STATISTICAL STRUCTURED PREDICTION

José Miguel Benedí **e-mail**: jmbenedi@prhlt.upv.es

Joan Andreu Sánchez **e-mail**: jandreu@prhlt.upv.es

Syllabus

1. Introduction
2. Models for Statistical Structured Prediction
3. Making Predictions: Decoding and Inference
4. Model Parameter Estimation

Bibliography

Statistical Models for Natural Language Processing

- Christopher D. Manning and Hinrich Schütze: Foundations of Statistical Natural Language Processing. The MIT Press, 1999.
- Daniel Jurafsky and James H. Martin: Speech and Language Processing. Prentice Hall, 2007 (2nd ed.), 2009.

Pattern Recognition and Machine Learning

- Richard O. Duda, Peter E. Hart and David G. Stork: Pattern Classification (2nd ed.). Wiley Interscience, 2000.
- Christopher M. Bishop: Pattern Recognition and Machine Learning. Springer, 2006.
- Kevin P. Murphy: Machine Learning: A Probabilistic Perpective. The MIT Press, 2012.

Statistical Structured Prediction

Formal Languages Theory

- M.A.Harrison: Introduction to Formal Languages Theory. Addison Wesley, 1978.
- J.E.Hopcroft and Jeffrey D. Ullman: Introduction to Automata Theory, Languages and Computation. Addison Wesley, 1979.

Probability Theory

- Morris H. Degroot: Probability and Statistics. Addison Wesley, 1986.

Information Theory

- Thomas M. Cover and Joy A. Thomas: Elements of Information Theory. John Wiley and Sons, 1991.

Natural Language Processing

- James Allen: Natural Language Understanding. The Benjamin/Cummings Publishing Company, Inc., 1995.

Lectures

Monday 15:00 - 18:00 (10 sessions of 3 hours)

J.M. Benedí November 06 first session

J.A. Sánchez December 11 first session

Assessment

2 Collections of selected exercises (theoretical and/or practical)

1 Multiple-choice test (February 12 2024)

	delivery	**deadline**
Q2	02 - 02 - 2024	09 - 02 - 2024

Tutoring

Tutoring available by previous appointment

4. Model Parameter Estimation

4.1. Introduction

4.2. Unsupervised Learning: Generative Models

- HMMs: *Forward-Backward* algorithm
- PCFGs: *Inside-outside* algorithm
- PCFGs: estimation from a subset of derivations

4.3. Supervised Learning: Generative Models

- Discriminative Training

4.4. Supervised Learning: Discriminative Models

- Parameter estimation in CRFs
- Learning structured predictors: Structured Perceptron

4.5. Advanced Issues in Model Parameter Estimation

- On-Line learning
- Active learning

Model Parameter Estimation

Let $\mathcal{D}$ be a training corpus, and given a statistical model defined by the parameter θ, the problem to learn θ can be formulated as:

$$\hat{\theta} = \arg\max_{\theta} L_\theta(\mathcal{D})$$

Model Parameter Estimation

Let $\mathcal{D}$ be a training corpus, and given a statistical model defined by the parameter θ, the problem to learn θ can be formulated as:

$$\hat{\theta} = \arg\max_{\theta} L_\theta(\mathcal{D})$$

1. Optimization methods:

- Analytical optimization
- Constrained optimization: Lagrange multipliers
- Gradient descent
- Expectation-maximization algorithm
- Growth transformations (Baum-Eagon and extended Baum-Eagon methods)

2. Learning strategies:

- Given a fully annotated training set $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$ Supervised learning
- Given a training set only with inputs $\mathcal{D} = \{x^{(n)}\}_{n=1}^{N}$ Unsupervised learning

Other learning strategies

- Semi-supervised learning makes use of a small amount of labeled data along with a large amount of unlabeled data.
- Reinforcement Learning allows models to be learnt by maximizing their performance.
  A reward feedback (reinforcement signal) is required for the model to learn the ideal behaviour within a specific context.
- On-Line learning
- Active learning

3. Objective functions

➢ Maximum likelihood (generative) estimation

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(\boldsymbol{x}^{(n)})$$

3. Objective functions

➢ Maximum likelihood (generative) estimation

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(\boldsymbol{x}^{(n)})$$

➢ **Supervised learning**

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(x^{(n)}, y^{(n)}) = \prod_{n=1}^{N} P_\theta(x^{(n)}|y^{(n)}) \cdot P_\theta(y^{(n)})$$

Can be learned independently

3. Objective functions

➢ Maximum likelihood (generative) estimation

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(\boldsymbol{x}^{(n)})$$

➢ **Supervised learning**

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(x^{(n)}, y^{(n)}) = \prod_{n=1}^{N} P_\theta(x^{(n)}|y^{(n)}) \cdot P_\theta(y^{(n)})$$

Can be learned independently

➢ **Unsupervised learning**

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(x^{(n)}) = \prod_{n=1}^{N} \sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)}, y) = \prod_{n=1}^{N} \sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)}|y) \cdot P_\theta(y)$$

Model Parameter Estimation

➢ Maximum conditional likelihood (discriminative) estimation (Supervised learning)

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)})$$

➢ Maximum conditional likelihood (discriminative) estimation (Supervised learning)

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}, y^{(n)})}{P_\theta(x^{(n)})} = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)} \mid y^{(n)}) \cdot P_\theta(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)} \mid y) \cdot P_\theta(y)}$$

Model Parameter Estimation

➢ Maximum conditional likelihood (discriminative) estimation (Supervised learning)

$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}, y^{(n)})}{P_\theta(x^{(n)})} = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)} \mid y^{(n)}) \cdot P_\theta(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)} \mid y) \cdot P_\theta(y)}$$

➢ Maximum mutual information (discriminative) estimation (Supervised learning)

$$L_\theta(\mathcal{D}) = \sum_{n=1}^{N} \log \left[\frac{P_\theta^\alpha(x^{(n)}|y^{(n)}) \cdot P_\theta^\alpha(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta^\alpha(x^{(n)}|y) \cdot P_\theta^\alpha(y)} \right]$$

Model Parameter Estimation

➢ Maximum conditional likelihood (discriminative) estimation (Supervised learning)

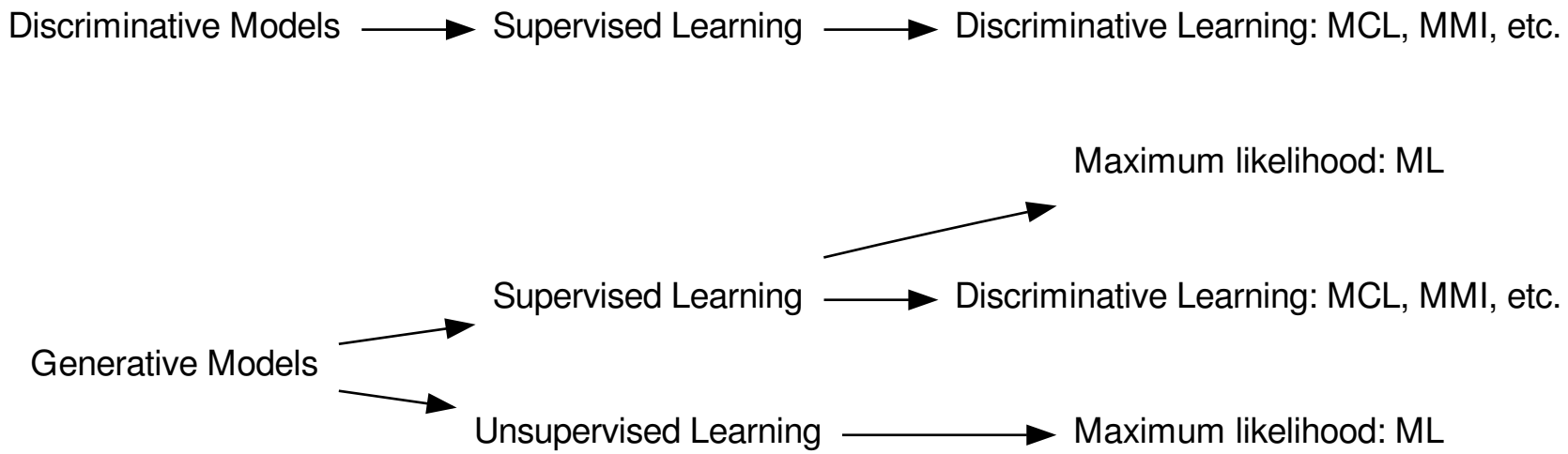
$$L_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}, y^{(n)})}{P_\theta(x^{(n)})} = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)} \mid y^{(n)}) \cdot P_\theta(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)} \mid y) \cdot P_\theta(y)}$$

➢ Maximum mutual information (discriminative) estimation (Supervised learning)

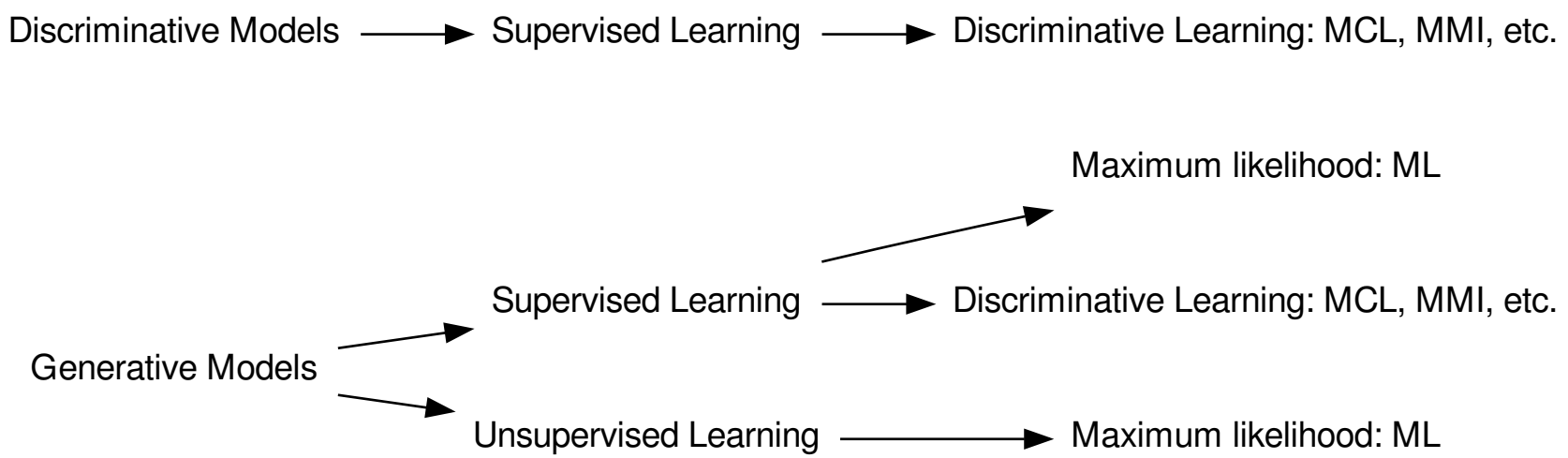
$$L_\theta(\mathcal{D}) = \sum_{n=1}^{N} \log \left[\frac{P_\theta^\alpha(x^{(n)}|y^{(n)}) \cdot P_\theta^\alpha(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta^\alpha(x^{(n)}|y) \cdot P_\theta^\alpha(y)} \right]$$

➢ Minimum classification error, ...

Model Parameter Estimation



Model Parameter Estimation



- L.R.Bahl, P.F.Brown, P.V.De Souza, R.L.Mercer (1986) Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In Proceedings of IEEE (ICASSP'86):49-52.
- L.E.Baum, G.Sell (1968) Growth transformation for functions on manifolds. Pacific Journal of Mathematics 27(2):211-227.
- A.P.Dempster, N.M.Laird, D.B.Rubin (1977) Maximum likelihood from incomplete data via the EM algorithm (with discussion). Journal of the Royal Statistical Society B39:1-38.
- P.S.Gopalakrishnan, D.Kanevsky, A.Nadas, D.Nahamoo (1991) An inequality for rational functions with applications to some statistical estimation problems. IEEE Transactions on Information Theory 37(1):107-113.
- T.Jebara, A.Pentland (1998) Maximum conditional likelihood via bound maximization and the CEM algorithm. In: Proceedings of Advances in Neural Information Processing Systems, vol.11.

