- CCA projection matrices are vectors $u_1 \in \mathbb{R}^d$, $v_1 \in \mathbb{R}^{d'}$
- Project $x^{(i)}$ and $y^{(i)}$ to scalars $u_1 \cdot x^{(i)}$ and $v_1 \cdot y^{(i)}$

## Simple Case: $p = 1$

- CCA projection matrices are vectors $u_1 \in \mathbb{R}^d$, $v_1 \in \mathbb{R}^{d'}$
- Project $x^{(i)}$ and $y^{(i)}$ to scalars $u_1 \cdot x^{(i)}$ and $v_1 \cdot y^{(i)}$
- What vectors does CCA find? Answer:

$$u_1, v_1 = \arg\max_{u,v} \ \text{Corr}\left(\{u \cdot x^{(i)}\}_{i=1}^{n}, \{v \cdot y^{(i)}\}_{i=1}^{n}\right)$$

## Finding the Next Projections

- After finding $u_1$ and $v_1$, what vectors $u_2$ and $v_2$ does CCA find? Answer:

$$u_2, v_2 = \arg\max_{u,v} \ \text{Corr}\left(\{u \cdot x^{(i)}\}_{i=1}^n, \{v \cdot y^{(i)}\}_{i=1}^n\right)$$

subject to the constraints

$$\text{Corr}\left(\{u_2 \cdot x^{(i)}\}_{i=1}^n, \{u_1 \cdot x^{(i)}\}_{i=1}^n\right) = 0$$
$$\text{Corr}\left(\{v_2 \cdot y^{(i)}\}_{i=1}^n, \{v_1 \cdot y^{(i)}\}_{i=1}^n\right) = 0$$

## CCA as an Optimization Problem

- CCA finds for $j = 1 \dots p$ (each column of $A$ and $B$)

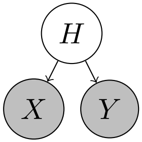
$$u_j, v_j = \arg\max_{u,v} \ \text{Corr}\left(\{u \cdot x^{(i)}\}_{i=1}^n, \{v \cdot y^{(i)}\}_{i=1}^n\right)$$

subject to the constraints

$$\text{Corr}\left(\{u_j \cdot x^{(i)}\}_{i=1}^n, \{u_k \cdot x^{(i)}\}_{i=1}^n\right) = 0$$
$$\text{Corr}\left(\{v_j \cdot y^{(i)}\}_{i=1}^n, \{v_k \cdot y^{(i)}\}_{i=1}^n\right) = 0$$

for $k < j$

## Guarantees for CCA



- Assume data is generated from a Naive Bayes model
- Latent-variable $H$ is of dimension $k$, variables $X$ and $Y$ are of dimension $d$ and $d'$ (typically $k \ll d$ and $k \ll d'$)
- Use CCA to project $X$ and $Y$ down to $k$ dimensions (needs $(x, y)$ pairs only!)
- Theorem: the projected samples are as good as the original samples for prediction of $H$
  (Foster, Johnson, Kakade, Zhang, 2009)
- Because $k \ll d$ and $k \ll d'$ we can learn to predict $H$ with far fewer labeled examples

## Guarantees for CCA (continued)

Kakade and Foster, 2007 - cotraining-style setting:

- Assume that we have a regression problem: predict some value $z$ given two "views" $x$ and $y$
- Assumption: either view $x$ or $y$ is sufficient for prediction
- Use CCA to project $x$ and $y$ down to a low-dimensional space
- Theorem: if correlation coefficients drop off to zero quickly, we will need far fewer samples to learn when using the projected representation
- Very similar setting to cotraining, but:
  - No assumption of independence between the two views
  - CCA is an exact algorithm - no need for heuristics

## Summary of the Section

- SVD is an efficient optimization technique
  - Low-rank matrix approximation
- CCA derives a new representation of paired data that maximizes correlation
  - SVD as a subroutine
- Next: use of CCA in deriving vector representations of words (“eigenwords”)

## Overview

Basic concepts

Lexical representations

- Eigenwords found using the thin SVD between words and context

  capture distributional similarity
  contain POS and semantic information about words
  are useful features for supervised learning

Hidden Markov Models

Latent-variable PCFGs

Conclusion

## Uses of Spectral Methods in NLP

- Word sequence labeling
  - Part of Speech tagging (POS)
  - Named Entity Recognition (NER)
  - Word Sense Disambiguation (WSD)
  - Chunking, prepositional phrase attachment, ...
- Language modeling
  - What is the most likely next word given a sequence of words (or of sounds)?
  - What is the most likely parse given a sequence of words?

## Uses of Spectral Methods in NLP

- Word sequence labeling: **semi-supervised learning**
  - Use CCA to learn vector representation of words (*eigenwords*) on a large unlabeled corpus.
  - Eigenwords map from words to vectors, which are used as features for supervised learning.
- Language modeling: **spectral estimation of probabilistic models**
  - Use eigenwords to reduce the dimensionality of generative models (HMMs,...)
  - Use those models to compute the probability of an observed word sequence

## The Eigenword Matrix $U$

- $U$ contains the singular vectors from the thin SVD of the bigram count matrix

| | ate | cheese | ham | I | You |
|---|---|---|---|---|---|
| ate | 0 | 1 | 1 | 0 | 0 |
| cheese | 0 | 0 | 0 | 0 | 0 |
| ham | 0 | 0 | 0 | 0 | 0 |
| I | 1 | 0 | 0 | 0 | 0 |
| You | 2 | 0 | 0 | 0 | 0 |

**I ate ham**
**You ate cheese**
**You ate**

## The Eigenword Matrix *U*

- *U* contains the singular vectors from the thin SVD of the bigram matrix ($w_{t-1} * w_t$) analogous to LSA, but uses context instead of documents
  - Context can be multiple neighboring words (we often use the words before and after the target)
  - Context can be neighbors in a parse tree
  - Eigenwords can also be computed using the CCA between words and their contexts
- Words close in the transformed space are distributionally, semantically and syntactically similar
- We will later use $U$ in HMMs and parse trees to project words to low dimensional vectors.

## Two Kinds of Spectral Models

- Context oblivious (*eigenwords*)
  - learn a vector representation of each word *type* based on its *average* context
- Context sensitive (*eigentokens* or *state*)
  - estimate a vector representation of each word *token* based on its *particular* context using an HMM or parse tree

## Eigenwords in Practice

- Work well with corpora of 100 million words
- We often use trigrams from the Google n-gram collection
- We generally use 30-50 dimensions
- Compute using fast randomized SVD methods

## How Big Should Eigenwords Be?

- A 40-D cube has $2^{40}$ (about a trillion) vertices.
- More precisely, in a 40-D space about $1.5^{40} \sim$ 11 million vectors can all be approximately orthogonal.
- So 40 dimensions gives *plenty* of space for a vocabulary of a million words

## Fast SVD: Basic Method

| | |
|---|---|
| problem | Find a low rank approximation to a $n \times m$ matrix $M$. |
| solution | Find an $n \times k$ matrix $A$ such that $M \approx AA^\top M$ |

## Fast SVD: Basic Method

problem Find a low rank approximation to a $n \times m$ matrix $M$.

solution Find an $n \times k$ matrix $A$ such that $M \approx AA^\top M$

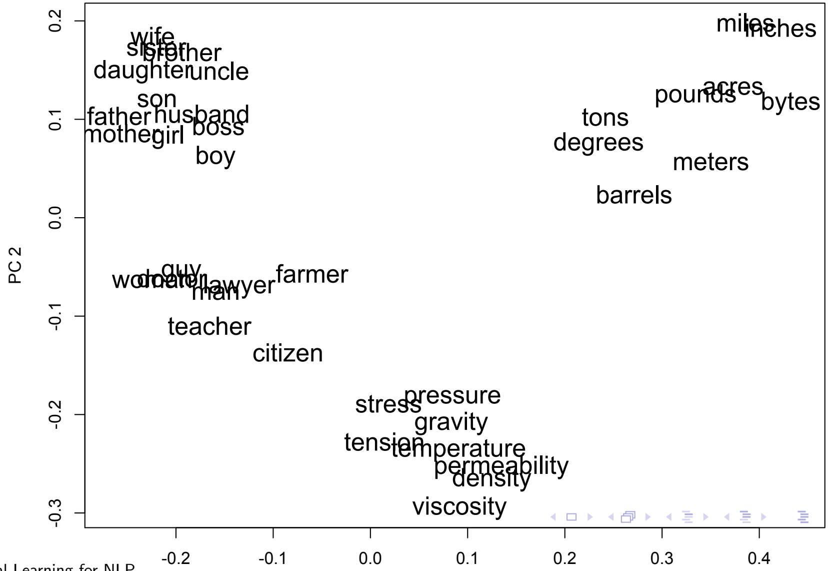
Construction $A$ is constructed by:

1. create a random $m \times k$ matrix $\Omega$ (iid normals)
2. compute $M\Omega$
3. Compute thin SVD of result: $UDV^\top = M\Omega$
4. $A = U$

