Spectracular Learning Algorithms for Natural Language Processing

Shay Cohen

University of Edinburgh

June 10, 2014

Latent-variable Models

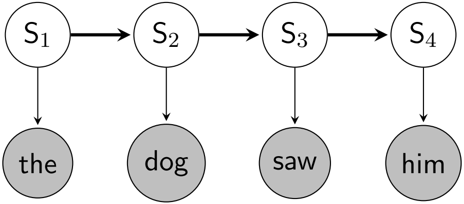
Latent-variable models are used in many areas of NLP, speech, etc.:

- Hidden Markov Models
- Latent-variable PCFGs
- 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 (Dempster et al., 1977)

Other relevant algorithms: cotraining, clustering methods

Example 1: Hidden Markov Models



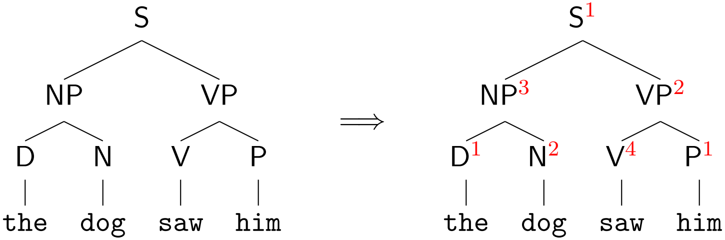
Parameterized by $\pi(s)$, $t(s|s')$ and $o(w|s)$

Spectral learning: Hsu et al. (2009)

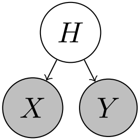
Dynamical systems: Siddiqi et al. (2009), Boots and Gordon (2011)

Head-automaton grammars for dep. parsing: Luque et al. (2012)

Example 2: Latent-Variable PCFGs (Matsuzaki et al., 2005; Petrov et al., 2006)



Example 3: Naïve Bayes

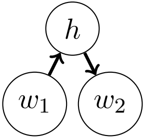


$$p(h, x, y) = p(h) \times p(x|h) \times p(y|h)$$

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

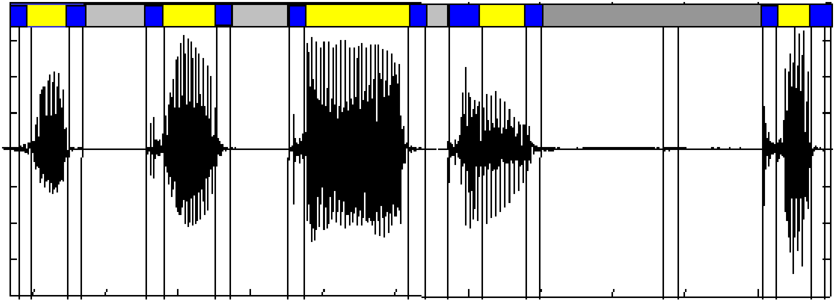
- EM can be used to estimate parameters

Example 4: Language Modelling



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

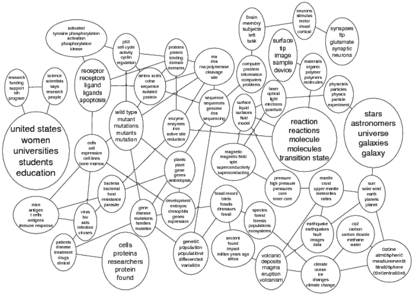
Example 5: HMMs for Speech



Phoneme boundaries are hidden variables

Refinement HMMs (Stratos et al., 2013)

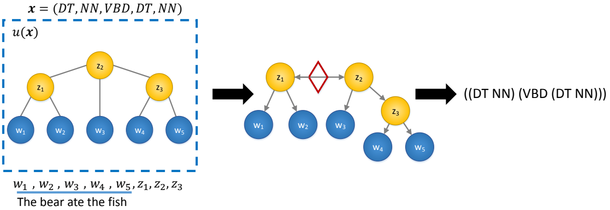
Example 6: Topic Models



Latent topics attached to a document or to each word in a document

Method of moments algorithms such as Arora et al. (2012; 2013)