Objective function: **(Log -) Maximum Likelihood estimation**

$$\log\ L_\theta(\mathcal{D}) \ = \ \log \prod_{x \in \mathcal{D}} P_\theta(x) \ = \ \log \prod_{x \in \mathcal{D}} \sum_{y \in \theta(x)} P_\theta(x, y) \ = \ \sum_{x \in \mathcal{D}} \log\ P_\theta(x)$$

Objective function: **(Log -) Maximum Likelihood estimation**

$$\log\ L_\theta(\mathcal{D})\ =\ \log \prod_{x\in\mathcal{D}} P_\theta(x)\ =\ \log \prod_{x\in\mathcal{D}} \sum_{y\in\theta(x)} P_\theta(x,y)\ =\ \sum_{x\in\mathcal{D}} \log\ P_\theta(x)$$

Optimization method: **Growth Transformations**

Baum and Eagon Theorem [Baum and Sell, 68] [Gopalakrishnan et al., 91]

Let $L(\theta)$ be a homogeneous polynomial with non-negative coefficients. Let $\theta = \{\theta_{ij}\}$ be a point in the domain $\Theta = \{\theta_{ij} \mid \theta_{ij} \geq 0; \sum_{j=1}^{q_i} \theta_{ij} = 1,\ i = 1, \ldots, p;\}$, and let $Q(\theta)$ be a close transformation in Θ, that is defined as:

$$Q(\theta_{ij}^{(w)})\ =\ \theta_{ij}^{(w+1)}\ =\ \frac{\theta_{ij}^{(w)}\ \left(\frac{\partial\ L}{\partial\ \theta_{ij}}\right)_{\theta_{ij}=\theta_{ij}^{(w)}}}{\sum_{k=1}^{q_i}\ \theta_{ik}^{(w)}\ \left(\frac{\partial\ L}{\partial\ \theta_{ik}}\right)_{\theta_{ik}=\theta_{ik}^{(w)}}} \tag{1}$$

with the denominator different from zero. Then, $L(Q(\theta)) > L(\theta)$ except if $Q(\theta) = \theta$

Unsupervised Learning

input $L(\theta)$
$\theta =$ initial values
repeat
    compute $Q(\theta)$ using $L(\theta)$
    $\theta = Q(\theta)$
until convergence
output θ

input $L(\theta)$
$\theta =$ initial values
repeat
    compute $Q(\theta)$ using $L(\theta)$
    $\theta = Q(\theta)$
until convergence
output θ

$$\overline{\theta}_{ij} = \frac{\theta_{ij} \frac{\partial \log P_\theta(\mathcal{D})}{\partial \theta_{ij}}}{\sum_{k=1}^{q_i} \theta_{ik} \frac{\partial \log P_\theta(\mathcal{D})}{\partial \theta_{ik}}} = \frac{\theta_{ij} \frac{\partial \log \prod_{x \in \mathcal{D}} P_\theta(x)}{\partial \theta_{ij}}}{\sum_{k=1}^{q_i} \theta_{ik} \frac{\partial \log \prod_{x \in \mathcal{D}} P_\theta(x)}{\partial \theta_{ik}}}$$

$$\frac{\partial \sum_{x \in \mathcal{D}} \log P_\theta(x)}{\partial \theta_{ij}} = \frac{\sum_{x \in \mathcal{D}} \partial \log P_\theta(x)}{\partial \theta_{ij}} = \sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \frac{\partial P_\theta(x)}{\partial \theta_{ij}}$$

$$\overline{\theta}_{ij} = \frac{\sum_{x \in \mathcal{D}} \frac{\theta_{ij}}{P_\theta(x)} \frac{\partial P_\theta(x)}{\partial \theta_{ij}}}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{k=1}^{q_i} \theta_{ik} \frac{\partial P_\theta(x)}{\partial \theta_{ik}}} \quad (2)$$

HMMs: Forward-Backward algorithm

Given a HMM $(\mathcal{Q}, \Sigma, \theta)$, where $\mathcal{Q}$ is a set of states, Σ is a set of output symbols, and θ is a vector of parameters that contains two types of parameters:

(for the sake of simplicity we consider HMMs with only one non-emiting intiial state and one non-emiting final state)

- $q(u|v) : u, v \in \mathcal{Q}$ is the transition probability from state v to state u $(v \xrightarrow{q(u|v)} u)$.
- $e(b|u) : u \in \mathcal{Q}, b \in \Sigma$ is the probability of emitting symbol b from state u $\begin{pmatrix} b \\ \uparrow e(b|u) \\ u \end{pmatrix}$.

Given a HMM $(\mathcal{Q}, \Sigma, \theta)$, where $\mathcal{Q}$ is a set of states, Σ is a set of output symbols, and θ is a vector of parameters that contains two types of parameters:

$q(u|v): \ u, v \in \mathcal{Q}$ is the transition probability from state v to state u $\left(v \overset{q(u|v)}{\longrightarrow} u\right)$.

$e(b|u): \ u \in \mathcal{Q}, b \in \Sigma$ is the probability of emitting symbol b from state u $\begin{pmatrix} b \\ \uparrow e(b|u) \\ u \end{pmatrix}$.

From (2):

$$\overline{q}(u|v) = \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_\pi \mathrm{N}(v \to u, \pi) \, P_\theta(x, \pi)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_\pi \mathrm{N}(v, \pi) \, P_\theta(x, \pi)} \tag{3}$$

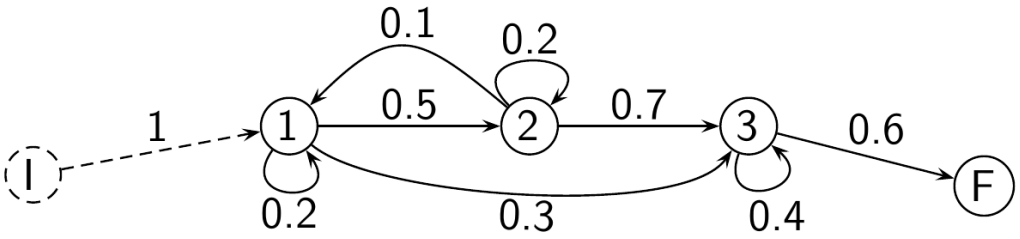
$$\overline{e}(b|u) = \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_\pi \mathrm{N}(\overset{b}{\underset{u}{\uparrow}}, \pi) \, P_\theta(x, \pi)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_\pi \mathrm{N}(v, \pi) \, P_\theta(x, \pi)} \tag{4}$$

where $\mathrm{N}(\cdot, \cdot)$ is the number of times that first argument appears in the second argument, and π_x is a state sequence that accounts for string x.

HMMs: Forward-Backward algorithm

Example:

a	0.0	0.3	0.8
b	0.3	0.6	0.2
c	0.7	0.1	0.0



$$\pi_x^1: \quad I \xrightarrow{1.0} 1 \xrightarrow[\uparrow 0.3]{0.2} 1 \xrightarrow[\uparrow 0.7]{0.5} 2 \xrightarrow[\uparrow 0.6]{0.7} 3 \xrightarrow[\uparrow 0.8]{0.6} F \quad 0.004234$$

$$\pi_x^2: \quad I \xrightarrow{1.0} 1 \xrightarrow[\uparrow 0.3]{0.5} 2 \xrightarrow[\uparrow 0.1]{0.2} 2 \xrightarrow[\uparrow 0.6]{0.7} 3 \xrightarrow[\uparrow 0.8]{0.6} F \quad 0.000605$$

$$\pi_x^3: \quad I \xrightarrow{1.0} 1 \xrightarrow[\uparrow 0.3]{0.2} 1 \xrightarrow[\uparrow 0.7]{0.3} 3 \xrightarrow[\uparrow 0.2]{0.4} 3 \xrightarrow[\uparrow 0.8]{0.6} F \quad 0.000484$$

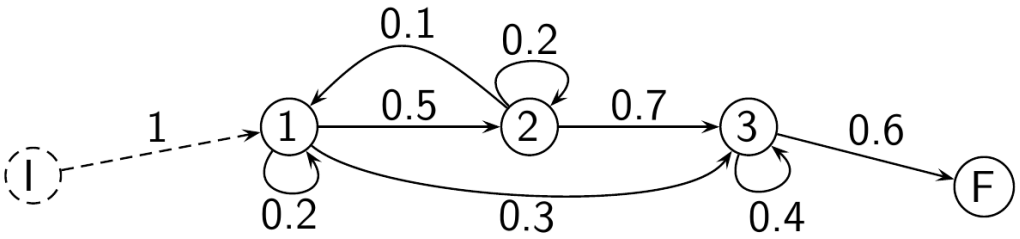
$$\pi_x^4: \quad I \xrightarrow{1.0} 1 \xrightarrow[\uparrow 0.3]{0.5} 2 \xrightarrow[\uparrow 0.1]{0.7} 3 \xrightarrow[\uparrow 0.2]{0.4} 3 \xrightarrow[\uparrow 0.8]{0.6} F \quad 0.000403$$

$$\pi_x^5: \quad I \xrightarrow{1.0} 1 \xrightarrow[\uparrow 0.3]{0.2} 1 \xrightarrow[\uparrow 0.7]{0.2} 1 \xrightarrow[\uparrow 0.3]{0.3} 3 \xrightarrow[\uparrow 0.8]{0.6} F \quad 0.000363$$

$$\pi_x^6: \quad I \xrightarrow{1.0} 1 \xrightarrow[\uparrow 0.3]{0.5} 2 \xrightarrow[\uparrow 0.1]{0.1} 1 \xrightarrow[\uparrow 0.3]{0.3} 3 \xrightarrow[\uparrow 0.8]{0.6} F \quad 0.000065$$

Example:

$$\begin{array}{c} a \\ b \\ c \end{array}\begin{bmatrix} 0.0 \\ 0.3 \\ 0.7 \end{bmatrix} \qquad \begin{bmatrix} 0.3 \\ 0.6 \\ 0.1 \end{bmatrix} \qquad \begin{bmatrix} 0.8 \\ 0.2 \\ 0.0 \end{bmatrix}$$



For $\overline{q}(1|1)$ the inner addition in (4) is:

$$0.004234 + 0.000484 + 2 \cdot 0.000363$$

$$\pi_x^1: \quad I \xrightarrow{1.0} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.2} \underset{\uparrow 0.7}{\overset{c}{1}} \xrightarrow{0.5} \underset{\uparrow 0.6}{\overset{b}{2}} \xrightarrow{0.7} \underset{\uparrow 0.8}{\overset{a}{3}} \xrightarrow{0.6} F \quad 0.004234$$

$$\pi_x^2: \quad I \xrightarrow{1.0} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.5} \underset{\uparrow 0.1}{\overset{c}{2}} \xrightarrow{0.2} \underset{\uparrow 0.6}{\overset{b}{2}} \xrightarrow{0.7} \underset{\uparrow 0.8}{\overset{a}{3}} \xrightarrow{0.6} F \quad 0.000605$$

$$\pi_x^3: \quad I \xrightarrow{1.0} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.2} \underset{\uparrow 0.7}{\overset{c}{1}} \xrightarrow{0.3} \underset{\uparrow 0.2}{\overset{b}{3}} \xrightarrow{0.4} \underset{\uparrow 0.8}{\overset{a}{3}} \xrightarrow{0.6} F \quad 0.000484$$

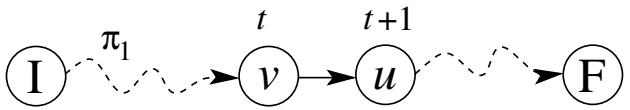
$$\pi_x^4: \quad I \xrightarrow{1.0} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.5} \underset{\uparrow 0.1}{\overset{c}{2}} \xrightarrow{0.7} \underset{\uparrow 0.2}{\overset{b}{3}} \xrightarrow{0.4} \underset{\uparrow 0.8}{\overset{a}{3}} \xrightarrow{0.6} F \quad 0.000403$$

$$\pi_x^5: \quad I \xrightarrow{1.0} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.2} \underset{\uparrow 0.7}{\overset{c}{1}} \xrightarrow{0.2} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.3} \underset{\uparrow 0.8}{\overset{a}{3}} \xrightarrow{0.6} F \quad 0.000363$$

$$\pi_x^6: \quad I \xrightarrow{1.0} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.5} \underset{\uparrow 0.1}{\overset{c}{2}} \xrightarrow{0.1} \underset{\uparrow 0.3}{\overset{b}{1}} \xrightarrow{0.3} \underset{\uparrow 0.8}{\overset{a}{3}} \xrightarrow{0.6} F \quad 0.000065$$

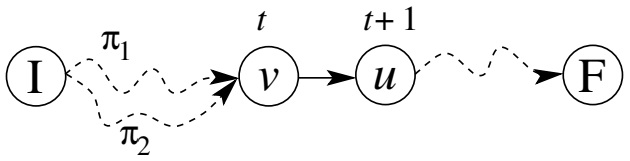
HMMs: Forward-Backward algorithm

Another interpretation of numerator in (4):



$$P(x, \pi_1)$$

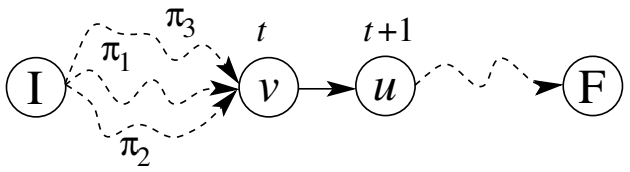
Another interpretation of numerator in (4):



$$P(x, \pi_1) + P(x, \pi_2)$$

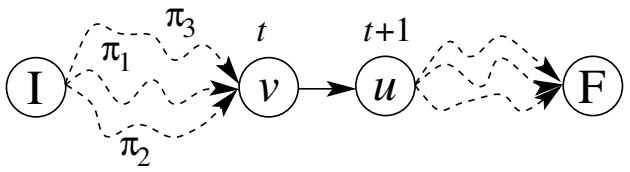
HMMs: Forward-Backward algorithm

Another interpretation of numerator in (4):



$$P(x, \pi_1) + P(x, \pi_2) + P(x, \pi_3)$$

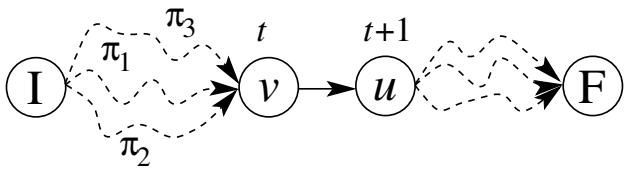
Another interpretation of numerator in (4):



$$P(x, \pi_1) + P(x, \pi_2) + P(x, \pi_3) + \cdots$$

HMMs: Forward-Backward algorithm

Another interpretation of numerator in (4):