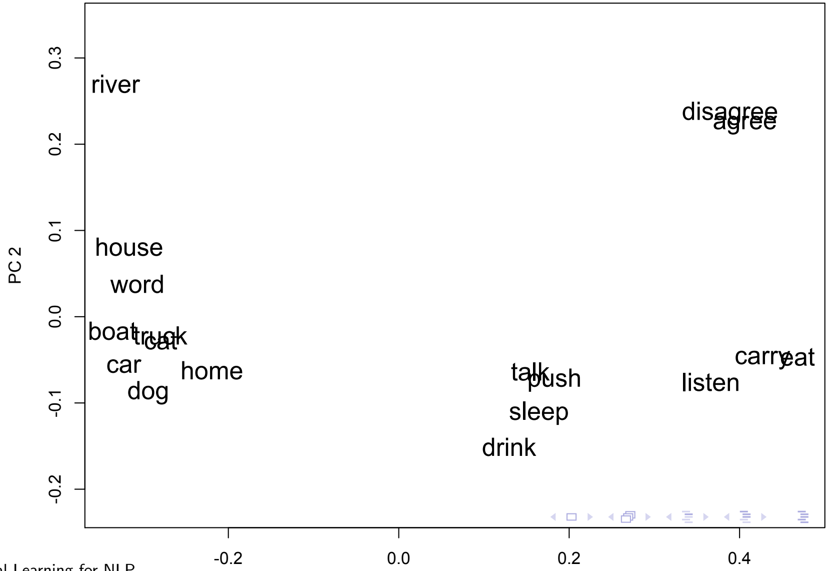
better: iterate a couple times

*"Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions" by N. Halko, P. G. Martinsson, and J. A. Tropp.*

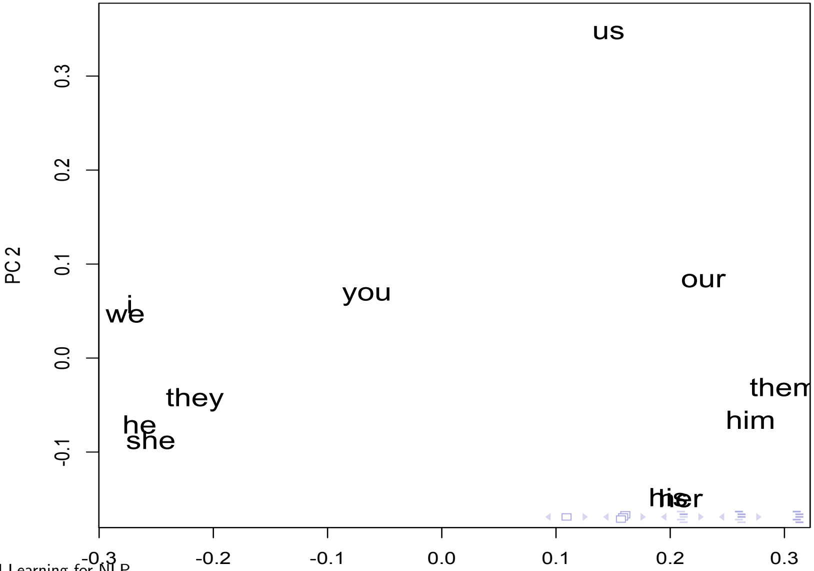
# Eigenwords for 'Similar' Words are Close



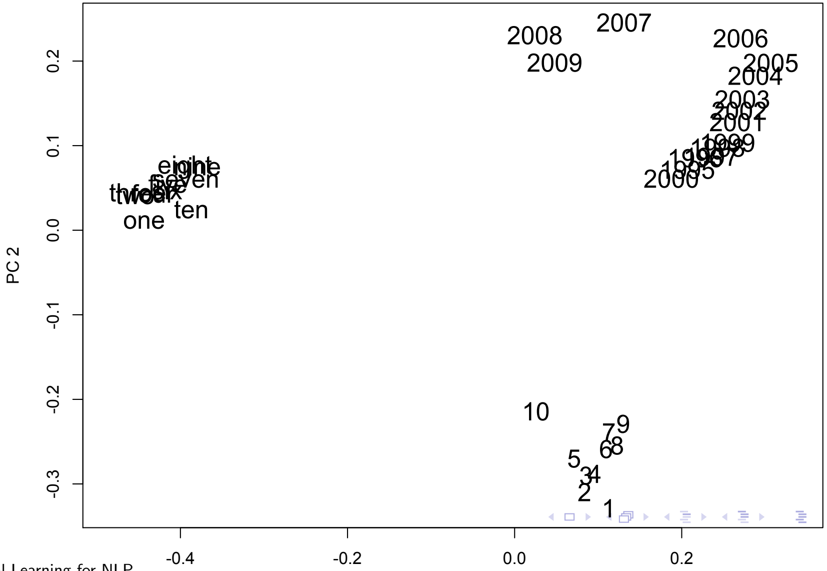
# Eigenwords Capture Part of Speech



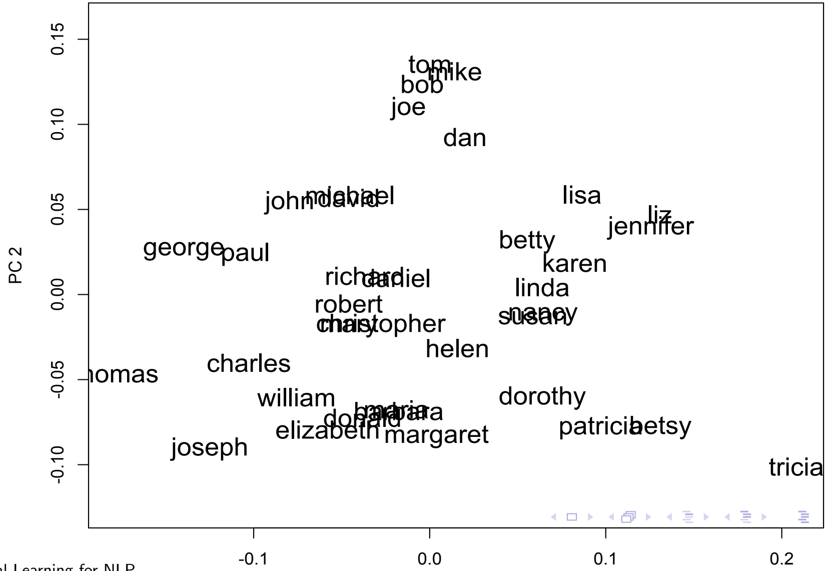
## Eigenwords: Pronouns



# Eigenwords: Numbers



# Eigenwords: Names



## CCA has Nice Properties for Computing Eigenwords

- When computing the SVD of a $word \times context$ matrix (as above) we need to decide how to scale the counts
- Using raw counts gives more emphasis to common words
- Better: rescale
  - Divide each row by the square root of the total count of the word in that row
  - Rescale the columns to account for the redundancy
- CCA between words and their contexts does this automatically and optimally
  - CCA 'whitens' the word-context covariance matrix

## Semi-supervised Learning Problems

- Sequence labeleing (Named Entity Recognition, POS, WSD...)
  - $X$ = target word
  - $Z$ = context of the target word
  - label = person / place / organization ...
- Topic identification
  - $X$ = words in title
  - $Z$ = words in abstract
  - label = topic category
- Speaker identification:
  - $X$ = video
  - $Z$ = audio
  - label = which character is speaking

## Semi-supervised Learning using CCA

- Find CCA between $X$ and $Z$
  - Recall: CCA finds projection matrices $A$ and $B$ such that

$$\underbrace{\underline{x}}_{k\times 1} = \underbrace{A^\top}_{k\times d} \underbrace{x}_{d\times 1} \qquad \underbrace{\underline{z}}_{k\times 1} = \underbrace{B^\top}_{k\times d'} \underbrace{z}_{d'\times 1}$$

- Project $X$ and $Z$ to estimate hidden state: $(\underline{x}, \underline{z})$
  - Note: if $x$ is the word and $z$ is its context, then $A$ is the matrix of eigenwords, $\underline{x}$ is the (context oblivious) eigenword corresponding to work $x$, and $\underline{z}$ gives a context-sensitive "eigentoken"
- Use supervised learning to predict label from hidden state
  - and from hidden state of neighboring words

## Theory: CCA has Nice Properties

- If one uses CCA to map from target word and context (two views, $X$ and $Z$) to reduced dimension hidden state and then uses that hidden state as features in a linear regression to predict a $y$, then we have provably almost as good a fit in the reduced dimsion (e.g. 40) as in the original dimension (e.g. million word vocabulary).
- In contrast, Principal Components Regression (PCR: regression based on PCA, which does not "whiten" the covariance matrix) can miss all the signal

[Foster and Kakade, '06]

## Semi-supervised Results

- Find spectral features on unlabeled data
  - RCV-1 corpus: Newswire
  - 63 million tokens in 3.3 million sentences.
  - Vocabulary size: 300k
  - Size of embeddings: k = 50
- Use in discriminative model
  - CRF for NER
  - Averaged perceptron for chunking
- Compare against state-of-the-art embeddings
  - C&W, HLBL, Brown, ASO and Semi-Sup CRF
  - Baseline features based on identity of word and its neighbors
- Benefit
  - Named Entity Recognition (NER): 8% error reduction
  - Chunking: 29% error reduction
  - Add spectral features to discriminative parser: 2.6% error reduction

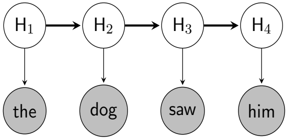
## Section Summary

- Eigenwords found using thin SVD between words and context
  - capture distributional similarity
  - contain POS and semantic information about words
  - perform competitively to a wide range of other embeddings
  - CCA version provides provable guarantees when used as features in supervised learning
- Next: eigenwords form the basis for fast estimation of HMMs and parse trees

## A Spectral Learning Algorithm for HMMs

- Algorithm due to Hsu, Kakade and Zhang (COLT 2009; JCSS 2012)
- Algorithm relies on singular value decomposition followed by very simple matrix operations
- Close connections to CCA
- Under assumptions on singular values arising from the model, has PAC-learning style guarantees (contrast with EM, which has problems with local optima)
- It is a *very* different algorithm from EM