Example 7: Unsupervised Parsing

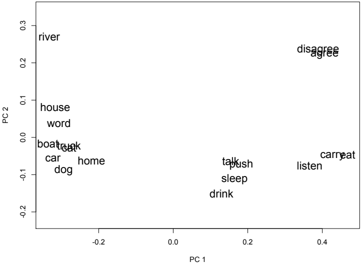


Latent structure is a bracketing (Parikh et al., 2014)

Similar in flavor to tree learning algorithms (e.g. Anandkumar, 2011)

Very different in flavor from estimation algorithms

Example 8: Word Embeddings



Embed a vocabulary into d-dimensional space

Can later be used for various NLP problems downstream

Related to canonical correlation analysis (Dhillon et al., 2012)

Spectral Methods

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

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

$$A \approx U \Sigma V^{\top}$$

which means

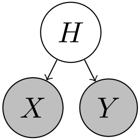
$$A_{jk} \approx \sum_{h=1}^{d} \sigma_h U_{jh} V_{kh}$$

where σ_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



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

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

- SVD approximation minimizes squared loss, not log-loss
- σ_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)

Outline

- Singular value decomposition
- Canonical correlation analysis
- Spectral learning of hidden Markov models
- Algorithm for latent-variable PCFGs

Singular Value Decomposition (SVD)

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

- ${\color{red}{d}} = \min(m, n)$

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)

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

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

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

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

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

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

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

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

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

rank 2

rank 3 (full-rank)

Matrix Rank: Alternative Definition

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

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

$$\Sigma = \begin{bmatrix} 4.53 & 0 & 0 \\ 0 & 0.7 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

rank 2

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

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

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)

SVD in Practice - Random Projections

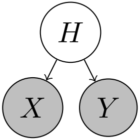
```
function [U,S,V] = svdsrand(A, k)

    n = size(A,2);
    m = size(A,1);

    Omega = randn(n, k);

    Y = A * Omega;

    [Q,R] = qr(Y, 0);

    B = Q' * A;

    [Uhat, S, V] = svds(B, k);

    U = Q*Uhat;
```

For large matrices (Halko et al., 2011)

Outline

- Singular value decomposition
- Canonical correlation analysis
- Spectral learning of hidden Markov models
- Algorithm for latent-variable PCFGs

Simplest Model in Complexity: Naive Bayes



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

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

CCA helps identify H

Canonical Correlation Analysis (CCA)

- Data consists of paired samples: $(x^{(i)}, y^{(i)})$ for $i = 1 \ldots n$
- 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
$x^{(1)} = (1, 0, 0, 0)$	$y^{(1)} = (1, 0, 0, 1, 0, 1, 0)$
$x^{(2)} = (0, 0, 1, 0)$	$y^{(2)} = (0, 1, 0, 0, 0, 0, 1)$
$\vdots$	$\vdots$
$x^{(100000)} = (0, 1, 0, 0)$	$y^{(100000)} = (0, 0, 1, 0, 1, 1, 1)$

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 $i = 1 \ldots n$:

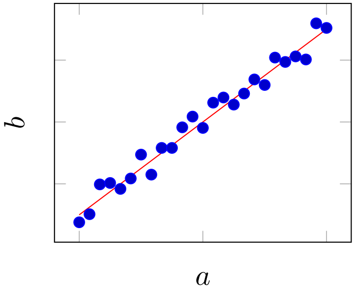
$$x^{(i)} \in \mathbb{R}^d \Longrightarrow A^\top x^{(i)} \in \mathbb{R}^p$$
$$y^{(i)} \in \mathbb{R}^{d'} \Longrightarrow B^\top y^{(i)} \in \mathbb{R}^p$$

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 ≈ 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 $j = 1 \ldots p$ (each column of A and B)

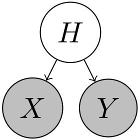
$$u_j, v_j = \arg\max_{u,v} \ \text{Corr}\left(\{u \cdot x^{(i)}\}_{i=1}^n, \{v \cdot y^{(i)}\}_{i=1}^n\right)$$

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

“Variants” of CCA

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

Centering leads to non-sparse C_{XY}.