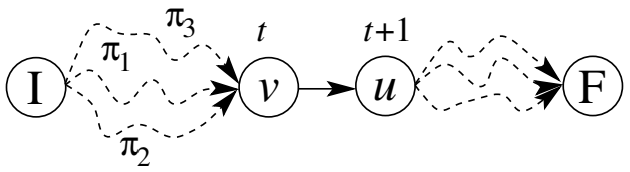
$$P(x, \pi_1) + P(x, \pi_x) + P(x, \pi_3) + \cdots$$

For the parameter $q(u|v)$:

$$\sum_t \sum_{\Psi_t(v,u,x_t)} \underbrace{p(x_1^t, s_t = v)}_{\alpha_t(v)} \cdot \underbrace{\Psi_t(s_t = v, s_{t+1} = u, x_{t+1})}_{q(u|v) \cdot e(x_{t+1}|u)} \cdot \underbrace{p(x_{t+2}^T \mid s_{t+1} = u)}_{\beta_{t+1}(u)}$$

HMMs: Forward-Backward algorithm

Another interpretation of numerator in (4):



$$P(x, \pi_1) + P(x, \pi_x) + P(x, \pi_3) + \cdots$$

For the parameter $q(u|v)$:

$$\sum_t \sum_{\Psi_t(v,u,x_t)} \underbrace{p(x_1^t, s_t = v)}_{\alpha_t(v)} \cdot \underbrace{\Psi_t(s_t = v, s_{t+1} = u, x_{t+1})}_{q(u|v) \cdot e(x_{t+1}|u)} \cdot \underbrace{p(x_{t+2}^T \mid s_{t+1} = u)}_{\beta_{t+1}(u)}$$

Therefore equation (2) for $q(u|v)$ is:

$$\overline{q}(u|v) = \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{t=0}^{T} \alpha_t(v) \cdot q(u|v) \cdot e(x_{t+1}|u) \cdot \beta_{t+1}(u)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_t \alpha_t(u) \cdot \beta_t(u)}$$

(Transitions to the final state require a slight modification)

For the parameter $e(b \mid u)$:

$$\sum_{\substack{t=1 \\ x_t = b}}^{T} \sum_{u} \underbrace{p(x_1^t, s_t = u)}_{\alpha_t(u)} \cdot \underbrace{p(x_t^T \mid s_t = u)}_{\beta_t(u)}$$

For the parameter $e(b \mid u)$:

$$\sum_{\substack{t=1 \\ x_t = b}}^{T} \sum_{u} \underbrace{p(x_1^t, s_t = u)}_{\alpha_t(u)} \cdot \underbrace{p(x_t^T \mid s_t = u)}_{\beta_t(u)}$$

Therefore equation (2) for $e(b|u)$ is:

$$\overline{e}(b|u) = \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{t=1;\ x_t=b}^{T} \alpha_t(u) \cdot \beta_t(u)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{t=1}^{T} \alpha_t(u) \cdot \beta_t(u)} \tag{5}$$

Time complexity: $O(|x| \ |\mathcal{Q}|)$

Forward algorithm

➢ **Definition**: Score for $x_1 \dots x_t$ arriving to $\pi_t = s$

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\pi_1^t;\ \pi_t=s, \pi_0=I} \prod_{i=0}^{t} \Psi_i\left(s_i,\ s_{i+1},\ x_{i+1}\right)$$

➢ **Initialization**:

$$\alpha_0(I) \ = \ 1$$

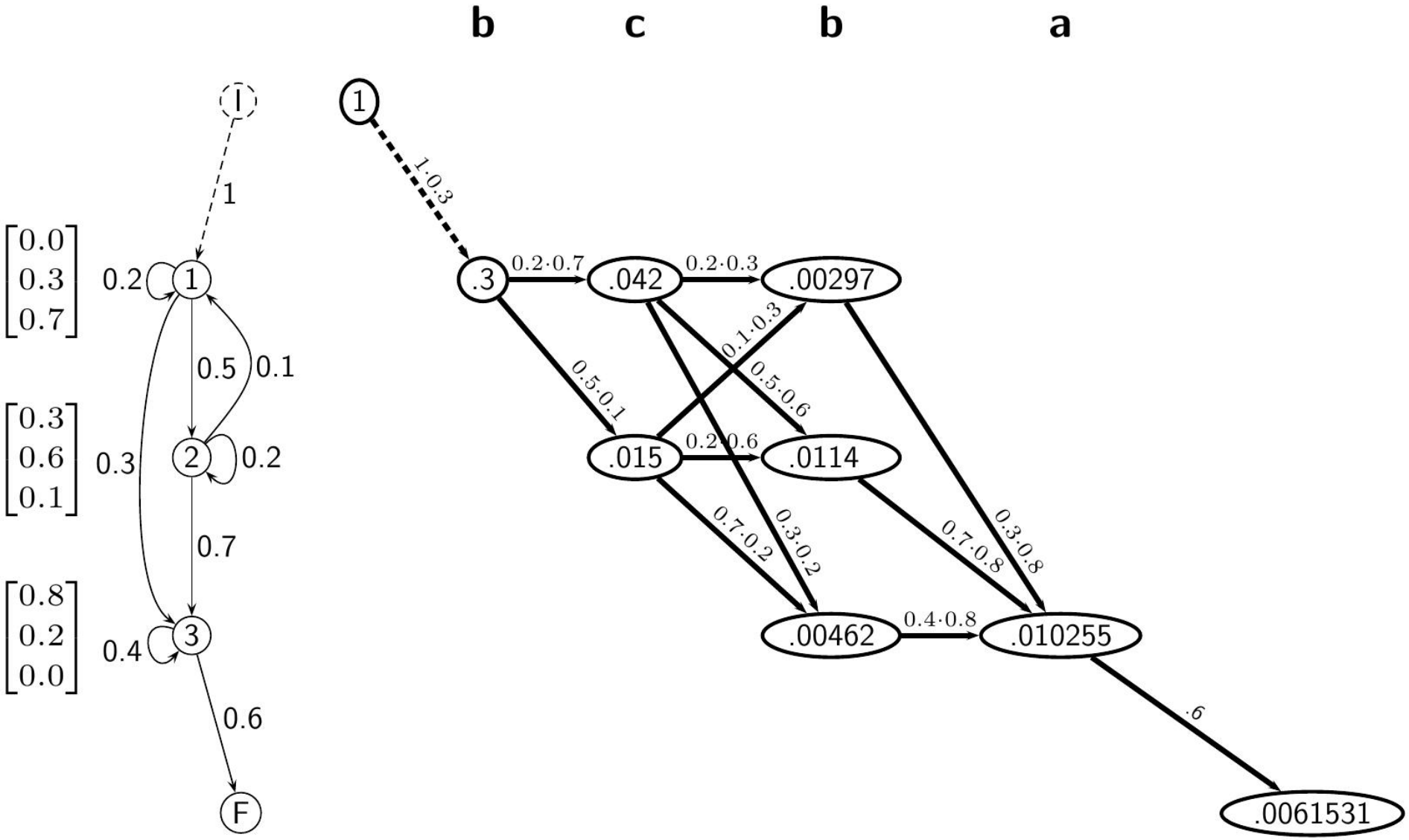
➢ **Recursion**: $\forall\, i = 1 \dots T;$ and $\forall\, s$

$$\alpha_t(s) \ = \ \sum_{s'} \alpha_{t-1}(s') \ \cdot \ \Psi_{t+1}\left(s',\ s,\ x_t\right)$$

➢ **Final result**:

$$\alpha_{T+1}(F) \ = \ \sum_{s'} \alpha_T(s') \ \cdot \ q(F|s')$$

Example: forward computation



Backward algorithm

➢ **Definition**: Score for $x_{t+1} \dots x_T$ starting at $y_t = s$

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\pi_{t+1}^T;\, \pi_t = s} \prod_{i=t+1}^{T} \Psi_i\left(\pi_{i-1},\ \pi_i,\ x_i\right)$$

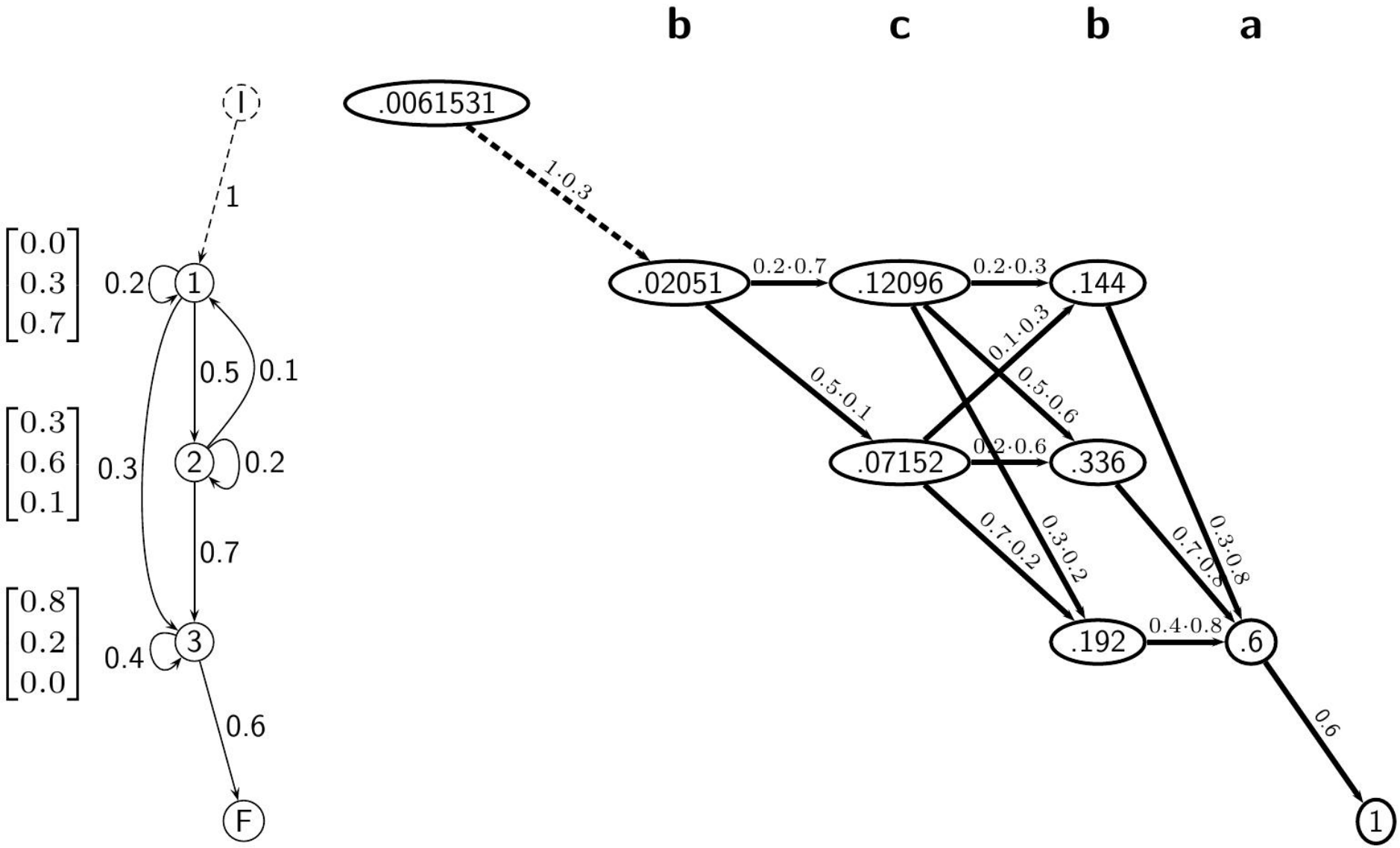
➢ **Initialization**: $\forall\ s$ and F

$$\beta_{T+1}(F) \ = \ 1 \qquad\qquad \beta_T(s) \ = \ q(F|s)$$

➢ **Recursion**: $\forall\ i = T-1 \dots 0;$ and $\forall\ s$

$$\beta_t(s) \ = \ \sum_{s'} \Psi_{t+1}\left(s,\ s',\ x_{t+1}\right) \ \cdot \ \beta_{t+1}(s')$$

Example: backward computation



HMMs: Practice

This practical part is devoted to develop a simple parser of bibligraphic cites. The parser is based on hidden Markov models (HMM). It accepts a bibligraphic cite and generates a bib entry with the tokens distributed among the corresponding fields (author, title, etc.). The parser doest not distiguish among different types of entries (@book, @article, etc.).