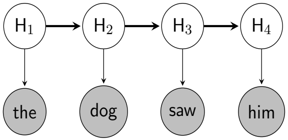
## Hidden Markov Models (HMMs)



$$p(\underbrace{\text{the dog saw him}}_{x_1 \ldots x_4}, \underbrace{\color{red}{1\ 2\ 1\ 3}}_{h_1 \ldots h_4})$$

$$= \quad \pi(1) \times t(2|1) \times t(1|2) \times t(3|1)$$

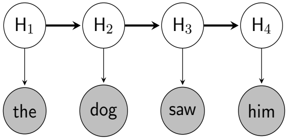
## Hidden Markov Models (HMMs)



$$p(\underbrace{\text{the dog saw him}}_{x_1 \ldots x_4}, \underbrace{\color{red}{1\ 2\ 1\ 3}}_{h_1 \ldots h_4})$$

$$\begin{aligned} = \quad & \pi(1) \times t(2|1) \times t(1|2) \times t(3|1) \\ & \times o(\text{the}|1) \times o(\text{dog}|2) \times o(\text{saw}|1) \times o(\text{him}|3) \end{aligned}$$

## Hidden Markov Models (HMMs)

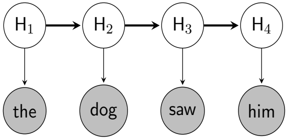


$$p(\underbrace{\text{the dog saw him}}_{x_1 \ldots x_4}, \underbrace{\color{red}{1\ 2\ 1\ 3}}_{h_1 \ldots h_4})$$

$$\begin{aligned} = \quad & \pi(1) \times t(2|1) \times t(1|2) \times t(3|1) \\ & \times o(\text{the}|1) \times o(\text{dog}|2) \times o(\text{saw}|1) \times o(\text{him}|3) \end{aligned}$$

- Initial parameters: $\pi(h)$ for each latent state $h$
- Transition parameters: $t(h'|h)$ for each pair of states $h'$, $h$
- Observation parameters: $o(x|h)$ for each state $h$, obs. $x$

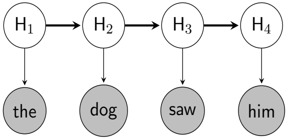
## Hidden Markov Models (HMMs)



Throughout this section:

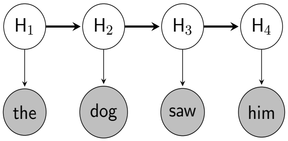
- **We use $m$ to refer to the number of hidden states**
- **We use $n$ to refer to the number of possible words (observations)**
- Typically, $m \ll n$ (e.g., $m = 20$, $n = 50,000$)

# HMMs: the forward algorithm



$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

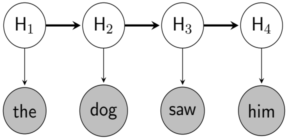
# HMMs: the forward algorithm



$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

## HMMs: the forward algorithm

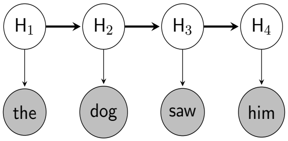


$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

$$f_h^0 = \pi(h)$$

## HMMs: the forward algorithm

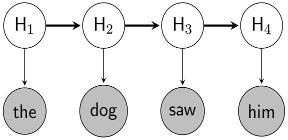


$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

$$f_h^0 = \pi(h) \quad f_h^1 = \sum_{h'} t(h|h')o(\text{the}|h')f_{h'}^0$$

## HMMs: the forward algorithm



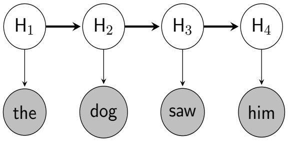
$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

$$f_h^0 = \pi(h) \quad f_h^1 = \sum_{h'} t(h|h')o(\text{the}|h')f_{h'}^0$$

$$f_h^2 = \sum_{h'} t(h|h')o(\text{dog}|h')f_{h'}^1$$

## HMMs: the forward algorithm



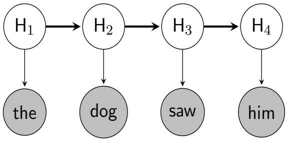
$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

$$f_h^0 = \pi(h) \quad f_h^1 = \sum_{h'} t(h|h')o(\text{the}|h')f_{h'}^0$$

$$f_h^2 = \sum_{h'} t(h|h')o(\text{dog}|h')f_{h'}^1 \quad f_h^3 = \sum_{h'} t(h|h')o(\text{saw}|h')f_{h'}^2$$

## HMMs: the forward algorithm



$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

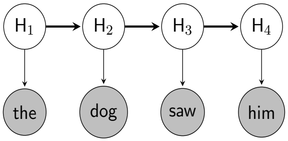
The forward algorithm:

$$f^0_h = \pi(h) \quad f^1_h = \sum_{h'} t(h|h')o(\text{the}|h')f^0_{h'}$$

$$f^2_h = \sum_{h'} t(h|h')o(\text{dog}|h')f^1_{h'} \quad f^3_h = \sum_{h'} t(h|h')o(\text{saw}|h')f^2_{h'}$$

$$f^4_h = \sum_{h'} t(h|h')o(\text{him}|h')f^3_{h'}$$

## HMMs: the forward algorithm



$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

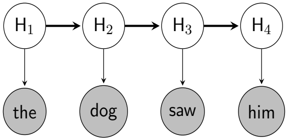
The forward algorithm:

$$f_h^0 = \pi(h) \quad f_h^1 = \sum_{h'} t(h|h')o(\text{the}|h')f_{h'}^0$$

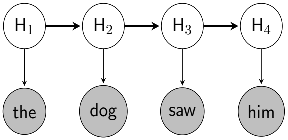
$$f_h^2 = \sum_{h'} t(h|h')o(\text{dog}|h')f_{h'}^1 \quad f_h^3 = \sum_{h'} t(h|h')o(\text{saw}|h')f_{h'}^2$$

$$f_h^4 = \sum_{h'} t(h|h')o(\text{him}|h')f_{h'}^3 \quad p(\ldots) = \sum_h f_h^4$$

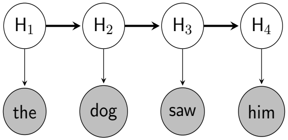
## HMMs: the forward algorithm in matrix form



## HMMs: the forward algorithm in matrix form



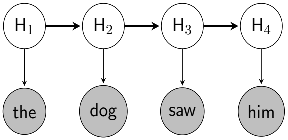
## HMMs: the forward algorithm in matrix form



- For each word $x$, define the matrix $A_x \in \mathbb{R}^{m \times m}$ as

$$[A_x]_{h',h} = t(h'|h)o(x|h)$$

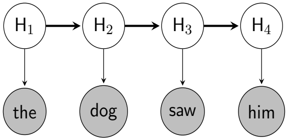
## HMMs: the forward algorithm in matrix form



- For each word $x$, define the matrix $A_x \in \mathbb{R}^{m \times m}$ as

$$[A_x]_{h',h} = t(h'|h)o(x|h) \quad \text{e.g., } [A_{\text{the}}]_{h',h} = t(h'|h)o(\text{the}|h)$$

## HMMs: the forward algorithm in matrix form

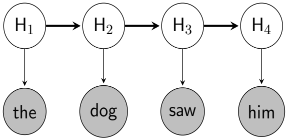


- For each word $x$, define the matrix $A_x \in \mathbb{R}^{m \times m}$ as

$$[A_x]_{h',h} = t(h'|h)o(x|h) \quad \text{e.g., } [A_{\text{the}}]_{h',h} = t(h'|h)o(\text{the}|h)$$

- Define $\pi$ as vector with elements $\pi_h$, $1$ as vector of all ones

## HMMs: the forward algorithm in matrix form



- For each word $x$, define the matrix $A_x \in \mathbb{R}^{m \times m}$ as

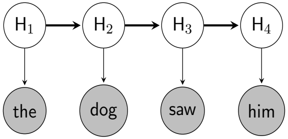
$$[A_x]_{h',h} = t(h'|h)o(x|h) \quad \text{e.g., } [A_{\text{the}}]_{h',h} = t(h'|h)o(\text{the}|h)$$

- Define $\pi$ as vector with elements $\pi_h$, $1$ as vector of all ones
- Then

$$p(\text{the dog saw him}) = 1^\top \times A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} \times \pi$$

Forward algorithm through matrix multiplication!

## The Spectral Algorithm: definitions



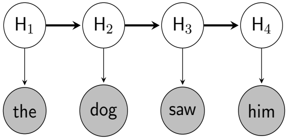
Define the following matrix $P_{2,1} \in \mathbb{R}^{n \times n}$:

$$[P_{2,1}]_{i,j} = \mathbf{P}(X_2 = i, X_1 = j)$$

Easy to derive an estimate:

$$[\hat{P}_{2,1}]_{i,j} = \frac{\text{Count}(X_2 = i, X_1 = j)}{N}$$

## The Spectral Algorithm: definitions



For each word $x$, define the following matrix $P_{3,x,1} \in \mathbb{R}^{n \times n}$:

$$[P_{3,x,1}]_{i,j} = \mathbf{P}(X_3 = i, X_2 = x, X_1 = j)$$

Easy to derive an estimate, e.g.,:

$$[\hat{P}_{3,\text{dog},1}]_{i,j} = \frac{\text{Count}(X_3 = i, X_2 = \text{dog}, X_1 = j)}{N}$$