Computing $C_{XX}^{-1/2}$ and $C_{YY}^{-1/2}$ leads to large non-sparse matrices

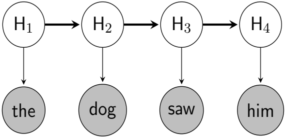
Outline

- Singular value decomposition
- Canonical correlation analysis
- Spectral learning of hidden Markov models
- Algorithm for latent-variable PCFGs

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

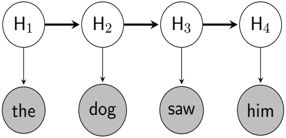
Hidden Markov Models (HMMs)



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

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

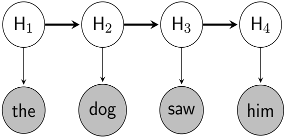
Hidden Markov Models (HMMs)



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

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

Hidden Markov Models (HMMs)

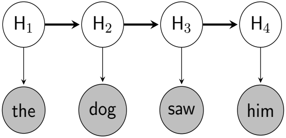


$$p(\underbrace{\text{the dog saw him}}_{x_1 \ldots x_4}, \underbrace{\color{red}{1\ 2\ 1\ 3}}_{h_1 \ldots h_4})$$

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

- 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

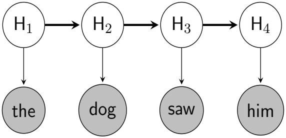
Hidden Markov Models (HMMs)



Throughout this section:

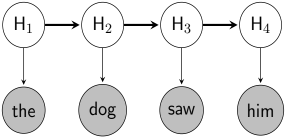
- **We use m to refer to the number of hidden states**
- **We use n to refer to the number of possible words (observations)**
- Typically, $m \ll n$ (e.g., $m = 20$, $n = 50,000$)

HMMs: the forward algorithm



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

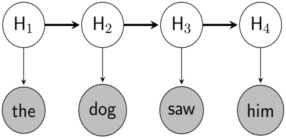
HMMs: the forward algorithm



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

The forward algorithm:

HMMs: the forward algorithm

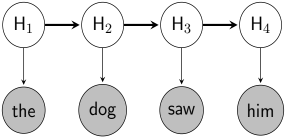


$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

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

HMMs: the forward algorithm

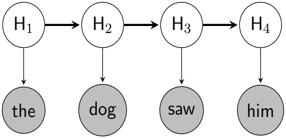


$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

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

HMMs: the forward algorithm



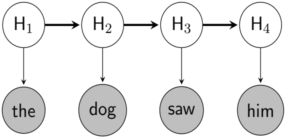
$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

The forward algorithm:

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

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

HMMs: the forward algorithm



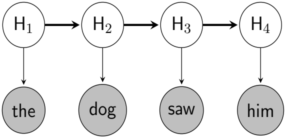
$$p(\text{the dog saw him}) = \sum_{h_1,h_2,h_3,h_4} p(\text{the dog saw him}, h_1\ h_2\ h_3\ h_4)$$

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



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

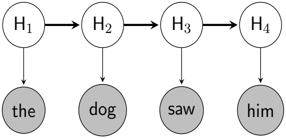
The forward algorithm:

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

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

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

HMMs: the forward algorithm



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

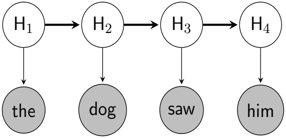
The forward algorithm:

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

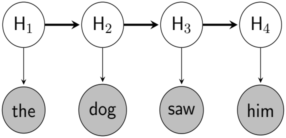
$$f_h^2 = \sum_{h'} t(h|h')o(\text{dog}|h')f_{h'}^1 \quad f_h^3 = \sum_{h'} t(h|h')o(\text{saw}|h')f_{h'}^2$$

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

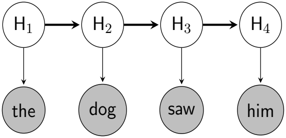
HMMs: the forward algorithm in matrix form