The student has a toolkit that has been developed for this purpose. The toolkit includes a program for estimating a HMM with the forward-backward algorithm and with the Viterbi algorithm. The program also performs the parsing by accepting bibliographic cites and generating bib entries. For example, given the following cite:

```
J. Kupiec Hidden Markov Estimation for Unrestricted Stochastic Context-Free Grammars Proc. of
ICASSP'92  Vol. 1 1992 177 180
```

the output should be

```
@inproceedings{Kupiec92,
  author =       {J. Kupiec},
  title =        {Hidden Markov Estimation for Unrestricted Stochastic Context-Free Grammars},
  booktitle =    {Proc. of ICASSP'92 Vol. 1},
  year =         1992,
  pages =        {177--180}
}
```

HMMs: Practice

Tools:

- `m0`: initial model to be trained
- `trainHMM`: training samples (punctuation marks have been removed)
- `testHMM`: test samples (punctuation marks have been removed)
- `nameStates`: mapping between states of the HMM and type of entries in a bibligraphic cite
- `hmm2dot.sh`: plot a model
- `hmml`: training tool. Options (if option -D is used, then options -v, -o, and -I are ignored):

```
Usage: hmml [-h] -i mI [-S sF] [-l sZ] [[-v] [-o mO] [-I ite] [-s sm]] | [-D]] [-V n]
-i mI input model file
-S sF sample file (stdin by default)
-l sZ use strings up to length sZ (60 by default)
-v Viterbi learning (Bawm-Welch by default)
-o mO output model file (mI by default)
-I ite number of traing iterations (1 by default)
-s sm smooth value for emission probabilities (float, not by default)
-D decoding (useless in training mode)
-V n verbosity level (0 by default, 3 max)
-h this help
```

Example: `hmml -i m0 -S trainHMM -o m1 -I 3 -s 0.00001` $\longrightarrow$ training

`hmml -i m0 -S testHMM -D` $\longrightarrow$ decoding

The output is:

```
J.@0 Kupiec@0 Hidden@1 Markov@1 Estimation@1 for@1 Unrestricted@1 Stochastic@1 Context-Free@1
Grammars@1 Proc.@4 of@4 ICASSP'92@4 Vol.@4 1@8 1992@2 177@8 180@8
```

the token before the symbol `@` corresponds to the input and the integer after the symbol `@` corresponds to the state in which the token has been observed. These integers are mapped into bibligraphic fields as indicated in the `nameStates` file. For example `J.@0 Kupiec@0` means that tokens `J. Kupiec` have been generated in state `author`.

Question

File `testHMM` contains two additional cites. You have to parse these cites with `hmml` with model `m0` and write the output in bib format as in the previous examples.

Question

Train model `m0` as in the previous example with 10 iterations. Then, parse the same cites included in `testHMM` with model `m1` and write the output in bib format as in the previous examples. Justify the changes with respect to the previous exercise.

Question

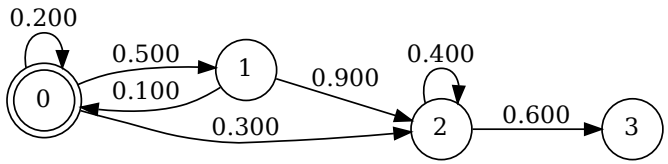
The model in first question can be written in a file as:

```
HMM 4 3
1.0 0.0 0.0
0.2 0.5 0.3 0.0
0.1 0.0 0.9 0.0
0.0 0.0 0.4 0.6
0.0 0.3 1.0 0.0 a
0.3 0.6 0.0 0.0 b
0.7 0.1 0.0 0.0 c
```

Then, the HMM we be can plot (only the transitions) as follows:

```
hmm2dot.sh t0 > t0.dot
dot -Tpdf t0.dot -o t0.pdf
```

and we get



You have to plot the HMMs obtained in first question and and then in second question after the training process. For this process you have the scripts `hmm2dot.sh` and `hmm2dotWithLabels.sh`.

HMMs: Practice

Complete/execute the following steps and have a look to intermediate files:

1. `./prepareBib.sh teds_bib.bib > refs.bib` ⟵ prepare the bib file
2. `./createMapFile.sh refs.bib > tabLabels` ⟵ create map label file
3. `./prepareData4HMM.sh refs.bib tabLabels > train` ⟵ prepare train data
4. `./createHMM.sh refs.bib > hmm0` ⟵ create initial HMM
5. `./hmml -i hmm0 -S train -o hmm1 -I 10 -s 0.00001` ⟵ train
6. `./hmml -i hmm1 -S train -D` ⟵ decode

Question

In the previous steps, the training set is used as test. Remove some samples from the training and used them for testing.

Question

Define an evalution method and perform a complete experiment.

PCFGs: Inside-Outside algorithm

Given a PCFG $(N, \Sigma, S, \mathcal{P}, \theta)$ in Chomsky Normal Form, where θ is a vector of parameters that contains two types of parameters for the two types of rules:

$$p(A \to BC) : \ A, B, C \in N \quad \text{and} \quad p(A \to b) : \ A \in N \ b \in \Sigma$$

PCFGs: Inside-Outside algorithm

Given a PCFG $(N, \Sigma, S, \mathcal{P}, \theta)$ in Chomsky Normal Form, where θ is a vector of parameters that contains two types of parameters for the two types of rules:

$$p(A \to BC): \ A, B, C \in N \quad \text{and} \quad p(A \to b): \ A \in N \ b \in \Sigma$$

Then, Eq (2) becomes (see [Benedí 05]) :

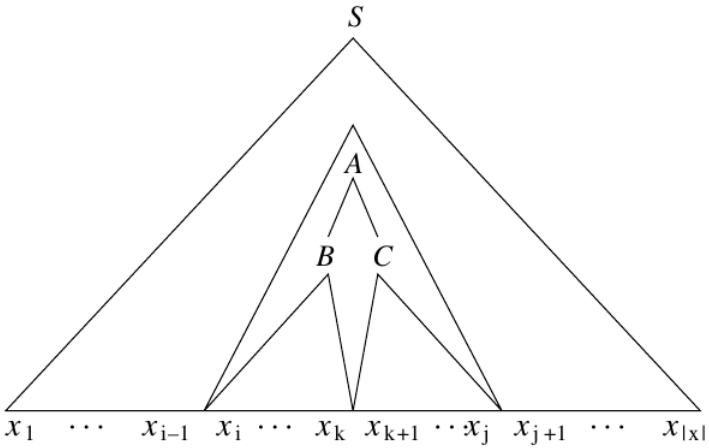
$$\overline{p}(A \to BC) = \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{\forall d_x} \mathrm{N}(A \to BC, d_x) P_\theta(x, d_x)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{\forall d_x} \mathrm{N}(A, d_x) P_\theta(x, d_x)},$$

$$\overline{p}(A \to b) = \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{\forall d_x} \mathrm{N}(A \to b, d_x) P_\theta(x, d_x)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{\forall d_x} \mathrm{N}(A, d_x) P_\theta(x, d_x)},$$

Note that the number of derivation trees can be exponential.

PCFGs: Inside-Outside algorithm

Let $\Delta_{x,i,j,k,A\to BC}$ be the subset of derivations of x in which the rule $A \to BC$ appears delimited by positions i, j, k



For the parameters $p(A \to BC)$, we can express $P_\theta(x)$ as:

$$\sum_{\forall d_x} \mathrm{N}(A \to BC, d_x) P_\theta(x, d_x)$$

$$= \sum_{A\to BC} \sum_{l=2}^{T} \sum_{i=0}^{T-l} \sum_{k=1}^{l-1} \underbrace{p(S \overset{*}{\Rightarrow} x_1^i \; A \; x_{i+l+1}^T)}_{f(A,i,i+l)} \cdot \underbrace{p(B \overset{*}{\Rightarrow} x_{i+1}^{i+k})}_{e(B,i,i+k)} \cdot \underbrace{p(C \overset{*}{\Rightarrow} x_{i+k+1}^{i+l})}_{e(C,i+k,i+l)} \cdot p(A \to BC)$$

Therefore equation (2) for $p(A \rightarrow BC)$ is (see [Benedí 05] for alternative interpretation):

$$\overline{p}(A \rightarrow BC) =$$

$$\frac{\sum_{x \in \mathcal{D}} \frac{p(A \rightarrow BC)}{P_\theta(x)} \sum_{l=2}^{T} \sum_{i=0}^{T-l} \sum_{k=1}^{l-1} f(A, i, i+l) \cdot e(B, i, i+k) \cdot e(C, i+k, i+l)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{l=1}^{T} \sum_{i=0}^{T-l} f(A, i, i+l) \cdot e(A, i, i+l)} \quad (6)$$

PCFGs: Inside-Outside algorithm

For the parameter $p(A \to b)$, we can express $P_\theta(x)$ as:

$$\sum_{A \to b} \sum_{i=0}^{T-1} \underbrace{p(S \overset{*}{\Rightarrow} x_1^i \; A \; x_{i+2}^T)}_{f(A,i,i+1)} \cdot p(A \to b) \cdot \delta(b, x_{i+1})$$

Therefore equation (2) for $p(A \to b)$ is:

$$\overline{p}(A \to b) \quad = \quad \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{i=0}^{T-1} f(A, i, i+1) \cdot p(A \to b) \cdot \delta(b, x_{i+1})}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{l=1}^{T} \sum_{i=0}^{T-l} f(A, i, i+l) \cdot e(A, i, i+l)} \tag{7}$$

Time complexity: $O(|x|^3 \; |\mathcal{P}|)$ [Baker 79, Lari and Young 90, Benedí 05]

Outside ALGORITHM

➢ **Definition**: Given $x = x_1 \dots x_T \in \Sigma^*$ and $A \in N$

$$f(A, i, j) \;=\; P_{G_p}(S \stackrel{*}{\Rightarrow} x_1 \dots x_i \; A \; x_{j+1} \dots x_T)$$

Outside ALGORITHM

➢ **Definition**: Given $x = x_1 \dots x_T \in \Sigma^*$ and $A \in N$

$$f(A, i, j) \;=\; P_{G_p}(S \stackrel{*}{\Rightarrow} x_1 \dots x_i \; A \; x_{j+1} \dots x_T)$$

➢ **Initialization**: $\forall A \in N$

$$f(A, 0, T) = \delta(A, S)$$

➢ **Recursion**: $\forall\, A \in N; \qquad \forall\, l : 1 \dots T; \qquad \forall\, i : 0 \dots T - l;$

$$f(A, i, i+l) \;=\; \sum_{B,C \in N} p(B \to CA) \sum_{k=1}^{i} f(B, i-k, i+l) \;\cdot\; e(C, i-k, i)$$

$$+ \sum_{B,C \in N} p(B \to AC) \sum_{k=1}^{T-i-l} f(B, i, i+l+k) \;\cdot\; e(C, i+l, i+l+k >)$$

➢ **Final result**: The sentence probability is:

$$P_{G_p}(x) \;=\; \sum_{A \in N} f(A, i, i+1) \;\cdot\; p(A \to x_{i+1}) \qquad\qquad 0 \le i \le T-1$$

Optimization function: Given a statistical model θ, and a training sample $\mathcal{D}$

$$\log\ P_\theta(\mathcal{D}, \widehat{t}_{\mathcal{D}}) \ = \ \log \prod_{x \in \mathcal{D}} P_\theta(x, \widehat{t}_x) \ = \ \sum_{x \in \mathcal{D}} \log\ P_\theta(x, \widehat{t}_x)$$

Optimization function: Given a statistical model θ, and a training sample $\mathcal{D}$

$$\log\ P_\theta(\mathcal{D}, \widehat{t}_{\mathcal{D}}) \ = \ \log \prod_{x \in \mathcal{D}} P_\theta(x, \widehat{t}_x) \ = \ \sum_{x \in \mathcal{D}} \log\ P_\theta(x, \widehat{t}_x)$$

For the parameters $p(A \to \alpha)$, we can express $P_\theta(x, \widehat{t}_x)$ as:

$$P_\theta(x, \widehat{t}_x) \ = \ \prod_{(A \to \alpha)} p(A \to \alpha)^{N(A \to \alpha, \widehat{t}_x)}$$

Optimization function: Given a statistical model θ, and a training sample $\mathcal{D}$

$$\log\ P_\theta(\mathcal{D}, \widehat{t}_{\mathcal{D}}) \ = \ \log \prod_{x \in \mathcal{D}} P_\theta(x, \widehat{t}_x) \ = \ \sum_{x \in \mathcal{D}} \log\ P_\theta(x, \widehat{t}_x)$$

For the parameters $p(A \to \alpha)$, we can express $P_\theta(x, \widehat{t}_x)$ as:

$$P_\theta(x, \widehat{t}_x) \ = \ \prod_{(A \to \alpha)} p(A \to \alpha)^{N(A \to \alpha, \widehat{t}_x)}$$

Therefore equation (2) for $p(A \to \alpha)$ is:

$$\overline{p}(A \to \alpha) \ = \ \frac{\sum_{x \in \mathcal{D}} \ N(A \to \alpha, \widehat{t}_x)}{\sum_{x \in \mathcal{D}} \ N(A, \widehat{t}_x)}.$$

Time complexity: $O(|x|\ \mathcal{P}|)$ [Ney 91, Benedí 05]

Theorem [Sánchez 97]

Let $G_\theta = (G, \theta)$ be a PCFGs, and let $\mathcal{D}$ be a training sample of $L(G)$.

If $\overline{G}_\theta$ is a PCFG obtained by applying the previous transformations,

then the following proprerties are accomplished:

- $\overline{G}_\theta$ is consistent.
- The expected number of times that a symbol from Σ appears in $L(\overline{G}_\theta)$ is equal to the expected number of times that this symbols appears in the sample $\mathcal{D}$.
- The expected length of the strings in $L(\overline{G}_\theta)$ is equal to the expected length of the strings in the sample $\mathcal{D}$.

See [Sánchez and Romero, 20] for additional information about the computation of moments on PFA.