## Main Result Underlying the Spectral Algorithm

- Define the following matrix $P_{2,1} \in \mathbb{R}^{n \times n}$:

$$[P_{2,1}]_{i,j} = \mathbf{P}(X_2 = i, X_1 = j)$$

- For each word $x$, define the following matrix $P_{3,x,1} \in \mathbb{R}^{n \times n}$:

$$[P_{3,x,1}]_{i,j} = \mathbf{P}(X_3 = i, X_2 = x, X_1 = j)$$

## Main Result Underlying the Spectral Algorithm

- Define the following matrix $P_{2,1} \in \mathbb{R}^{n \times n}$:

$$[P_{2,1}]_{i,j} = \mathbf{P}(X_2 = i, X_1 = j)$$

- For each word $x$, define the following matrix $P_{3,x,1} \in \mathbb{R}^{n \times n}$:

$$[P_{3,x,1}]_{i,j} = \mathbf{P}(X_3 = i, X_2 = x, X_1 = j)$$

- SVD$(P_{2,1}) \Rightarrow U \in \mathbb{R}^{n \times m}, \Sigma \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{n \times m}$

## Main Result Underlying the Spectral Algorithm

- Define the following matrix $P_{2,1} \in \mathbb{R}^{n \times n}$:

$$[P_{2,1}]_{i,j} = \mathbf{P}(X_2 = i, X_1 = j)$$

- For each word $x$, define the following matrix $P_{3,x,1} \in \mathbb{R}^{n \times n}$:

$$[P_{3,x,1}]_{i,j} = \mathbf{P}(X_3 = i, X_2 = x, X_1 = j)$$

- SVD$(P_{2,1}) \Rightarrow U \in \mathbb{R}^{n \times m}, \Sigma \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{n \times m}$

- Definition:

$$B_x = \underbrace{U^\top \times P_{3,x,1} \times V}_{m \times m} \times \underbrace{\Sigma^{-1}}_{m \times m}$$

## Main Result Underlying the Spectral Algorithm

- Define the following matrix $P_{2,1} \in \mathbb{R}^{n \times n}$:
$$[P_{2,1}]_{i,j} = \mathbf{P}(X_2 = i, X_1 = j)$$
- For each word $x$, define the following matrix $P_{3,x,1} \in \mathbb{R}^{n \times n}$:
$$[P_{3,x,1}]_{i,j} = \mathbf{P}(X_3 = i, X_2 = x, X_1 = j)$$
- $\text{SVD}(P_{2,1}) \Rightarrow U \in \mathbb{R}^{n \times m}, \Sigma \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{n \times m}$
- Definition:
$$B_x = \underbrace{U^\top \times P_{3,x,1} \times V}_{m \times m} \times \underbrace{\Sigma^{-1}}_{m \times m}$$
- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
**Forward algorithm through matrix multiplication!**

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
  $$B_x = G A_x G^{-1}$$
  where $G \in \mathbb{R}^{m \times m}$ is invertible
- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
  **Forward algorithm through matrix multiplication!**
- Now note that
  $$B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}}$$

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
**Forward algorithm through matrix multiplication!**

- Now note that
$$\begin{aligned} & B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \\ = \quad & G A_{\text{him}} G^{-1} \times G A_{\text{saw}} G^{-1} \times G A_{\text{dog}} G^{-1} \times G A_{\text{the}} G^{-1} \end{aligned}$$

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
**Forward algorithm through matrix multiplication!**

- Now note that
$$\begin{aligned} & B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \\ = \quad & G A_{\text{him}} G^{-1} \times G A_{\text{saw}} G^{-1} \times G A_{\text{dog}} G^{-1} \times G A_{\text{the}} G^{-1} \\ = \quad & G A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} G^{-1} \end{aligned}$$
**The $G$'s cancel!!**

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
**Forward algorithm through matrix multiplication!**

- Now note that
$$\begin{aligned} & B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \\ = \; & G A_{\text{him}} G^{-1} \times G A_{\text{saw}} G^{-1} \times G A_{\text{dog}} G^{-1} \times G A_{\text{the}} G^{-1} \\ = \; & G A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} G^{-1} \end{aligned}$$
**The $G$'s cancel!!**

- Follows that if we have $b^\infty = 1^\top G^{-1}$ and $b^0 = G\pi$ then

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
**Forward algorithm through matrix multiplication!**

- Now note that
$$\begin{aligned} & B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \\ = \quad & G A_{\text{him}} G^{-1} \times G A_{\text{saw}} G^{-1} \times G A_{\text{dog}} G^{-1} \times G A_{\text{the}} G^{-1} \\ = \quad & G A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} G^{-1} \end{aligned}$$
**The $G$'s cancel!!**

- Follows that if we have $b^\infty = 1^\top G^{-1}$ and $b^0 = G\pi$ then
$$b^\infty \times B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \times b^0$$

## Why does this matter?

- Theorem: if $P_{2,1}$ is of rank $m$, then
$$B_x = G A_x G^{-1}$$
where $G \in \mathbb{R}^{m \times m}$ is invertible

- Recall $p(\text{the dog saw him}) = 1^\top A_{\text{him}} A_{\text{saw}} A_{\text{dog}} A_{\text{the}} \pi$.
**Forward algorithm through matrix multiplication!**

- Now note that
$$\begin{aligned} & B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \\ = \; & G A_{\text{him}} G^{-1} \times G A_{\text{saw}} G^{-1} \times G A_{\text{dog}} G^{-1} \times G A_{\text{the}} G^{-1} \\ = \; & G A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} G^{-1} \end{aligned}$$
**The $G$'s cancel!!**

- Follows that if we have $b^\infty = 1^\top G^{-1}$ and $b^0 = G\pi$ then
$$\begin{aligned} & b^\infty \times B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \times b^0 \\ = \; & 1^\top \times A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} \times \pi \end{aligned}$$

## The Spectral Learning Algorithm

1. Derive estimates

$$[\hat{P}_{2,1}]_{i,j} = \frac{\text{Count}(X_2 = i, X_1 = j)}{N}$$

For all words $x$,

$$[\hat{P}_{3,x,1}]_{i,j} = \frac{\text{Count}(X_3 = i, X_2 = x, X_1 = j)}{N}$$

## The Spectral Learning Algorithm

1. Derive estimates

$$[\hat{P}_{2,1}]_{i,j} = \frac{\text{Count}(X_2 = i, X_1 = j)}{N}$$

   For all words $x$,

$$[\hat{P}_{3,x,1}]_{i,j} = \frac{\text{Count}(X_3 = i, X_2 = x, X_1 = j)}{N}$$

2. SVD$(\hat{P}_{2,1}) \Rightarrow U \in \mathbb{R}^{n\times m}, \Sigma \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times m}$

## The Spectral Learning Algorithm

1. Derive estimates

$$[\hat{P}_{2,1}]_{i,j} = \frac{\text{Count}(X_2 = i, X_1 = j)}{N}$$

   For all words $x$,

$$[\hat{P}_{3,x,1}]_{i,j} = \frac{\text{Count}(X_3 = i, X_2 = x, X_1 = j)}{N}$$

2. SVD$(\hat{P}_{2,1}) \Rightarrow U \in \mathbb{R}^{n \times m}, \Sigma \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{n \times m}$
3. For all words $x$, define $B_x = \underbrace{U^\top \times \hat{P}_{3,x,1} \times V}_{m \times m} \times \underbrace{\Sigma^{-1}}_{m \times m}$.

   (similar definitions for $b^0$, $b^\infty$, details omitted)

## The Spectral Learning Algorithm

1. Derive estimates

$$[\hat{P}_{2,1}]_{i,j} = \frac{\text{Count}(X_2 = i, X_1 = j)}{N}$$

   For all words $x$,

$$[\hat{P}_{3,x,1}]_{i,j} = \frac{\text{Count}(X_3 = i, X_2 = x, X_1 = j)}{N}$$

2. SVD$(\hat{P}_{2,1}) \Rightarrow U \in \mathbb{R}^{n\times m}, \Sigma \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times m}$
3. For all words $x$, define $B_x = \underbrace{U^\top \times \hat{P}_{3,x,1} \times V}_{m\times m} \times \underbrace{\Sigma^{-1}}_{m\times m}$.

   (similar definitions for $b^0$, $b^\infty$, details omitted)
4. For a new sentence $x_1 \dots x_n$, can calculate its probability, e.g.,

$$\begin{aligned} & \hat{p}(\text{the dog saw him}) \\ = \ & b^\infty \times B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \times b^0 \end{aligned}$$

## Guarantees

- Throughout the algorithm we've used estimates $\hat{P}_{2,1}$ and $\hat{P}_{3,x,1}$ in place of $P_{2,1}$ and $P_{3,x,1}$
- If $\hat{P}_{2,1} = P_{2,1}$ and $\hat{P}_{3,x,1} = P_{3,x,1}$ then the method is **exact**. **But** we will always have estimation errors
- A PAC-Style Theorem: Fix some length $T$. To have

$$\underbrace{\sum_{x_1 \ldots x_T} |p(x_1 \ldots x_T) - \hat{p}(x_1 \ldots x_T)|}_{L_1 \text{ distance between } p \text{ and } \hat{p}} \leq \epsilon$$

with probability at least $1 - \delta$, then number of samples required is polynomial in

$$n, m, 1/\epsilon, 1/\delta, 1/\sigma, T$$

where $\sigma$ is $m$'th largest singular value of $P_{2,1}$

## Intuition behind the Theorem

- Define

$$||\hat{A} - A||_2 = \sqrt{\sum_{j,k} (\hat{A}_{j,k} - A_{j,k})^2}$$

- With $N$ samples, with probability at least $1 - \delta$

$$||\hat{P}_{2,1} - P_{2,1}||_2 \leq \epsilon$$

$$||\hat{P}_{3,x,1} - P_{3,x,1}||_2 \leq \epsilon$$

  where

$$\epsilon = \sqrt{\frac{1}{N} \log \frac{1}{\delta}} + \sqrt{\frac{1}{N}}$$

- Then need to carefully bound how the error $\epsilon$ propagates through the SVD step, the various matrix multiplications, etc etc. The "rate" at which $\epsilon$ propagates depends on $T$, $m$, $n$, $1/\sigma$