HMMs: the forward algorithm in matrix form



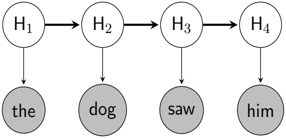
HMMs: the forward algorithm in matrix form



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

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

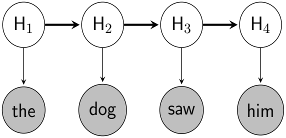
HMMs: the forward algorithm in matrix form



- 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

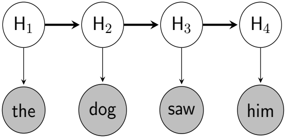


- For each word x, define the matrix $A_x \in \mathbb{R}^{m \times m}$ as

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

- Define π as vector with elements π_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

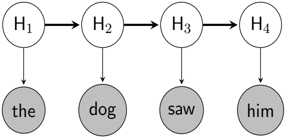
$$[A_x]_{h',h} = t(h'|h)o(x|h) \quad \text{e.g., } [A_{\text{the}}]_{h',h} = t(h'|h)o(\text{the}|h)$$

- Define π as vector with elements π_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



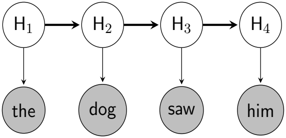
Define the following matrix $P_{2,1} \in \mathbb{R}^{n \times n}$:

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

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 $P_{3,x,1} \in \mathbb{R}^{n \times n}$:

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

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

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

Why does this matter?

- Theorem: if $P_{2,1}$ is of rank m, then
$$B_x = GA_xG^{-1}$$
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
$$\begin{aligned} & B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \\ = \quad & GA_{\text{him}}G^{-1} \times GA_{\text{saw}}G^{-1} \times GA_{\text{dog}}G^{-1} \times GA_{\text{the}}G^{-1} \\ = \quad & GA_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}}G^{-1} \end{aligned}$$
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

$$B_x = GA_xG^{-1}$$

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

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

The G's cancel!

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

$$\begin{aligned} & b^\infty \times B_{\text{him}} \times B_{\text{saw}} \times B_{\text{dog}} \times B_{\text{the}} \times b^0 \\ = \quad & 1^\top \times A_{\text{him}} \times A_{\text{saw}} \times A_{\text{dog}} \times A_{\text{the}} \times \pi \end{aligned}$$

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^∞, 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^∞, 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 σ 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 ϵ propagates through the SVD step, the various matrix multiplications, etc etc. The "rate" at which ϵ 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 $x_1 \ldots x_n$
- 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

Outline

- Singular value decomposition
- Canonical correlation analysis
- Spectral learning of hidden Markov models
- Algorithm for latent-variable PCFGs

Problems with spectral HMM learning algorithm

Parameters are masked by an unknown linear transformation

- Negative marginals (due to sampling error)
- Parameters cannot be easily interpreted
- Cannot improve parameters using, for example, EM

Hsu et al. suggest a way to extract probabilities, but the method is unstable

This part of the talk

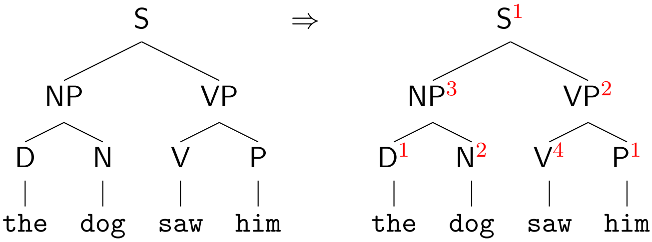
Like the spectral algorithm, has theoretical guarantees

Estimates are actual probabilities

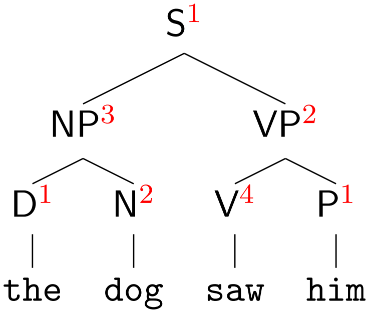
More efficient than EM

Can be used to initialize EM, which converges in an iteration or two

L-PCFGs (Matsuzaki et al., 2005; Petrov et al., 2006)