Palindrome Language

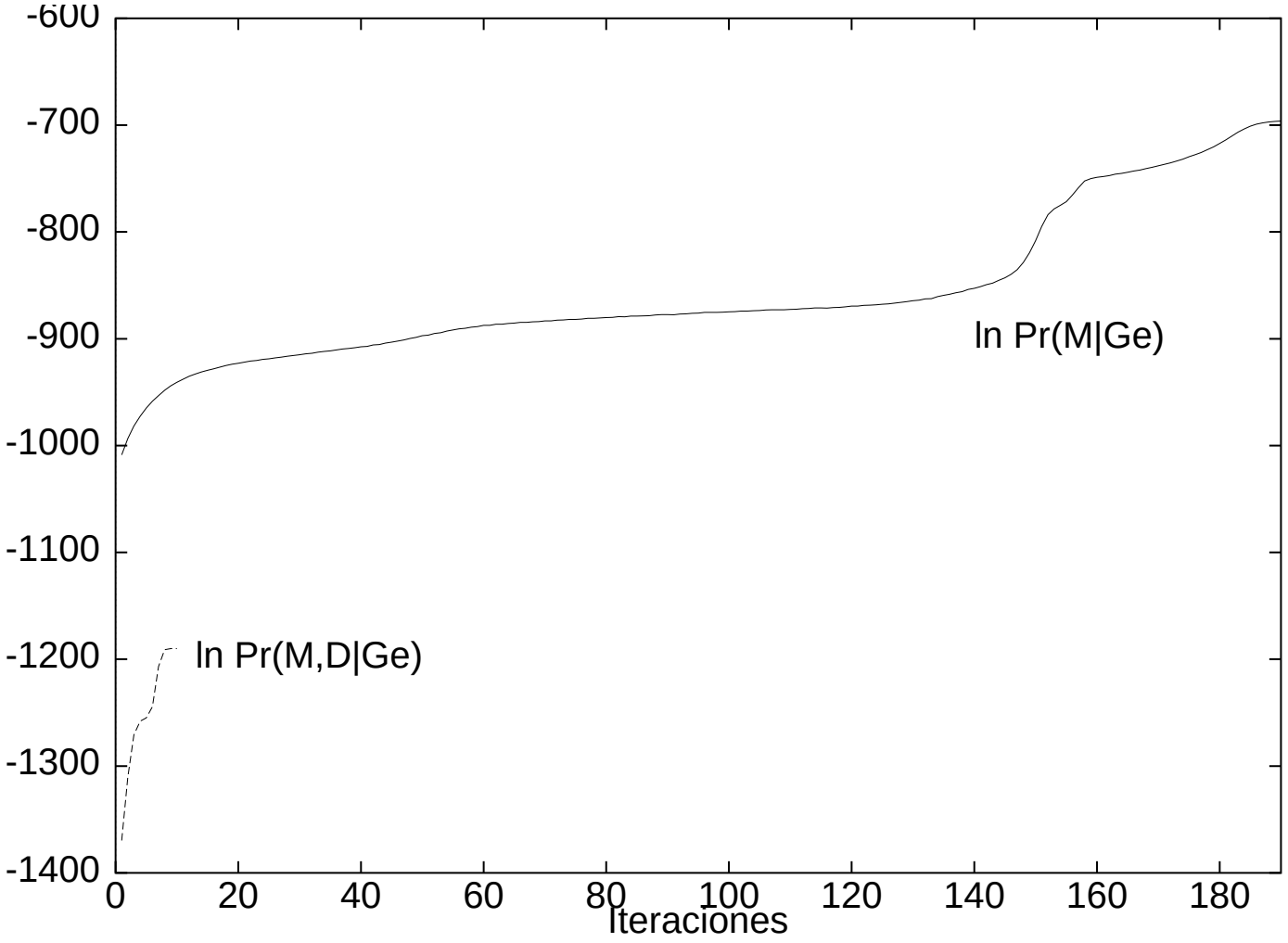
$$\{\, ww^R \mid w \in \{a,b\}^+ \}$$ w^R means the reverse string of w

- Original model

$S \to AC$ 0.4	$S \to BB$ 0.1	$C \to SA$ 1.0	$A \to a$ 1.0
$S \to BD$ 0.4	$S \to AA$ 0.1	$D \to SB$ 1.0	$B \to b$ 1.0

- Training sample: 1000 sentences
- Initial model
  - 5 non-terminal symbols and 2 terminal symbols $\Rightarrow$ 130 rules
  - Rule probabilities were randomly generated

Optimized functions with IO and VS algorithms



Accumulated probability mass

	IO	
iteration	apm	5bd
0	0.002	50.0%
340	0.724	97.4%

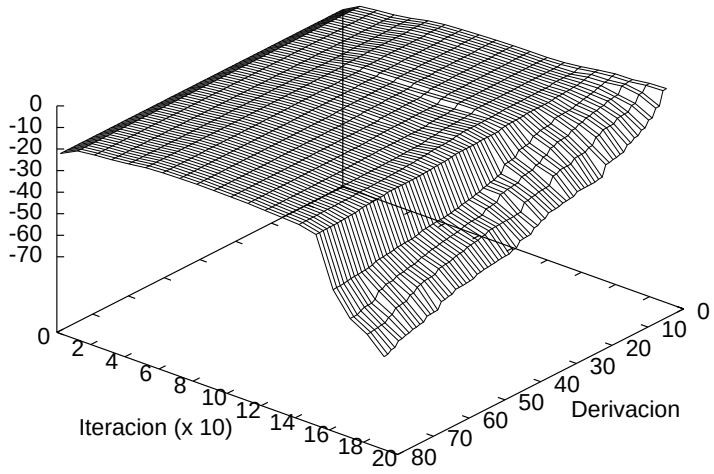
	VS	
iteration	apm	5bd
0	0.002	50.0%
10	0.024	95.7%

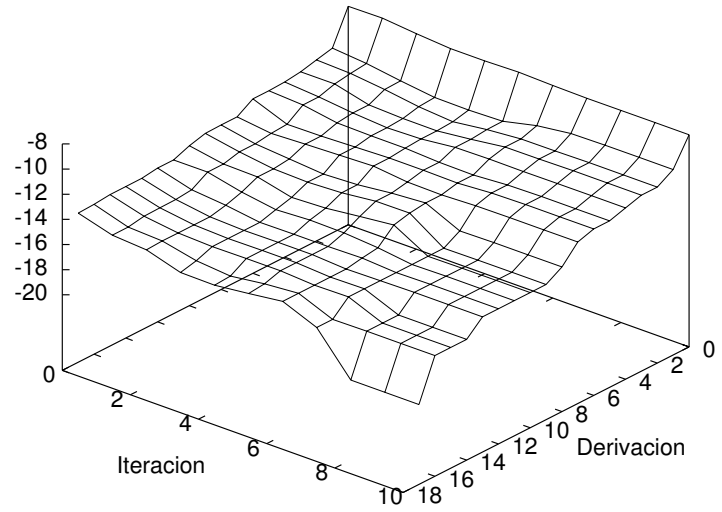
Accumulated probability mass

	IO	
iteration	apm	5bd
0	0.002	50.0%
340	0.724	97.4%

	VS	
iteration	apm	5bd
0	0.002	50.0%
10	0.024	95.7%

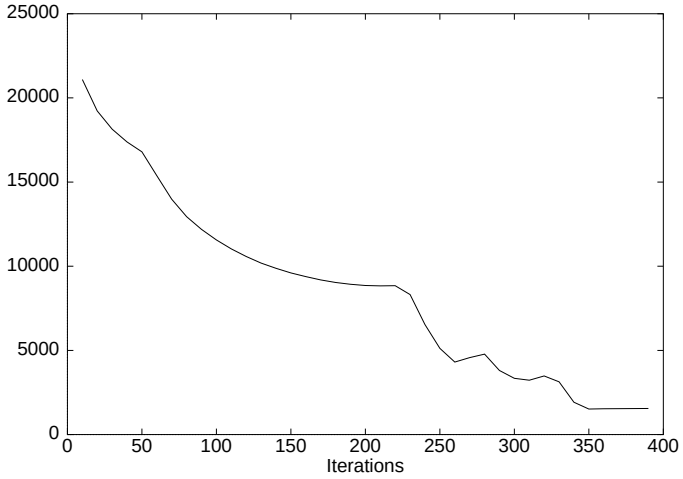
Evolution of the logarithm of the probability



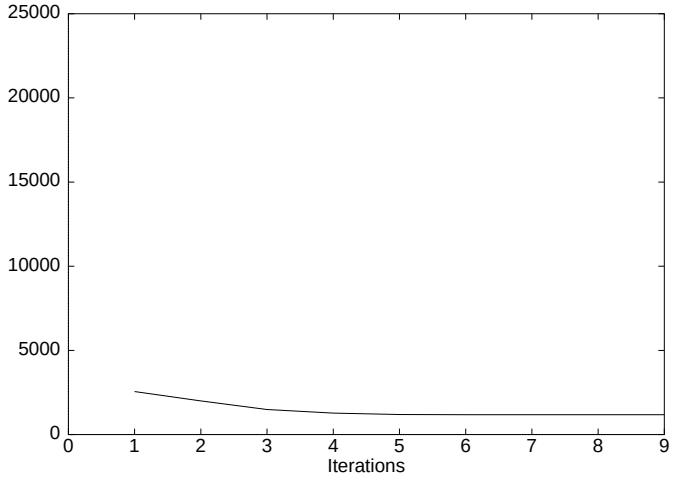


Upper bound estimate

IO



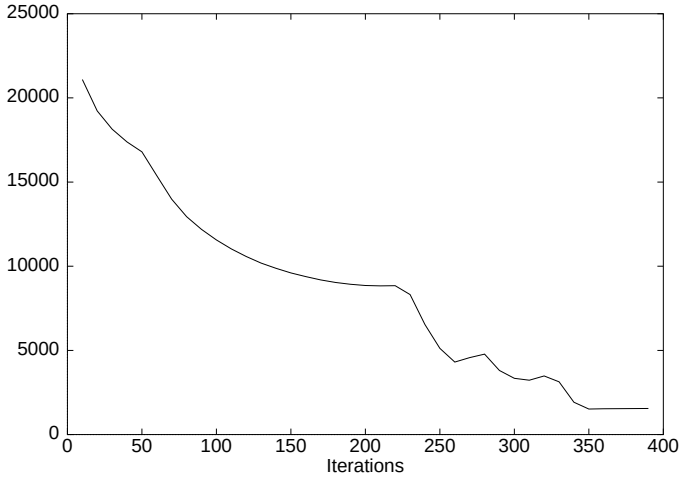
VS



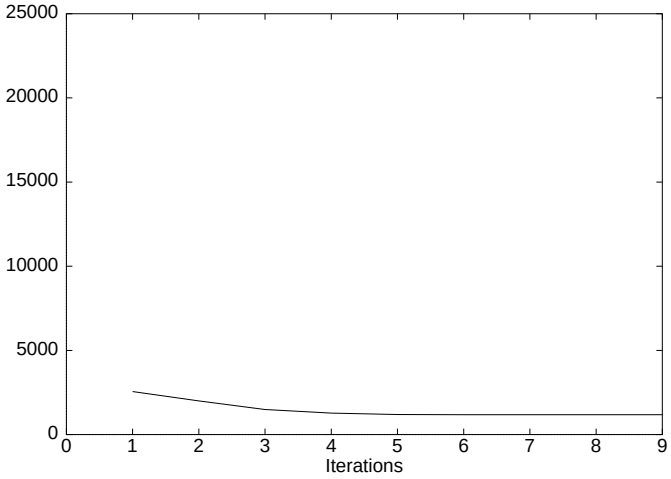
$$\log P_\theta(\mathcal{D}) \; - \; \log P_\theta(\mathcal{D}, \widehat{t}_{\mathcal{D}}) \; \leq \; 2 \sum_{x \in \mathcal{D}} |x| \; \log \; 2 \; |N|$$

Upper bound estimate

IO



VS



$$\log P_\theta(\mathcal{D}) \; - \; \log P_\theta(\mathcal{D}, \widehat{t}_{\mathcal{D}}) \; \leq \; 2 \sum_{x \in \mathcal{D}} |x| \; \log \; 2 \; |N|$$

Evaluation of models

Algorithm	kld	Palindromes (%)	Non palindromes (%)
VS	6.00	1.9	98.1
IO	1.88	76.0	24.0

Optimization function: Given G_θ, and a training sample $\mathcal{D}$

$$\lg P_\theta(\mathcal{D}, \Delta_\mathcal{D}) = \lg \prod_{x \in \mathcal{D}} P_\theta(x, \Delta_x) = \sum_{x \in \mathcal{D}} \lg P_\theta(x, \Delta_x)$$

Optimization function: Given G_θ, and a training sample $\mathcal{D}$

$$\lg\ P_\theta(\mathcal{D}, \Delta_\mathcal{D}) = \lg \prod_{x \in \mathcal{D}} P_\theta(x, \Delta_x) = \sum_{x \in \mathcal{D}} \lg\ P_\theta(x, \Delta_x)$$

For the parameters $p(A \to \alpha)$, we can express $P_\theta(x, \Delta_x)$ as:

$$P_\theta(x, \Delta_x) = \sum_{t_x \in \Delta_x} P_\theta(x, t_x) = \sum_{t_x \in \Delta_x} \prod_{(A \to \alpha)} p(A \to \alpha)^{N(A \to \alpha, t_x)}$$