## Summary

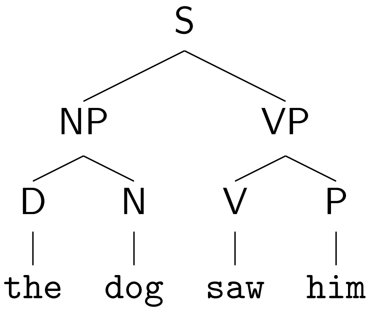
- The problem solved by EM: estimate HMM parameters $\pi(h)$, $t(h'|h)$, $o(x|h)$ from observation sequences $x_1 \dots x_n$
- The spectral algorithm:
  - Calculate estimates $\hat{P}_{2,1}$ (bigram counts) and $\hat{P}_{3,x,1}$ (trigram counts)
  - Run an SVD on $\hat{P}_{2,1}$
  - Calculate parameter estimates using simple matrix operations
  - Guarantee: we recover the parameters up to linear transforms that cancel

## Overview

## Probabilistic Context-free Grammars

- Used for natural language parsing and other structured models
- Induce probability distributions over phrase-structure trees

## The Probability of a Tree



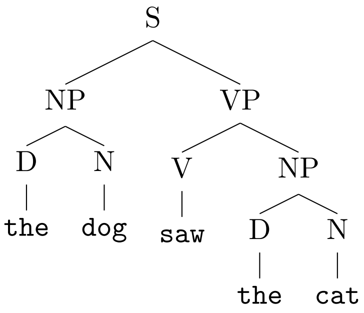
$$\begin{aligned} & p(\text{tree}) \\ &= \pi(\mathsf{S}) \times \\ &\quad t(\mathsf{S} \rightarrow \mathsf{NP}\ \ \mathsf{VP}|\mathsf{S}) \times \\ &\quad t(\mathsf{NP} \rightarrow \mathsf{D}\ \ \mathsf{N}|\mathsf{NP}) \times \\ &\quad t(\mathsf{VP} \rightarrow \mathsf{V}\ \ \mathsf{P}|\mathsf{VP}) \times \\ &\quad q(\mathsf{D} \rightarrow \mathtt{the}|\mathsf{D}) \times \\ &\quad q(\mathsf{N} \rightarrow \mathtt{dog}|\mathsf{N}) \times \\ &\quad q(\mathsf{V} \rightarrow \mathtt{saw}|\mathsf{V}) \times \\ &\quad q(\mathsf{P} \rightarrow \mathtt{him}|\mathsf{P}) \end{aligned}$$

We assume PCFGs in Chomsky normal form

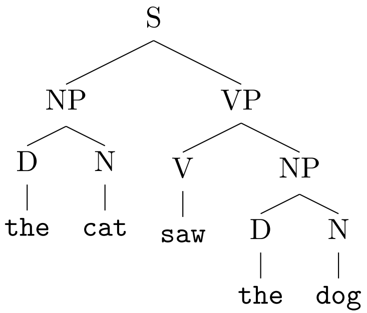
## PCFGs - Advantage

"Context-freeness" leads to **generalization** ("NP" - noun phrase):

Seen in data:



Unseen in data (grammatical):

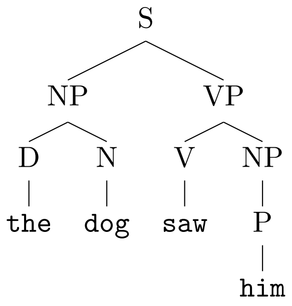


An NP subtree can be combined anywhere an NP is expected

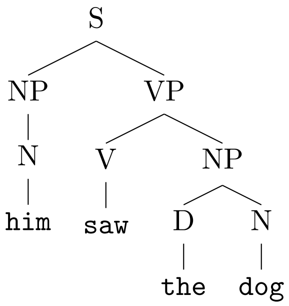
## PCFGs - Disadvantage

"Context-freeness" can lead to over-generalization:

Seen in data:



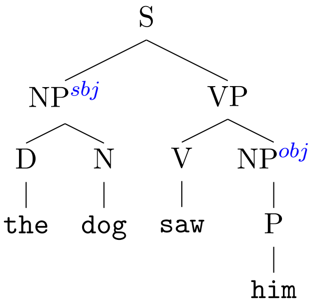
Unseen in data (ungrammatical):



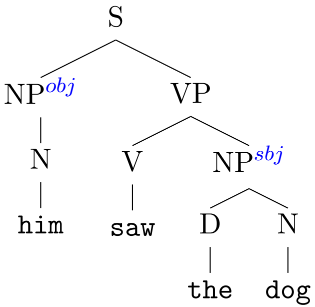
## PCFGs - a Fix

Adding context to the nonterminals fixes that:

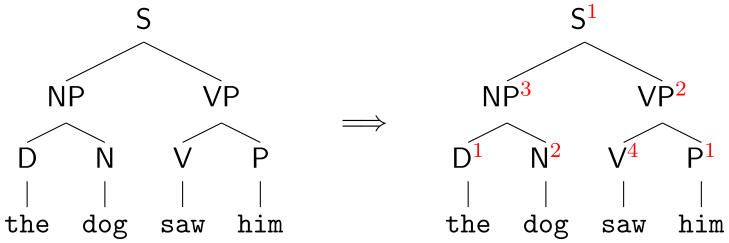
Seen in data:



Low likelihood:

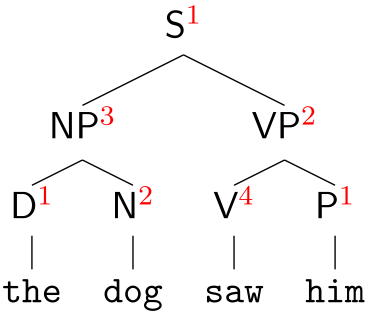


## Idea: Latent-Variable PCFGs (Matsuzaki et al., 2005; Petrov et al., 2006)



The **latent states** for each node are never observed

## The Probability of a Tree



$$
\begin{aligned}
& p(\text{tree}, 1\ 3\ 1\ 2\ 2\ 4\ 1) \\
& = \pi(\mathsf{S}^1) \times \\
& \quad t(\mathsf{S}^1 \rightarrow \mathsf{NP}^3\ \ \mathsf{VP}^2 | \mathsf{S}^1) \times \\
& \quad t(\mathsf{NP}^3 \rightarrow \mathsf{D}^1\ \ \mathsf{N}^2 | \mathsf{NP}^3) \times \\
& \quad t(\mathsf{VP}^2 \rightarrow \mathsf{V}^4\ \ \mathsf{P}^1 | \mathsf{VP}^2) \times \\
& \quad q(\mathsf{D}^1 \rightarrow \texttt{the} | \mathsf{D}^1) \times \\
& \quad q(\mathsf{N}^2 \rightarrow \texttt{dog} | \mathsf{N}^2) \times \\
& \quad q(\mathsf{V}^4 \rightarrow \texttt{saw} | \mathsf{V}^4) \times \\
& \quad q(\mathsf{P}^1 \rightarrow \texttt{him} | \mathsf{P}^1)
\end{aligned}
$$

$$p(\text{tree}) = \sum_{h_1 \ldots h_7} p(tree, h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7)$$

## Learning L-PCFGs

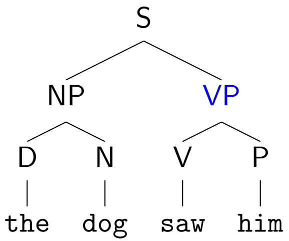
- Expectation-maximization (Matsuzaki et al., 2005)
- Split-merge techniques (Petrov et al., 2006)

Neither solves the issue of local maxima or statistical consistency

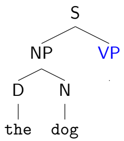
## Overview

## Inside and Outside Trees

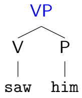
At node VP:



Outside tree $o =$



Inside tree $t =$

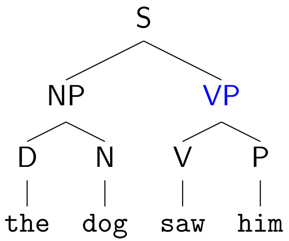


Conditionally independent given the label and the hidden state

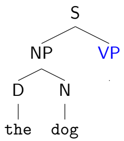
$$p(o, t|\text{VP}, h) = p(o|\text{VP}, h) \times p(t|\text{VP}, h)$$

## Inside and Outside Trees

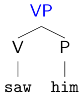
At node VP:



Outside tree $o =$



Inside tree $t =$



Conditionally independent given the label and the hidden state

$$p(o, t|\text{VP}, h) = p(o|\text{VP}, h) \times p(t|\text{VP}, h)$$

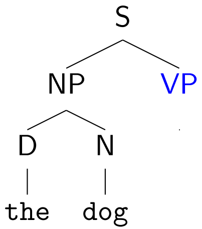
## Vector Representation of Inside and Outside Trees

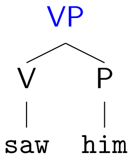
Assume functions $Z$ and $Y$:

$Z$ maps any outside tree to a vector of length $m$.

$Y$ maps any inside tree to a vector of length $m$.

Convention: $m$ is the number of hidden states under the L-PCFG.





Outside tree $o \Rightarrow$

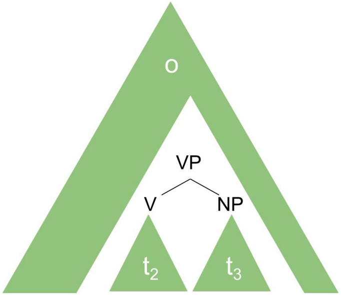
$Z(o) = [1, 0.4, -5.3, \ldots, 72] \in \mathbb{R}^m$

Inside tree $t \Rightarrow$

$Y(t) = [-3, 17, 2, \ldots, 3.5] \in \mathbb{R}^m$

## Parameter Estimation for Binary Rules

Take $M$ samples of nodes with rule VP $\rightarrow$ V NP.