The probability of a tree

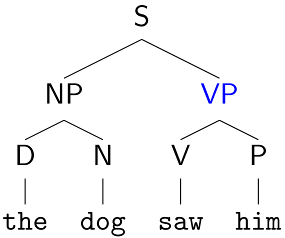


$$
\begin{aligned}
& p(\text{tree}, 1\ 3\ 1\ 2\ 2\ 4\ 1) \\
&= \pi(\mathsf{S}^1)\times \\
&\quad t(\mathsf{S}^1 \rightarrow \mathsf{NP}^3\ \ \mathsf{VP}^2 | \mathsf{S}^1)\times \\
&\quad t(\mathsf{NP}^3 \rightarrow \mathsf{D}^1\ \ \mathsf{N}^2 | \mathsf{NP}^3)\times \\
&\quad t(\mathsf{VP}^2 \rightarrow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

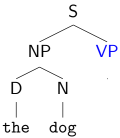
$$
p(\text{tree}) = \sum_{h_1 \ldots h_7} p(tree, h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7)
$$

Inside and Outside Trees

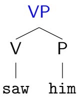
At node VP:



Outside tree $o =$



Inside tree $t =$



Conditionally independent given the label and the hidden state

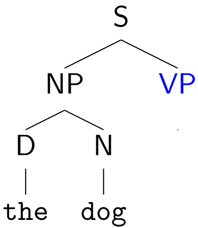
$$p(o, t|\text{VP}, h) = p(o|\text{VP}, h) \times p(t|\text{VP}, h)$$

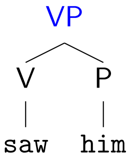
Designing Feature Functions

Design functions ψ and ϕ:

ϕ maps any inside tree to a binary vector of length d

ψ maps any outside tree to a binary vector of length d'





Outside tree $o \Rightarrow$

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

Inside tree $t \Rightarrow$

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

ψ and ϕ as multinomials $p(f)$ for $f \in [d]$ and $p(g)$ for $g \in [d']$.

Latent State Distributions

Think of f and g as representing a whole inside/outside tree

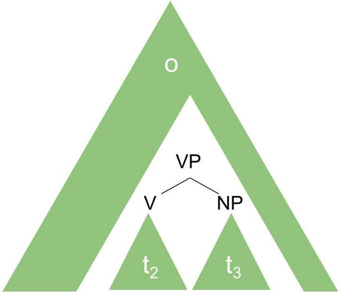
Say we had a way of getting:

- $p(f|h, \text{VP})$ for each h and f inside feature
- $p(g|h, \text{VP})$ for each h and g outside feature

Then we could run EM on a convex problem to find parameters.
How?

Binary rule estimation

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



At sample i

- $g^{(i)}$ = outside feature at VP
- $f_2^{(i)}$ = inside feature at V
- $f_3^{(i)}$ = inside feature at NP

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

$$= \max_{\hat{t}} \sum_{i=1}^{M} \log \sum_{h_1,h_2,h_3} \Big(\hat{t}(h_1, h_2, h_3 | \text{VP} \rightarrow \text{V NP}) \times$$

$$p(g^{(i)} | h_1, \text{VP}) p(f_2^{(i)} | h_2, \text{V}) p(f_3^{(i)} | h_3, \text{NP}) \Big)$$

Binary Rule Estimation

- Use Bayes rule to convert

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

to

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

- The log-likelihood function is convex, and therefore EM converges to global maximum

- Estimation of π and q is similar in flavor

Main question: how do we get the latent state distributions $p(h|f, \textbf{VP})$ and $p(h|g, \textbf{VP})$?

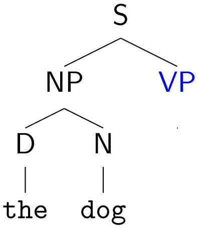
Vector Representation of Inside and Outside Trees

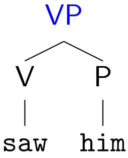
Design functions Z and Y:

Y maps any inside feature value $f \in [d']$ to a vector of length m.