Optimization function: Given G_θ, and a training sample $\mathcal{D}$

$$\lg\ P_\theta(\mathcal{D}, \Delta_\mathcal{D}) \ = \ \lg \prod_{x \in \mathcal{D}} P_\theta(x, \Delta_x) \ = \ \sum_{x \in \mathcal{D}} \lg\ P_\theta(x, \Delta_x)$$

For the parameters $p(A \to \alpha)$, we can express $P_\theta(x, \Delta_x)$ as:

$$P_\theta(x, \Delta_x) \ = \ \sum_{t_x \in \Delta_x} P_\theta(x, t_x) \ = \ \sum_{t_x \in \Delta_x} \prod_{(A \to \alpha)} p(A \to \alpha)^{N(A \to \alpha,\, t_x)}$$

Therefore equation (2) for $p(A \to \alpha)$ is:

$$\overline{p}(A \to \alpha) \ = \ \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x, \Delta_x)} \sum_{t_x \in \Delta_x} N(A \to \alpha, t_x) \cdot P_\theta(x, t_x)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x, \Delta_x)} \sum_{t_x \in \Delta_x} N(A, t_x) \cdot P_\theta(x, t_x)} \qquad (8)$$

Time complexity: $O(|x|^3\ |\mathcal{P}|)$ [Benedí 05]

➢ kVS algorithm:

1. Obtaining the k-best derivations
2. Applying the transformation

➢ Calculation of k-best derivations [Jiménez 00]

➪ Time complexity

- ➢ Worst case: $O(|x|^3|\mathcal{P}| + k|x|\log k + |x|^3|\mathcal{P}|\log k)$
- ➢ Best case: $O(|x|^3|\mathcal{P}| + k + \frac{|x||\mathcal{P}|}{|N|})$

➪ Space complexity

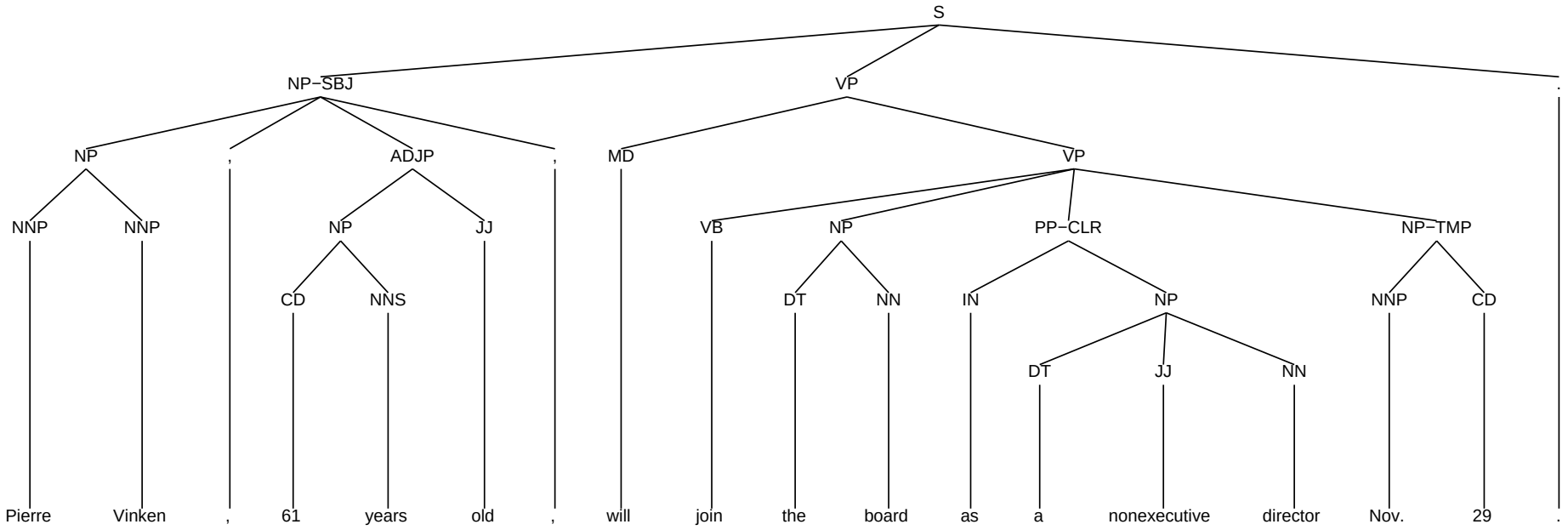
- ➢ $O(k(|x|^2|N| + |x||N|))$

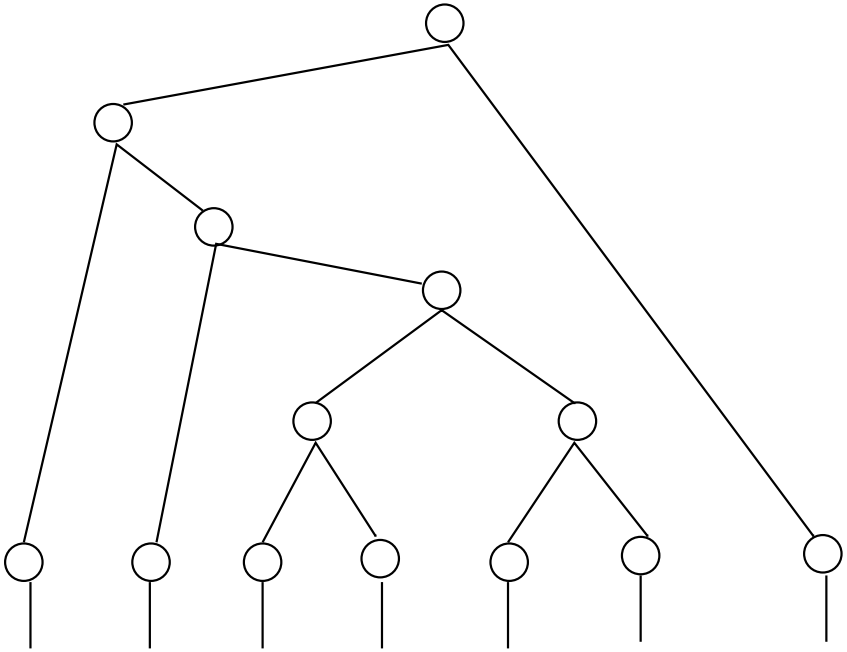
kVS: EXPERIMENTS WITH PALINDROMES

Estimation algorithm	kld	Palindromes		Non palindromes	
		%	apm	%	apm
VS	6.00	1.9	0.023	98.1	0.851
kVS $(k = 2)$	5.76	2.3	0.028	97.7	0.835
kVS $(k = 4)$	5.68	3.7	0.040	96.3	0.822
kVS $(k = 8)$	5.56	7.9	0.074	92.1	0.800
kVS $(k = 16)$	5.45	7.8	0.076	92.2	0.793
kVS $(k = 32)$	5.07	13.2	0.111	86.8	0.696
kVS $(k = 64)$	5.03	12.7	0.112	87.3	0.691
kVS $(k = 128)$	4.98	11.2	0.095	88.8	0.685
kVS $(k = 256)$	4.95	11.9	0.119	88.1	0.637
kVS $(k = 512)$	3.59	17.6	0.156	82.4	0.372
kVS $(k = 1024)$	3.59	17.9	0.156	82.1	0.373
kVS $(k = 2048)$	3.59	17.9	0.156	82.1	0.373
IO	1.88	76.0	0.710	24.0	0.044

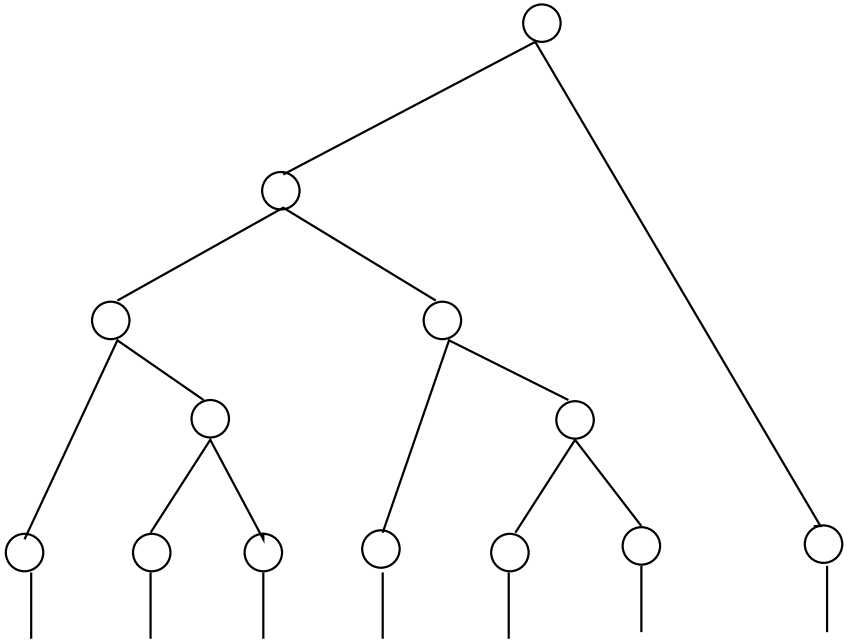
Example: *"Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."*

((S (NP-SBJ (NP (NNP Pierre) (NNP Vinken)) (,,) (ADJP (NP (CD 61) (NNS years)) (JJ old)) (,,)) (VP (MD will) (VP (VB join) (NP (DT the) (NN board)) (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director))) (NP-TMP (NNP Nov.) (CD 29)))) (..)))

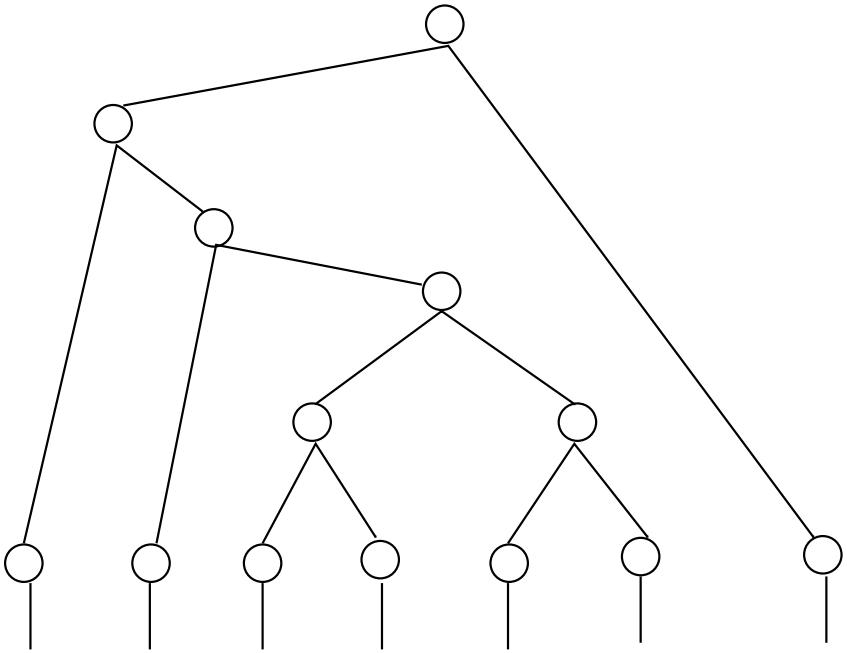




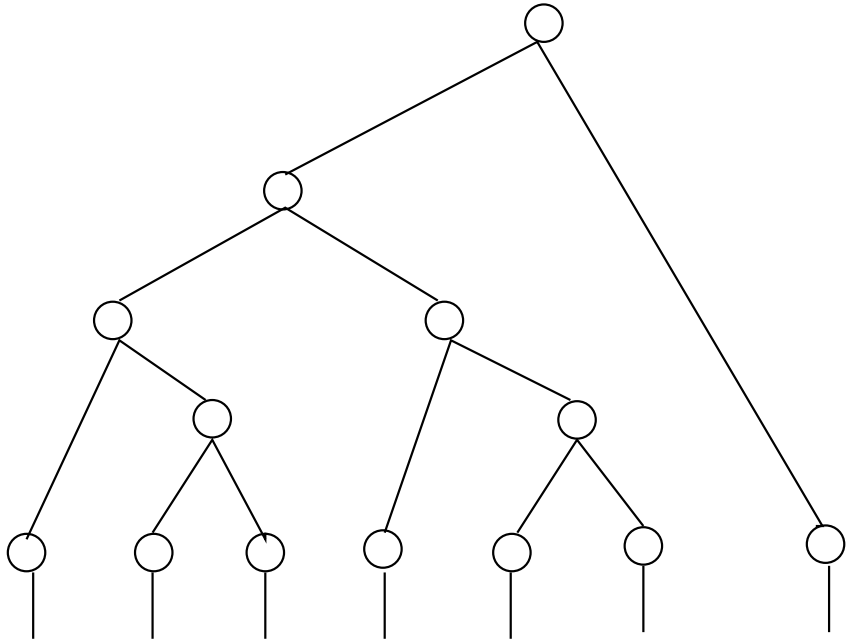
(There (were (many pioneer PC contributors)) .)



(There (were (many pioneer PC contributors)) .)



(There (were (many pioneer PC contributors)) .)



(There (were (many pioneer PC contributors)) .)