At sample $i$

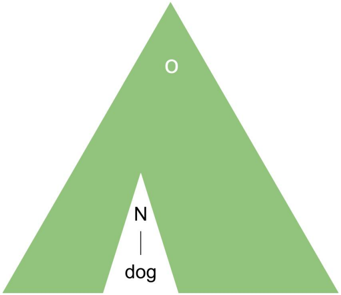
- $o^{(i)}$ = outside tree at VP
- $t_2^{(i)}$ = inside tree at V
- $t_3^{(i)}$ = inside tree at NP

$$\hat{t}(\text{VP}^{h_1} \rightarrow \text{V}^{h_2} \ \ \text{NP}^{h_3} | \text{VP}^{h_1})$$

$$= \frac{\textbf{count}(\text{VP} \rightarrow \text{V} \ \ \text{NP})}{\textbf{count}(\text{VP})} \times \frac{1}{M} \sum_{i=1}^{M} \left( Z_{h_1}(o^{(i)}) \times Y_{h_2}(t_2^{(i)}) \times Y_{h_3}(t_3^{(i)}) \right)$$

## Parameter Estimation for Unary Rules

Take $M$ samples of nodes with rule N $\rightarrow$ dog.



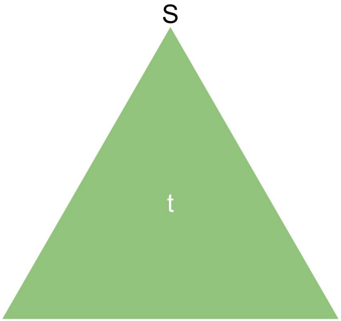
At sample $i$

- $o^{(i)}$ = outside tree at N

$$\hat{q}(\mathsf{N}^{h} \rightarrow \mathsf{dog}|\mathsf{N}^{h}) = \frac{\textbf{count}(\mathsf{N} \rightarrow \mathsf{dog})}{\textbf{count}(\mathsf{N})} \times \frac{1}{M} \sum_{i=1}^{M} Z_{h}(o^{(i)})$$

## Parameter Estimation for the Root

Take $M$ samples of the root S.



At sample $i$

- $t^{(i)}$ = inside tree at S

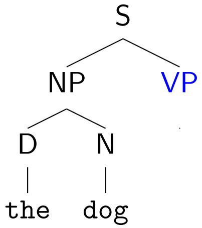
$$\hat{\pi}(\mathsf{S}^{\color{red}{h}}) = \frac{\textbf{count}(\text{root=S})}{\textbf{count}(\text{root})} \times \frac{1}{M} \sum_{i=1}^{M} Y_{\color{red}{h}}(t^{(i)})$$

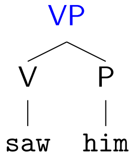
## Deriving $Z$ and $Y$

Design functions $\psi$ and $\phi$:

$\psi$ maps any outside tree to a vector of length $d'$

$\phi$ maps any inside tree to a vector of length $d$





Outside tree $o \Rightarrow$

$\psi(o) = [0, 1, 0, 0, \ldots, 0, 1] \in \mathbb{R}^{d'}$

Inside tree $t \Rightarrow$

$\phi(t) = [1, 0, 0, 0, \ldots, 1, 0] \in \mathbb{R}^{d}$

$Z$ and $Y$ will be reduced dimensional representations of $\psi$ and $\phi$.

## Reducing Dimensions via a Singular Value Decomposition

Have $M$ samples of a node with non-terminal $a$. At sample $i$, $o^{(i)}$ is the outside tree rooted at $a$ and $t^{(i)}$ is the inside tree rooted at $a$.

- Compute a matrix $\hat{\Omega}^a \in \mathbb{R}^{d \times d'}$ with entries

$$[\hat{\Omega}^a]_{j,k} = \frac{1}{M} \sum_{i=1}^{M} \phi_j(t^{(i)}) \psi_k(o^{(i)})$$

## Reducing Dimensions via a Singular Value Decomposition

Have $M$ samples of a node with non-terminal $a$. At sample $i$, $o^{(i)}$ is the outside tree rooted at $a$ and $t^{(i)}$ is the inside tree rooted at $a$.

- Compute a matrix $\hat{\Omega}^a \in \mathbb{R}^{d \times d'}$ with entries

$$[\hat{\Omega}^a]_{j,k} = \frac{1}{M} \sum_{i=1}^{M} \phi_j(t^{(i)}) \psi_k(o^{(i)})$$

- An SVD:

$$\underbrace{\hat{\Omega}^a}_{d \times d'} \approx \underbrace{U^a}_{d \times m} \underbrace{\Sigma^a}_{m \times m} \underbrace{(V^a)^T}_{m \times d'}$$

## Reducing Dimensions via a Singular Value Decomposition

Have $M$ samples of a node with non-terminal $a$. At sample $i$, $o^{(i)}$ is the outside tree rooted at $a$ and $t^{(i)}$ is the inside tree rooted at $a$.

- Compute a matrix $\hat{\Omega}^a \in \mathbb{R}^{d\times d'}$ with entries

$$[\hat{\Omega}^a]_{j,k} = \frac{1}{M}\sum_{i=1}^{M} \phi_j(t^{(i)})\psi_k(o^{(i)})$$

- An SVD:

$$\underbrace{\hat{\Omega}^a}_{d\times d'} \approx \underbrace{U^a}_{d\times m}\underbrace{\Sigma^a}_{m\times m}\underbrace{(V^a)^T}_{m\times d'}$$

- Projection:

$$Y(t^{(i)}) = \underbrace{(U^a)^T}_{m\times d}\underbrace{\phi(t^{(i)})}_{d\times 1} \in \mathbb{R}^m$$

$$Z(o^{(i)}) = \underbrace{(\Sigma^a)^{-1}}_{m\times m}\underbrace{(V^a)^T}_{m\times d'}\underbrace{\psi(o^{(i)})}_{d'\times 1} \in \mathbb{R}^m$$

## A Summary of the Algorithm

1. Design feature functions $\phi$ and $\psi$ for inside and outside trees.
2. Use SVD to compute vectors

   $Y(t) \in \mathbb{R}^m$ for inside trees

   $Z(o) \in \mathbb{R}^m$ for outside trees
3. Estimate the parameters $\hat{t}$, $\hat{q}$, and $\hat{\pi}$ from the training data.

## Overview

## Justification of the Algorithm: Roadmap

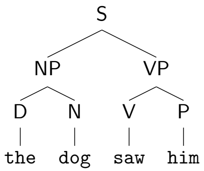
How do we marginalize latent states? Dynamic programming

Succinct tensor form of representing the DP algorithm

Estimation guarantees explained through the tensor form

How do we parse? Dynamic programming again

## Calculating Tree Probability with Dynamic Programming: Revisited



$$\hat{b}^1_h = \sum_{h_2,h_3} \hat{t}(\mathsf{NP}^h \rightarrow \mathsf{D}^{h_2}\ \ \mathsf{N}^{h_3} | \mathsf{NP}^h) \times \hat{q}(\mathsf{D}^{h_2} \rightarrow \texttt{the} | \mathsf{D}^{h_2}) \times \hat{q}(\mathsf{N}^{h_3} \rightarrow \texttt{dog} | \mathsf{N}^{h_3})$$

$$\hat{b}^2_h = \sum_{h_2,h_3} \hat{t}(\mathsf{VP}^h \rightarrow \mathsf{V}^{h_2}\ \ \mathsf{P}^{h_3} | \mathsf{VP}^h) \times \hat{q}(\mathsf{V}^{h_2} \rightarrow \texttt{saw} | \mathsf{V}^{h_2}) \times \hat{q}(\mathsf{P}^{h_3} \rightarrow \texttt{him} | \mathsf{P}^{h_3})$$

$$\hat{b}^3_h = \sum_{h_2,h_3} \hat{t}(\mathsf{S}^h \rightarrow \mathsf{NP}^{h_2}\ \ \mathsf{VP}^{h_3} | \mathsf{S}^h) \times \hat{b}^1_{h_2} \times \hat{b}^2_{h_3}$$

$$p(\mathsf{tree}) = \sum_h \hat{\pi}(S^h) \times \hat{b}^3_h$$

## Tensor Form of the Parameters

For each non-terminal $a$, define a vector $\pi^a \in \mathbb{R}^m$ with entries

$$[\pi^a]_{h} = \pi(a^{h})$$

For each rule $a \to x$, define a vector $q_{a\to x} \in \mathbb{R}^m$ with entries

$$[q_{a\to x}]_{h} = q_{a\to x}(a^{h} \to x|a^{h})$$

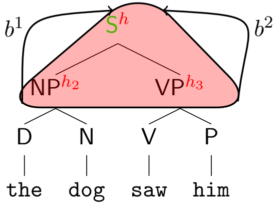
For each rule $a \to b\ c$, define a tensor $T^{a\to b\ c} \in \mathbb{R}^{m\times m\times m}$ with entries

$$[T^{a\to b\ c}]_{h_1,h_2,h_3} = t(a^{h_1} \to b^{h_2}\ c^{h_3}|a^{h_1})$$

# Tensor Formulation of Dynamic Programming

- The dynamic programming algorithm can be represented much more compactly based on basic tensor-matrix-vector products

Regular form:



$$b^3_h = \sum_{h_2,h_3} t(\mathsf{S}^h \to \mathsf{NP}^{h_2} \;\; \mathsf{VP}^{h_3} | \mathsf{S}^h) \times b^1_{h_2} \times b^2_{h_3}$$