Z maps any outside feature value $g \in [d]$ to a vector of length m.

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





Outside tree $o \Rightarrow$

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

Inside tree $t \Rightarrow$

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

Z and Y reduce the dimensionality of ϕ and ψ using CCA

Identifying Latent State Distributions

- For each $f \in [d]$, define:
$v(f) = \sum_{g=1}^{d'} p(g|f, \text{VP}) Z(g) = E[Z(g)|f, \text{VP}]$

- $v(f) \in \mathbb{R}^m$ is **"the expected value of an outside tree (representation) given an inside tree (feature)"**

Identifying Latent State Distributions

- For each $f \in [d]$, define:
$v(f) = \sum_{g=1}^{d'} p(g|f, \text{VP})Z(g) = E[Z(g)|f, \text{VP}]$

- $v(f) \in \mathbb{R}^m$ is **"the expected value of an outside tree (representation) given an inside tree (feature)"**

- By conditional independence:

$$v(f) = \sum_{h=1}^{m} p(h|f, \text{VP})w(h)$$

where $w(h) \in \mathbb{R}^m$ and
$w(h) = \sum_{g=1}^{d'} p(g|h, \text{VP})Z(g) = E[Z(g)|h, \text{VP}]$.

- $w(h)$ is **"the expected value of an outside tree (representation) given a latent state"**

Pivot Assumption

Reminder: $v(f) = \sum_{h=1}^{m} p(h|f, \text{VP}) w(h)$

- If we know $w(h)$, we can find latent state distributions:
  - Given an inside tree (feature f) and a node such as VP, compute $v(f)$
  - Solve

$$\arg \min_{p(h|f,\text{VP})} ||v(f) - \sum_{h=1}^{m} p(h|f, \text{VP}) w(h)||_2$$

Assumption: For each latent state there is $f \in [d]$ a "pivot feature value" s.t.

$$p(h|f, \text{VP}) = 1$$

.

Result of this: $v(f) = w(h)$ for any pivot feature f

Identifying Latent State Distributions

- m pivot features $\{f_1, \ldots, f_m\}$ such that $v(f_h) = w(h)$
Then, for all $f \in [d]$

$$v(f) = \sum_{h=1}^{m} p(h|f, \text{VP}) v(f_h)$$

- Therefore, we can identify $p(h|f, \text{VP})$ for all f by solving:

$$\arg \min_{p(h|f,\text{VP})} ||v(f) - \sum_{h=1}^{m} p(h|f, \text{VP}) v(f_h)||_2$$

Identifying Pivot Features

- $v(f)$ are observable quantities, can be calculated from data
- Arora et al. (2012) showed how to find the pivot features
- Basic idea: find the corners of the convex hull spanned by the d features

Identifying Latent State Distributions

Algorithm: • Identify m pivot features $f_1, \ldots, f_m$ by finding vertices of $\mathrm{ConvexHull}(v_1, \ldots, v_d)$ (Arora et al., 2012)
• Solve for each $f \in [d]$:

$$\arg \min_{p(h|f,\mathsf{VP})} ||v(f) - \sum_{h=1}^{m} p(h|f,\mathsf{VP}) v(f_h)||_2$$

Output:

- Latent state distributions $p(h|f,\mathsf{VP})$ for any $f \in [d]$

Can analogously get:

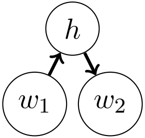
- Latent state distributions $p(h|g,\mathsf{VP})$ for any $g \in [d']$

• We managed to extract latent state probabilities from observed data only!