From a set of bracketed corpus, the following function is defined [Pereira 92]:

$$\overline{c}(i,j) = \begin{cases} 1 & \text{if } (i,j) \text{ is consistent with the bracketed sample,} \\ 0 & \text{otherwise} \end{cases}$$

(Bracketed) *Viterbi* algorithm [Pereira 92]

$\forall A \in N$, we compute:

$$\widehat{e}^{(b)}(A, i, i+1) = p(A \to b) \cdot \delta(b, x_{i+1}) \qquad 0 \leq i < T-1$$

$$\widehat{e}^{(b)}(A, i, i+l) = \overline{c}(i, i+l) \max_{B,C \in N} \cdot p(A \to BC)$$

$$\max_{k=1,\ldots,l-1} \widehat{e}^{(b)}(B, i, i+k)\, \widehat{e}^{(b)}(C, i+k, i+l)$$

$$2 \leq l \leq T;\ 0 \leq i \leq T-l$$

$$\widehat{P}_\theta(x) = \widehat{e}^{(b)}(S < 0, n >).$$

IOb: experiments with palindromes

Estimation algorithm	kld	Palindromes %	Palindromes apm	Non palindromes %	Non palindromes apm
VSb	4.49	17.1	0.159	82.9	0.600
kVSb $(k = 2)$	4.96	12.8	0.119	87.2	0.633
kVSb $(k = 4)$	4.49	16.8	0.159	83.2	0.600
kVSb $(k = 8)$	4.47	18.2	0.158	81.8	0.591
kVSb $(k = 16)$	4.45	17.6	0.162	82.4	0.584
kVSb $(k = 32)$	4.43	16.9	0.170	83.1	0.578
kVSb $(k = 64)$	4.41	17.5	0.174	82.5	0.572
kVSb $(k = 128)$	3.93	22.5	0.200	77.5	0.564
kVSb $(k = 256)$	3.80	31.6	0.303	68.4	0.442
kVSb $(k = 512)$	3.23	27.1	0.264	72.9	0.299
kVSb $(k = 1024)$	3.59	17.6	0.157	82.4	0.370
kVSb $(k = 2048)$	1.88	74.7	0.715	25.3	0.049
IOb	0.00	100.0	0.746	0.0	0.000

EXPERIMENTS WITH WSJ CORPUS

➢ Characteristics of WSJ corpus:

Data set	Directories	No. of senten.	No. of words
Training (full)	00-20	42,075	1,004,073
Training (≤ 50)	00-20	41,315 (98,2%)	959,390 (95,6%)
Tuning	21-22	3,371	80,156
Test	23-24	3,762	89,537

➢ Vocabulary (Training), only the 10.000 most frequent words

➢ Category-based PCFGs (length restriction ≤ 50):

$\|N\|$	(size)	VS		kVS ($k=128$)		VSb		kVSb ($k=128$)		IOb	
		tsp	size	tsp	size	tsp	size	tsp	size	tsp	size
14	(3,374)	20.13	195	18.56	195	21.10	256	19.53	256	13.34	471
20	(8,900)	20.42	255	17.46	280	19.38	406	18.01	311	12.33	888
25	(16,750)	17.58	407	16.76	349	16.93	499	16.89	398	11.86	1042
30	(28,350)	17.67	367	16.34	400	17.90	609	16.78	424	10.86	1454
35	(44,450)	16.92	600	16.02	454	16.64	789	16.24	619	10.24	1741

The goal of supervised (or predictive) learning is to learn a mapping from inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$, given a fully annotated training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$.

The goal of supervised (or predictive) learning is to learn a mapping from inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$, given a fully annotated training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$.

➢ Objective function: Maximum likelihood (generative) estimation [Bahl et al. 1983]

$$\log \prod_{n=1}^{N} P_\theta(x^{(n)}, y^{(n)}) = \sum_{n=1}^{N} \log P_\theta(x^{(n)}, y^{(n)})$$

The goal of supervised (or predictive) learning is to learn a mapping from inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$, given a fully annotated training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$.

➢ Objective function: Maximum likelihood (generative) estimation [Bahl et al. 1983]

$$\log \prod_{n=1}^{N} P_\theta(x^{(n)}, y^{(n)}) = \sum_{n=1}^{N} \log P_\theta(x^{(n)}, y^{(n)})$$

where

$$P_\theta(x^{(n)}, y^{(n)}) = \prod_{\theta_{ij}} \theta_{ij}^{N(\theta_{ij}, (x^{(n)}, y^{(n)}))}$$

The goal of supervised (or predictive) learning is to learn a mapping from inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$, given a fully annotated training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$.

➢ Objective function: Maximum likelihood (generative) estimation [Bahl et al. 1983]

$$\log \prod_{n=1}^{N} P_\theta(x^{(n)}, y^{(n)}) = \sum_{n=1}^{N} \log P_\theta(x^{(n)}, y^{(n)})$$

where

$$P_\theta(x^{(n)}, y^{(n)}) = \prod_{\theta_{ij}} \theta_{ij}^{N(\theta_{ij}, (x^{(n)}, y^{(n)}))}$$

➢ Optimization methods: Growth transformations

$$\overline{\theta}_{ij} = \frac{\sum_{n=1}^{N} \frac{\theta_{ij}}{P_\theta(x^{(n)}, y^{(n)})} \frac{\partial P_\theta(x^{(n)}, y^{(n)})}{\partial \theta_{ij}}}{\sum_{n=1}^{N} \frac{1}{P_\theta(x^{(n)}, y^{(n)})} \sum_k \theta_{ik} \frac{\partial P_\theta(x^{(n)}, y^{(n)})}{\partial \theta_{ik}}} = \frac{\sum_{n=1}^{N} \theta_{ij} N(\theta_{ij}, (x^{(n)}, y^{(n)}))}{\sum_{n=1}^{N} \sum_k \theta_{ik} N(\theta_{ik}, (x^{(n)}, y^{(n)}))} \quad (9)$$

Discriminative Training: Objective functions

➢ Maximum conditional likelihood (discriminative) estimation [Jebara 98]

$$F_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}, y^{(n)})}{P_\theta(x^{(n)})} = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}|y^{(n)})\ P_\theta(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)}|y)\ P_\theta(y)}$$

Discriminative Training: Objective functions

- Maximum conditional likelihood (discriminative) estimation [Jebara 98]

$$F_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}, y^{(n)})}{P_\theta(x^{(n)})} = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}|y^{(n)})\ P_\theta(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)}|y)\ P_\theta(y)}$$

- Maximum mutual information (discriminative) estimation [Bahl 86]
- Minimum classification error [Juang 97]
- Large Margin Estimation [Jiang 04]

Discriminative Training: Objective functions

- Maximum conditional likelihood (discriminative) estimation [Jebara 98]

$$F_\theta(\mathcal{D}) = \prod_{n=1}^{N} P_\theta(y^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}, y^{(n)})}{P_\theta(x^{(n)})} = \prod_{n=1}^{N} \frac{P_\theta(x^{(n)}|y^{(n)})\ P_\theta(y^{(n)})}{\sum_{y \in \theta(x^{(n)})} P_\theta(x^{(n)}|y)\ P_\theta(y)}$$

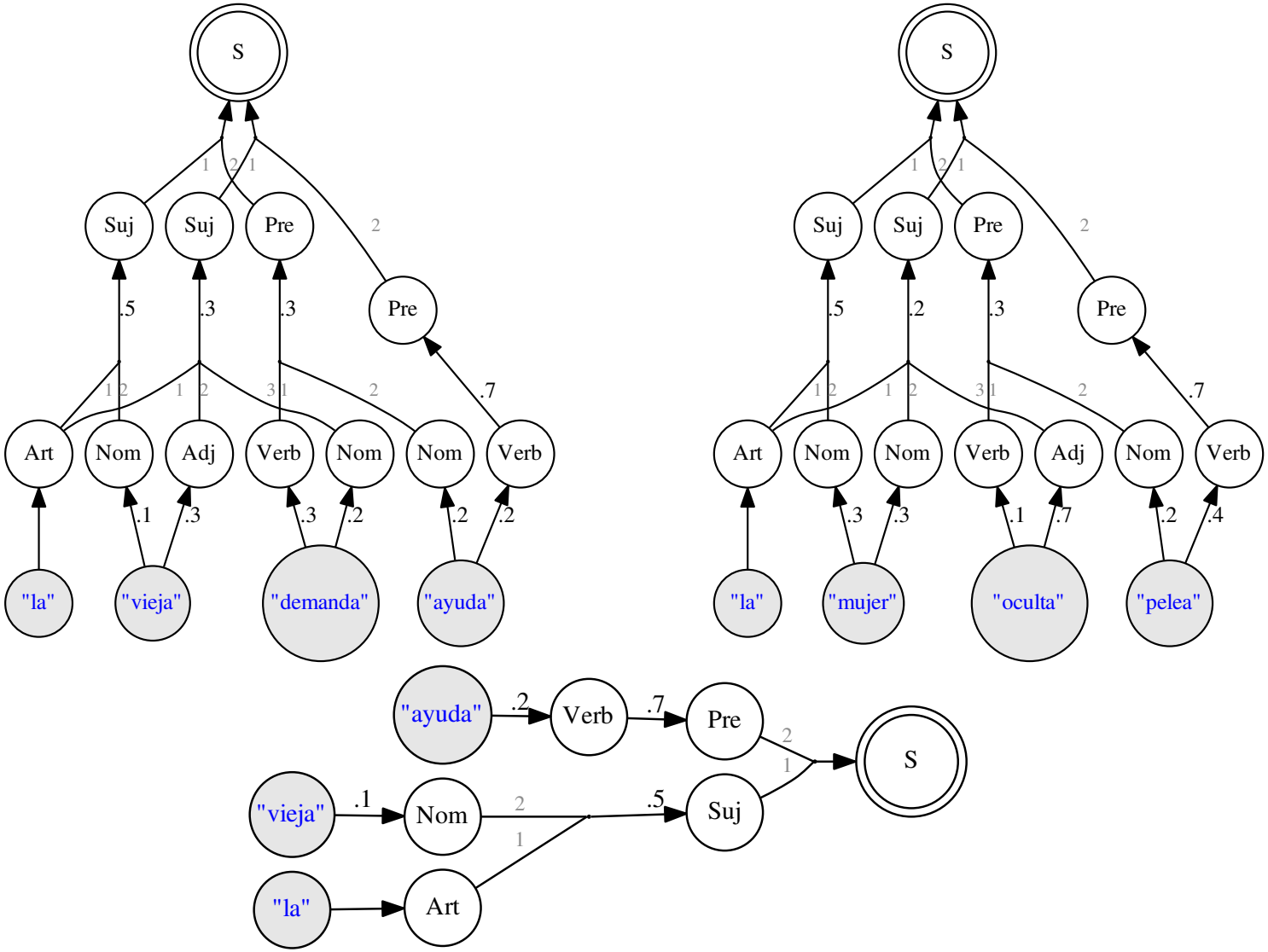
- Maximum mutual information (discriminative) estimation [Bahl 86]
- Minimum classification error [Juang 97]
- Large Margin Estimation [Jiang 04]

Discriminative Training: Optimization methods

- Gradient Descent
- Extended Baum method (Growth transformations) [Gopalakrishnan et al. 1991]

Grammar: (G, θ)

1.0 S → Suj Pre
0.5 Suj → Art Nom
0.3 Suj → Art Adj Nom
0.2 Suj → Art Nom Adj
0.3 Pre → Verb Nom
0.7 Pre → Verb
1.0 Art → "la"
0.1 Nom → "vieja"
0.2 Nom → "ayuda"
0.3 Nom → "mujer"
0.2 Nom → "pelea"
0.2 Nom → "demanda"
0.3 Verb → "demanda"
0.2 Verb → "ayuda"
0.1 Verb → "oculta"
0.4 Verb → "pelea"
0.3 Adj → "vieja"
0.7 Adj → "oculta"



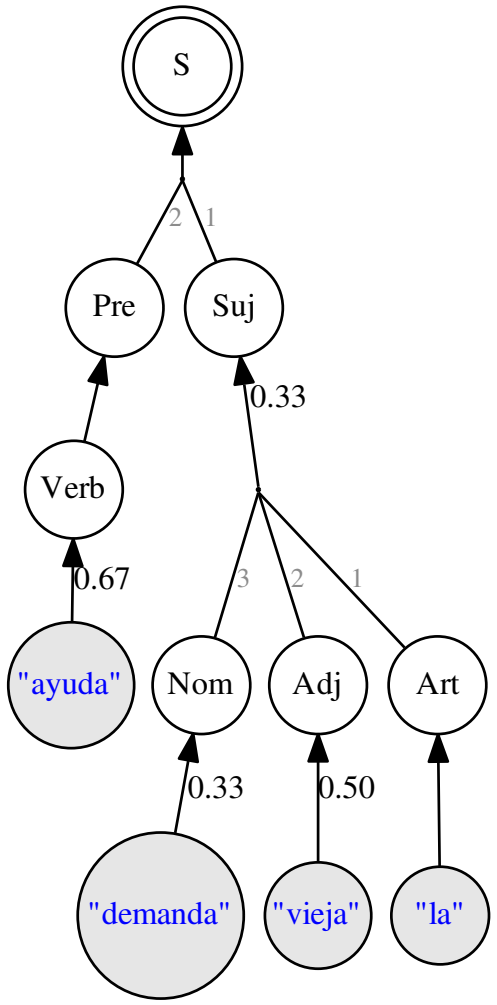
Sample: $\mathcal{D} = \{\text{la vieja demanda ayuda}, \text{la mujer oculta pelea}, \text{la vieja ayuda}\}$