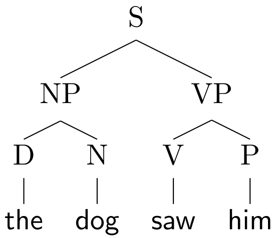
Equivalent tensor form:

$$b^3 = T^{\mathrm{S \to NP\ VP}}(b^1, b^2)$$

where $T^{\mathrm{S \to NP\ VP}} \in \mathbb{R}^{m \times m \times m}$ and

$$T^{\mathrm{S \to NP\ VP}}_{h,h_2,h_3} = t(\mathsf{S}^h \to \mathsf{NP}^{h_2} \;\; \mathsf{VP}^{h_3} | \mathsf{S}^h)$$

## Dynamic Programming in Tensor Form



$$T^{\mathrm{S}\rightarrow\mathrm{NP}\;\mathrm{VP}}(T^{\mathrm{NP}\rightarrow\mathrm{D}\;\mathrm{N}}(q_{\mathrm{D}\rightarrow\mathrm{the}}, q_{\mathrm{N}\rightarrow\mathrm{dog}}), T^{\mathrm{VP}\rightarrow\mathrm{V}\;\mathrm{P}}(q_{\mathrm{V}\rightarrow\mathrm{saw}}, q_{\mathrm{P}\rightarrow\mathrm{him}}))\;\pi^{\mathrm{S}}$$

$$|||$$

$$p(\textsf{tree}) = \sum_{h_1 \ldots h_7} p(tree, h_1\; h_2\; h_3\; h_4\; h_5\; h_6\; h_7)$$

## Thought Experiment

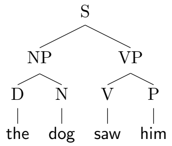
- We want the parameters (in tensor form)

$$\begin{aligned} \pi^a &\in \mathbb{R}^m \\ q_{a\to x} &\in \mathbb{R}^m \\ T^{a\to b\ c}(y_2, y_3) &\in \mathbb{R}^m \end{aligned}$$

- What if we had an invertible matrix $G^a \in \mathbb{R}^{m\times m}$ for every non-terminal $a$?
- And what if we had instead

$$\begin{aligned} c^a &= G^a \pi^a \\ c_{a\to x} &= q_{a\to x}(G^a)^{-1} \\ C^{a\to b\ c}(y_2, y_3) &= T^{a\to b\ c}(y_2 G^b, y_3 G^c)(G^a)^{-1} \end{aligned}$$

## Cancellation of the Linear Operators



$$C^{\mathrm{S}\to\mathrm{NP\,VP}}(C^{\mathrm{NP}\to\mathrm{D\,N}}(c_{\mathrm{D}\to\mathrm{the}}, c_{\mathrm{N}\to\mathrm{dog}}), C^{\mathrm{VP}\to\mathrm{V\,P}}(c_{\mathrm{V}\to\mathrm{saw}}, c_{\mathrm{P}\to\mathrm{him}}))\ c^{\mathrm{S}}$$

$$|||$$

$$T^{\mathrm{S}\to\mathrm{NP\,VP}}(T^{\mathrm{NP}\to\mathrm{D\,N}}(q_{\mathrm{D}\to\mathrm{the}}(G^{\mathrm{D}})^{-1}G^{\mathrm{D}}, q_{\mathrm{N}\to\mathrm{dog}}(G^{\mathrm{N}})^{-1}G^{\mathrm{N}})(G^{\mathrm{NP}})^{-1}G^{\mathrm{NP}}, T^{\mathrm{VP}\to\mathrm{V\,P}}(q_{\mathrm{V}\to\mathrm{saw}}(G^{\mathrm{V}})^{-1}G^{\mathrm{V}}, q_{\mathrm{P}\to\mathrm{him}}(G^{\mathrm{P}})^{-1}G^{\mathrm{P}})(G^{\mathrm{VP}})^{-1}G^{\mathrm{VP}})(G^{\mathrm{S}})^{-1}G^{\mathrm{S}}\pi^{\mathrm{S}}$$

$$|||$$

$$T^{\mathrm{S}\to\mathrm{NP\,VP}}(T^{\mathrm{NP}\to\mathrm{D\,N}}(q_{\mathrm{D}\to\mathrm{the}}, q_{\mathrm{N}\to\mathrm{dog}}), T^{\mathrm{VP}\to\mathrm{V\,P}}(q_{\mathrm{V}\to\mathrm{saw}}, q_{\mathrm{P}\to\mathrm{him}}))\ \pi^{\mathrm{S}}$$

$$|||$$

$$p(\text{tree}) = \sum_{h_1 \ldots h_7} p(tree, h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7)$$

## Estimation Guarantees

- Basic argument: If $\Omega^a$ has rank $m$, parameters $\hat{C}^{a \rightarrow b\ c}$, $\hat{c}_{a \rightarrow x}$, and $\hat{c}^a$ converge to

  $$\begin{aligned} C^{a \rightarrow b\ c}(y_2, y_3) &= T^{a \rightarrow b\ c}(y_2 G^b, y_3 G^c)(G^a)^{-1} \\ c_{a \rightarrow x} &= q_{a \rightarrow x}(G^a)^{-1} \\ c^a &= G^a \pi^a \end{aligned}$$

  for some $G^a$ that is invertible.
- $G^a$ are unknown, but they are there, canceling out perfectly

## Implications of Guarantees

- The dynamic programming algorithm calculates $\hat{p}(\text{tree})$
- As we have more data, $\hat{p}(\text{tree})$ converges to $p(\text{tree})$

But we are interested in parsing – trees are unobserved

## Cancellation of Linear Operators

Can compute any quantity that marginalizes out latent states

E.g.: the inside-outside algorithm can compute "marginals"

$\mu(a, i, j)$ : the probability that $a$ spans words $i$ through $j$

No latent states involved! They are marginalized out

They are used as auxiliary variables in the model

## Minimum Bayes Risk Decoding

Parsing algorithm:

- Find marginas $\mu(a, i, j)$ for each nonterminal $a$ and span $(i, j)$ in a sentence
- Compute using CKY the best tree $t$:

$$\arg\max_t \sum_{(a,i,j) \in t} \mu(a, i, j)$$

Minimum Bayes risk decoding (Goodman, 1996)

## Overview

## Results with EM (section 22 of Penn treebank)

| | |
|---|---|
| $m = 8$ | 86.87 |
| $m = 16$ | 88.32 |
| $m = 24$ | 88.35 |
| $m = 32$ | 88.56 |

Vanilla PCFG maximum likelihood estimation performance: 68.62%

We focus on $m = 32$

## Key Ingredients for Accurate Spectral Learning

Feature functions

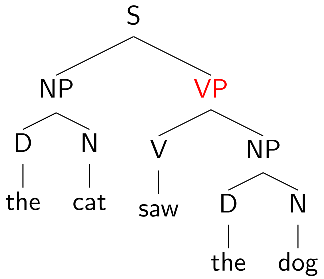
Handling negative marginals

Scaling of features

Smoothing

## Inside Features Used

Consider the VP node in the following tree:

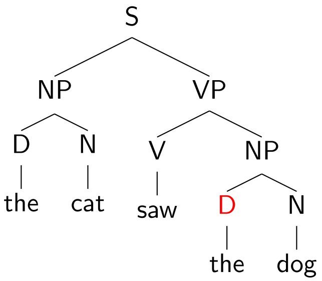


The inside features consist of:

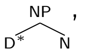
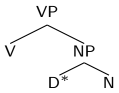
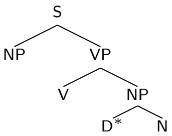
- The pairs `(VP, V)` and `(VP, NP)`
- The rule `VP` $\rightarrow$ `V NP`
- The tree fragment `(VP (V saw) NP)`
- The tree fragment `(VP V (NP D N))`
- The pair of head part-of-speech tag with VP: `(VP, V)`
- The width of the subtree spanned by VP: `(VP, 2)`

## Outside Features Used

Consider the D node in the following tree:



The outside features consist of:

- The fragments ,  and 
- The pair `(D, NP)` and triplet `(D, NP, VP)`
- The pair of head part-of-speech tag with D: `(D, N)`
- The widths of the spans left and right to D: `(D, 3)` and `(D, 1)`

## Accuracy (section 22 of the Penn treebank)

The accuracy out-of-the-box with these features is:

$$55.09\%$$

EM's accuracy: 88.56%

## Negative Marginals

Sampling error can lead to negative marginals

Signs of marginals are flipped

On certain sentences, this gives the world's worst parser:

$$t^* = \arg\max_t -\text{score}(t) = \arg\min_t \text{score}(t)$$

Taking the absolute value of the marginals fixes it

Likely to be caused by sampling error

## Accuracy (section 22 of the Penn treebank)

The accuracy with absolute-value marginals is:

$$80.23\%$$

EM's accuracy: $88.56\%$

## Scaling of Features by Inverse Variance

Features are mostly binary. Replace $\phi_i(t)$ by

$$\phi_i(t) \times \sqrt{\frac{1}{\mathrm{count}(i) + \kappa}}$$

where $\kappa = 5$

This is an approximation to replacing $\phi(t)$ by

$$(C)^{-1/2}\phi(t)$$

where $C = E[\phi\phi^\top]$

Closely related to canonical correlation analysis

## Accuracy (section 22 of the Penn treebank)

The accuracy with scaling is:

$$86.47\%$$

EM's accuracy: $88.56\%$

## Smoothing

Estimates required:

$$\hat{E}(\mathsf{VP}^{h_1} \rightarrow \mathsf{V}^{h_2} \ \ \mathsf{NP}^{h_3}|\mathsf{VP}^{h_1}) = \frac{1}{M}\sum_{i=1}^{M}\left(Z_{h_1}(o^{(i)}) \times Y_{h_2}(t_2^{(i)}) \times Y_{h_3}(t_3^{(i)})\right.$$