Experiments - Language Modeling

- Saul and Pereira (1997):

$$p(w_2|w_1) = \sum_{h} p(w_2|h)p(h|w_1).$$



This model is a specific case of L-PCFG

- Experimented with bi-gram modeling for two corpora: Brown corpus and Gigaword corpus

Results: perplexity

	Brown			NYT		
m	*128*	*256*	test	*128*	*256*	test
bigram Kneser-Ney	408		415	271		279
trigram Kneser-Ney	386		394	150		158
EM iterations	388 9	365 8	364	284 35	265 32	267
pivot	426	597	560	782	886	715

Results: perplexity

	Brown			NYT		
m	*128*	*256*	test	*128*	*256*	test
bigram Kneser-Ney	408		415	271		279
trigram Kneser-Ney	386		394	150		158
EM	388	365	364	284	265	267
iterations	9	8		35	32	
pivot	426	597	560	782	886	715
pivot+EM	**310**	**327**	357	**279**	292	281
iterations	1	1		19	12	

- Initialize EM with pivot algorithm output
- EM converges in much fewer iterations
- Still consistent - called "two-step estimation" (Lehmann and Casella, 1998)

Results with EM (section 22 of Penn treebank)

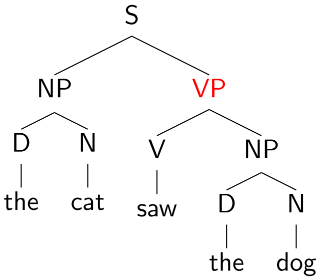
Performance with expectation-maximization ($m = 32$): 88.56%

Vanilla binarized PCFG maximum likelihood estimation performance: 68.62%

Performance with spectral algorithm (Cohen et al., 2013): 88.82%

Inside features used

Consider the VP node in the following tree:

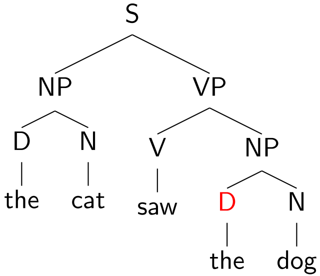


The inside features consist of:

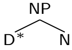
- The pairs `(VP, V)` and `(VP, NP)`
- The rule `VP` $\rightarrow$ `V NP`
- The tree fragment `(VP (V saw) NP)`
- The tree fragment `(VP V (NP D N))`
- The pair of head part-of-speech tag with VP: `(VP, V)`

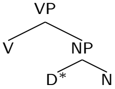
Outside features used

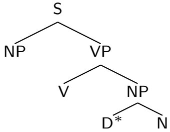
Consider the D node in the following tree:



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)`

Results

m	sec. 22 8	16	24	32	sec. 23
EM iterations	86.69 40	88.32 30	88.35 30	88.56 20	87.76
Spectral (Cohen et al., 2013)	85.60	87.77	88.53	88.82	88.05
Pivot	83.56	86.00	86.87	86.40	85.83
Pivot+EM iterations	86.83 2	88.14 6	88.64 2	88.55 2	88.03

Again, EM converges in very few iterations

Conclusion

Formal guarantees:

- Statistical consistency
- No problem of local maxima

Advantages over traditional spectral methods:

- No negative probabilities
- More intuitive to understand

Things we did not talk about

Theses algorithms can be kernelized (e.g. Song et al., 2010)

Many other algorithms similar in flavor (see reading list)

- Rely on some decomposition of observable quantities to get a handle on the parameters

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] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. *arXiv preprint arXiv:1212.4777*, 2012.

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

References II

[5] 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.

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

[7] Byron Boots and Geoffrey J Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *AAAI*, 2011.

[8] S. B. Cohen and M. Collins. A provably correct learning algorithm for latent-variable PCFGs. In *Proceedings of ACL*, 2014.

References III

[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.

[10] S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *Journal of Machine Learning Research*, 2014.

[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, D. P. Foster, and L. H. Ungar. Multi-view learning of word embeddings via CCA. In *Proceedings of NIPS*, 2011.

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

References IV

[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.

[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.

References V

[19] Percy Liang, Daniel J Hsu, and Sham M Kakade. Identifiability and unmixing of latent parse trees. In *Advances in Neural Information Processing Systems*, pages 1511–1519, 2012.

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

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

[22] 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.

References VI

[23] A. P. Parikh, S. B. Cohen, and E. Xing. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of ACL*, 2014.

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

[25] 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.

[26] L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and Alex J Smola. Hilbert space embeddings of hidden markov models. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 991–998, 2010.

References VII

[27] 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.

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