Smooth using "backed-off" estimates, e.g.:

$$\lambda\hat{E}(\mathsf{VP}^{h_1} \rightarrow \mathsf{V}^{h_2} \ \ \mathsf{NP}^{h_3}|\mathsf{VP}^{h_1}) + (1-\lambda)\hat{F}( \ \ \mathsf{VP}^{h_1} \rightarrow \mathsf{V}^{h_2} \ \ \mathsf{NP}^{h_3}|\mathsf{VP}^{h_1})$$

where

$$\hat{F}(\mathsf{VP}^{h_1} \rightarrow \mathsf{V}^{h_2} \ \ \mathsf{NP}^{h_3}|\mathsf{VP}^{h_1})$$
$$= \left(\frac{1}{M}\sum_{i=1}^{M}\left(Z_{h_1}(o^{(i)}) \times Y_{h_2}(t_2^{(i)})\right)\right) \times \left(\frac{1}{M}\sum_{i=1}^{M}Y_{h_3}(t_3^{(i)})\right)$$

## Accuracy (section 22 of the Penn treebank)

The accuracy with smoothing is:

$$88.82\%$$

EM's accuracy: 88.56%

## Final Results

Final results on the Penn treebank

| | section 22 | | section 23 | |
|---|---|---|---|---|
| | EM | spectral | EM | spectral |
| $m = 8$ | 86.87 | 85.60 | — | — |
| $m = 16$ | 88.32 | 87.77 | — | — |
| $m = 24$ | 88.35 | 88.53 | — | — |
| $m = 32$ | 88.56 | 88.82 | 87.76 | 88.05 |

## Simple Feature Functions

Use rule above (for outside) and rule below (for inside)

Corresponds to parent annotation and sibling annotation

Accuracy:

$$88.07\%$$

Accuracy of parent and sibling annotation: 82.59%

The spectral algorithm distills latent states

Avoids overfitting caused by Markovization

## Running Time

EM and the spectral algorithm are cubic in the number of latent states

But EM requires a few iterations

| $m$ | single EM iter. | EM best model | spectral algorithm total | SVD | $a \to b\,c$ | $a \to x$ |
|---|---|---|---|---|---|---|
| 8 | 6m | 3h | 3h32m | 36m | 1h34m | 10m |
| 16 | 52m | 26h6m | 5h19m | 34m | 3h13m | 19m |
| 24 | 3h7m | 93h36m | 7h15m | 36m | 4h54m | 28m |
| 32 | 9h21m | 187h12m | 9h52m | 35m | 7h16m | 41m |

SVD with sparse matrices is very efficient

## Related Work

Spectral algorithms have been used for parsing in other settings:

- Dependency parsing (Dhillon et al., 2012)
- Split head automaton grammars (Luque et al., 2012)
- Probabilistic grammars (Bailly et al., 2010)

## Summary

Presented spectral algorithms as a method for estimating latent-variable models

**Formal guarantees:**

- Statistical consistency
- No issue with local maxima

**Complexity:**

- Most time is spent on aggregating statistics
- Much faster than the alternative, expectation-maximization
- Singular value decomposition step is fast

**Widely applicable for latent-variable models:**

- Lexical representations
- HMMs, L-PCFGs (and R-HMMs)
- Topic modeling

## Addendum: Spectral Learning for Topic Modeling

## Spectral Topic Modeling: Bag-of-Words

- Bag-of-words model with $K$ topics and $d$ words
- Model parameters: for $i = 1 \dots K$,

$$\begin{aligned} w_i \in \mathbb{R} &: \text{probability of topic } i \\ \mu_i \in \mathbb{R}^d &: \text{word distribution of topic } i \end{aligned}$$

- Task: recover $w_i$ and $\mu_i$ for all topic $i = 1 \dots K$

## Spectral Topic Modeling: Bag-of-Words

- Estimate a matrix $A \in \mathbb{R}^{d \times d}$ and a tensor $T \in \mathbb{R}^{d \times d \times d}$ defined by

$$A = \mathbf{E}\left[x_1 x_2^\top\right] \qquad \text{(expectation over bigrams)}$$

$$T = \mathbf{E}\left[x_1 x_2^\top x_3^\top\right] \qquad \text{(expectation over trigrams)}$$

- Claim: these are symmetric tensors in $w_i$ and $\mu_i$

$$A = \sum_{i=1}^{K} w_i \mu_i \mu_i^\top$$

$$T = \sum_{i=1}^{K} w_i \mu_i \mu_i^\top \mu_i^\top$$

- We can decompose $T$ using $A$ to recover $w_i$ and $\mu_i$ (Anandkumar et al. 2012)

## Spectral Topic Modeling: LDA

- Latent Dirichlet Allocation model with $K$ topics and $d$ words
  - Parameter vector $\alpha = (\alpha_1 \ldots \alpha_K) \in \mathbb{R}^K$
  - Define $\alpha_0 = \sum_i \alpha_i$
  - Dirichlet distribution over probability simplex $h \in \triangle^{K-1}$

$$p_\alpha(h) = \frac{\Gamma(\alpha_0)}{\prod_i \Gamma(\alpha_i)} \prod_i h_i^{\alpha_i - 1}$$

- A document can be a mixture of topics:
  1. Draw topic distribution $h = (h_1 \ldots h_K)$ from Dir($\alpha$)
  2. Draw words $x_1 \ldots x_l$ from the word distribution

$$h_1\mu_1 + \cdots + h_K\mu_K \in \mathbb{R}^d$$

- Task: assume $\alpha_0$ is known, recover $\alpha_i$ and $\mu_i$ for all topic $i = 1 \ldots K$

## Spectral Topic Modeling: LDA

- Estimate a vector $v \in \mathbb{R}^d$, a matrix $A \in \mathbb{R}^{d\times d}$ and a tensor $T \in \mathbb{R}^{d\times d\times d}$ defined by

$$
\begin{aligned}
v &= \mathbf{E}[x_1] \\
A &= \mathbf{E}\left[x_1 x_2^\top\right] - \frac{\alpha_0}{\alpha_0+1} v v^\top \\
T &= \mathbf{E}\left[x_1 x_2^\top x_3^\top\right] \\
&\quad - \frac{\alpha_0}{\alpha_0+2}\left(\mathbf{E}\left[x_1 x_2^\top v^\top\right] + \mathbf{E}\left[x_1 v^\top x_2^\top\right] + \mathbf{E}\left[v x_1^\top x_2^\top\right]\right) \\
&\quad + \frac{2\alpha_0^2}{(\alpha_0+2)(\alpha_0+1)}(v v^\top v^\top)
\end{aligned}
$$

## Spectral Topic Modeling: LDA

- Claim: these are symmetric tensors in $\alpha_i$ and $\mu_i$

$$A = \sum_{i=1}^{K} \frac{\alpha_i}{(\alpha_0 + 1)\alpha_0} \mu_i \mu_i^\top$$

$$T = \sum_{i=1}^{K} \frac{2\alpha_i}{(\alpha_0 + 2)(\alpha_0 + 1)\alpha_0} \mu_i \mu_i^\top \mu_i^\top$$

- We can decompose $T$ using $A$ to recover $\alpha_i$ and $\mu_i$ (Anandkumar et al. 2012)

## References I

[1] A. Anandkumar, D. Foster, D. Hsu, S. M. Kakade, and Y. Liu. A spectral algorithm for latent dirichlet allocation. arXiv:1204.6703, 2012.

[2] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent-variable models. arXiv:1210.7559, 2012.

[3] R. Bailly, A. Habrar, and F. Denis. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of ALT*, 2010.

[4] B. Balle and M. Mohri. Spectral learning of general weighted automata via constrained matrix completion. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2168–2176. 2012.

## References II

[5] B. Balle, A. Quattoni, and X. Carreras. A spectral learning algorithm for finite state transducers. In *Proceedings of ECML*, 2011.

[6] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*, 1998.

[7] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[8] S. B. Cohen, K. Stratos, M. Collins, D. F. Foster, and L. Ungar. Spectral learning of latent-variable PCFGs. In *Proceedings of ACL*, 2012.

[9] S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*, 2013.

## References III

[10] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

[11] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[12] P. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. H. Ungar. Spectral dependency parsing with latent variables. In *Proceedings of EMNLP*, 2012.

[13] J. Goodman. Parsing algorithms and metrics. In *Proceedings of ACL*, 1996.

[14] D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

## References IV

[15] H. Hotelling. Relations between two sets of variants. *Biometrika*, 28:321–377, 1936.

[16] D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*, 2009.

[17] H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6), 2000.

[18] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, (25):259–284, 1998.

[19] F. M. Luque, A. Quattoni, B. Balle, and X. Carreras. Spectral learning for non-deterministic dependency parsing. In *Proceedings of EACL*, 2012.

[20] T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic CFG with latent annotations. In *Proceedings of ACL*, 2005.

## References V

[21] A. Parikh, L. Song, and E. P. Xing. A spectral algorithm for latent tree graphical models. In *Proceedings of The 28th International Conference on Machine Learning (ICML 2011)*, 2011.

[22] S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*, 2006.

[23] L. Saul, F. Pereira, and O. Pereira. Aggregate and mixed-order markov models for statistical language processing. In *In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 81–89, 1997.

[24] A. Tropp, N. Halko, and P. G. Martinsson. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. In *Technical Report No. 2009-05*, 2009.

## References VI

[25] S. Vempala and G. Wang. A spectral algorithm for learning mixtures of distributions. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.