$P_\theta(\text{la vieja demanda ayuda}) = .00090 + .00252 = .00342$
$P_\theta(\text{la mujer oculta pelea}) = .00090 + .01176 = .01266$
$P_\theta(\text{la vieja ayuda}) = .00700$

Inside-Outside:

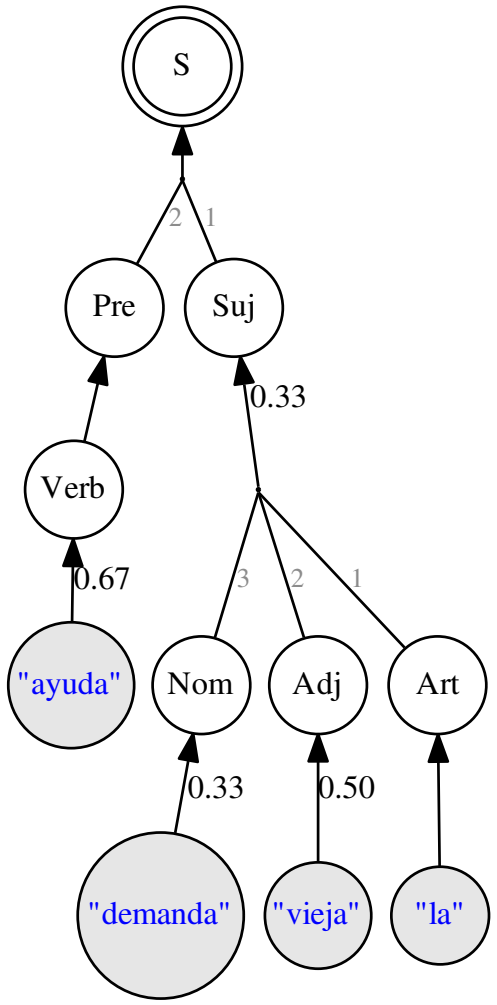
$$\begin{aligned}
&\overline{p}(\text{Suj} \rightarrow \text{Art Nom Adj}) \\
&= \frac{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{t_x} N(\text{Suj} \rightarrow \text{Art Nom Adj}, t_x) P_\theta(x, t_x)}{\sum_{x \in \mathcal{D}} \frac{1}{P_\theta(x)} \sum_{t_x} N(\text{Suj}, t_x)\ P_\theta(x, t_x)} \\
&= \frac{\frac{1}{.01266}\ .01176}{\frac{1}{.00342}\ (.00090 + .00252) + \frac{1}{.01266}\ (.00090 + .01176) + \frac{1}{.00700}\ (.00700)} \\
&= \frac{.9289}{3} = .3096
\end{aligned}$$

Inside-Outside:		θ_1	θ_2	θ_3
1.0 S	→ Suj Pre	1.0000	1.0000	1.0000
0.5 Suj	→ Art Nom	0.4447	0.3370	0.3333
0.3 Suj	→ Art Adj Nom	0.2456	0.3299	0.3333
0.2 Suj	→ Art Nom Adj	0.3096	0.3331	0.3334
0.3 Pre	→ Verb Nom	0.1114	0.0036	
0.7 Pre	→ Verb	0.8886	0.9964	1.0000
1.0 Art	→ "la"	1.0000	1.0000	1.0000
0.1 Nom	→ "vieja"	0.3788	0.3356	0.3333
0.2 Nom	→ "ayuda"	0.0789	0.0034	
0.3 Nom	→ "mujer"	0.2999	0.3322	0.3333
0.2 Nom	→ "pelea"	0.0213	0.0002	
0.2 Nom	→ "demanda"	0.2201	0.3286	0.3334
0.3 Verb	→ "demanda"	0.0877	0.0035	
0.2 Verb	→ "ayuda"	0.5790	0.6632	0.6667
0.1 Verb	→ "oculta"	0.0237	0.0002	
0.4 Verb	→ "pelea"	0.3096	0.3331	0.3333
0.3 Adj	→ "vieja"	0.4423	0.4976	0.5000
0.7 Adj	→ "oculta"	0.5577	0.5024	0.5000
$L_\Theta(\mathcal{D}) =$		$3.0\,10^{-7}$	$8.7\,10^{-6}$	$4.9\,10^{-5}$ $5.1\,10^{-5}$



Viterbi:

		θ_1
1.0 S	→ Suj Pre	1.0000
0.5 Suj	→ Art Nom	0.3333
0.3 Suj	→ Art Adj Nom	0.3333
0.2 Suj	→ Art Nom Adj	0.3334
0.3 Pre	→ Verb Nom	
0.7 Pre	→ Verb	1.0000
1.0 Art	→ "la"	1.0000
0.1 Nom	→ "vieja"	0.3333
0.2 Nom	→ "ayuda"	
0.3 Nom	→ "mujer"	0.3333
0.2 Nom	→ "pelea"	
0.2 Nom	→ "demanda"	0.3334
0.3 Verb	→ "demanda"	
0.2 Verb	→ "ayuda"	0.6667
0.1 Verb	→ "oculta"	
0.4 Verb	→ "pelea"	0.3333
0.3 Adj	→ "vieja"	0.5000
0.7 Adj	→ "oculta"	0.5000



$L_\Theta(\mathcal{D}) = 3.0\,10^{-7}$ $8.7\,10^{-6}$ $4.9\,10^{-5}$ $5.1\,10^{-5}$

- R.Schlüter, W.Macherey, B.Müller, H.Ney (2001) Comparison of discriminative training criteria and optimization methods for speech recognition. Speech Communication (34):287-310
- H.Jiang (2010) Discriminative training of HMMs for automatic speech recognition: A survey. Computer Speech and Language (24):589-608.

- R.Schlüter, W.Macherey, B.Müller, H.Ney (2001) Comparison of discriminative training criteria and optimization methods for speech recognition. Speech Communication (34):287-310
- H.Jiang (2010) Discriminative training of HMMs for automatic speech recognition: A survey. Computer Speech and Language (24):589-608.

- L.R.Bahl, F.Jelinek and R.Mercer (1983) A Maximum Likelihood Approach to Continuous Speech Recognition. IEEE Trans. on PAMI 5(2):179-190.
- T.Jebara, A.Pentland (1998) Maximum conditional likelihood via bound maximization and the CEM algorithm. In: Proceedings of Advances in Neural Information Processing Systems, vol.11.
- L.R.Bahl, P.F.Brown, P.V.De Souza, R.L.Mercer (1986) Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In Proceedings of IEEE (ICASSP'86):49-52.
- B.H.Juang, W.Chou, C.H.Lee (1997) Minimum classification error rate methods for speech recognition. IEEE Trans. on SAP 5(3):257-265.
- H.Jiang (2004) Discriminative training for large margin HMMs. Technical Report CS-2004-01, York University.
- P.S.Gopalakrishnan, D.Kanevsky, A.Nadas, D.Nahamoo (1991) An inequality for rational functions with applications to some statistical estimation problems. IEEE Transactions on Information Theory 37(1):107-113.

PCFG: Practice

This practical exercise is devoted to develop a PCFG for modeling geometric figures, namely, triangles. The toolkit includes a program for estimating a SCFG with the Inside-Outside algorithm, with the Viterbi-Score algorithm, with and without bracketed samples. For example, in the following commands:

```
scfg-toolkit/scfg_cgr -g MODELS/Gm1 -f MODELS/Gm1-new
scfg-toolkit/scfg_learn -g MODELS/Gm1-new -f MODELS/Gm1-new-100 -i 100 -m DATA/D
scfg-toolkit/scfg_learn -g MODELS/Gm1-new -f MODELS/Gm1-new-br-100 -i 100 -m DATA/Dpar
```

the first command creates a new SCFG, the second trains the grammar with the Inside-Outside algorithm with plain samples, and the third command trains the grammar with the Inside-Outside algorithm with bracketed samples. The second command could take a lot of time given the time complexity and should be used taking into account this restriction. Therefore if the sample has many repeated string then it can trained also as follows:

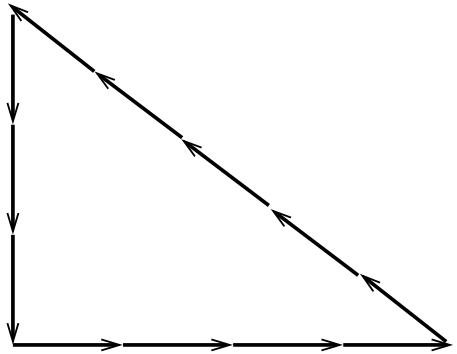
```
sort DATA/D | uniq -c | sed 's/^ \+//g;s/ \+/ /g;' > DATA/D-s-u
scfg-toolkit/scfg_learn -g MODELS/Gm1-new -f MODELS/Gm1-new-100-s-u -i 100 -m DATA/D-s-u -U
```

With the -U option, each uniq string is parsed only once and absolute frequencies are taken into account.

PCFG: Practice

PCFG can be used to represent geometric figures. Consider the following primitives:

$a = \nearrow,\ \ b = \rightarrow,\ \ c = \searrow,\ \ d = \downarrow, e = \swarrow,\ \ f = \leftarrow,\ \ g = \nwarrow,\ \ h = \uparrow .$



Then, a rectangle triangle like the one that can be seen to the right can be represented with the string $bbbbggggggddd$ (left bottom corner is assumed to be the begining of the string).

A dataset `SampleTriangle-10K` representing rectangle triangles is provided. The samples are bracketed as $(bbbb)(gggggg)(ddd)$. A PCFG can be trained and tested as follows:

```
scfg-toolkit/scfg_cgr -g MODELS/G-triangle -f MODELS/Gt-0
sort DATA/SampleTriangle-10K | uniq -c | sed 's/^ \+//g;s/ \+/ /g;' > DATA/SampleTriangle-10K-s-u
scfg-toolkit/scfg_learn -g MODELS/Gt-0 -f MODELS/Gt-100 -i 100 -m DATA/SampleTriangle-10K-s-u -U
scfg-toolkit/scfg_gstr -g MODELS/Gt-100 -c 1000 > tri-test
awk -f scfg-toolkit/checkTriangle tri-test | grep Y | wc -l
```

The last command outputs the number of rectangle triangles in 1000 generated strings. The learning process can take some time. The number of non-terminal symbols in the PCFG is a critical point since the more non-terminal symbols the more flexibility the PCFG has to learn the samples, but it takes more time and more training samples are needed.

Question

Complete the following table:

# non-terminal symbols	# rectangle triangles
5	
10	
15	
20	

PCFG: Practice

Now we will work with three types of triangles: right triangles, equilateral triangles, and isoscalen triangles. These type of triangles exhibit some relations that PCFG may capture. For this purpose, bracketed samples can be used to represent the relations. The following commands perform a simple experiment for given training and test sets:

```
# train models
scfg-toolkit/scfg_learn -g MODELS/G-0 -f MODELS/right -m DATA/Tr-right-s-u -i 100 -U
scfg-toolkit/scfg_learn -g MODELS/G-0 -f MODELS/equil -m DATA/Tr-equil-s-u -i 100 -U
scfg-toolkit/scfg_learn -g MODELS/G-0 -f MODELS/isosc -m DATA/Tr-isosc-s-u -i 100 -U
# classify with the trained models and get results
scfg-toolkit/scfg_prob -g MODELS/right -m DATA/Ts-right > r
scfg-toolkit/scfg_prob -g MODELS/equil -m DATA/Ts-right > e
scfg-toolkit/scfg_prob -g MODELS/isosc -m DATA/Ts-right > i
paste r e i | awk '{m=$1;argm="right"; if ($2>m) {m=$2;argm="equil";}
              if ($3>m) {m=$3;argm="isosc";}printf("right %s\n",argm);}' > results
scfg-toolkit/scfg_prob -g MODELS/right -m DATA/Ts-equil > r
scfg-toolkit/scfg_prob -g MODELS/equil -m DATA/Ts-equil > e
scfg-toolkit/scfg_prob -g MODELS/isosc -m DATA/Ts-equil > i
paste r e i | awk '{m=$2;argm="equil"; if ($1>m) {m=$1;argm="right";}
              if ($3>m) {m=$3;argm="isosc";} printf("equil %s\n",argm);}' >> results
scfg-toolkit/scfg_prob -g MODELS/right -m DATA/Ts-isosc > r
scfg-toolkit/scfg_prob -g MODELS/equil -m DATA/Ts-isosc > e
scfg-toolkit/scfg_prob -g MODELS/isosc -m DATA/Ts-isosc > i
paste r e i | awk '{m=$3;argm="isosc"; if ($1>m) {m=$1;argm="right";}
              if ($2>m) {m=$2;argm="equil";} printf("isosc %s\n",argm);}' >> results
cat results | scfg-toolkit/confus
#      equi isos righ  Err Err%
# equi  923   77    0   77  7.7
# isos  551  220  229  780 78.0
# righ  123  105  772  228 22.8
# Error: 1085/3000 = 36.17%
```

PCFG: Practice

Question

Study the classification results depending on the algorithm used for training (Inside-Outside or Viterbi) and the type of samples (bracketed or not). Analyze the results.

Question

Study the classification results depending on the algorithm used for learning the PCFG and the size of the training data. You can generate more training data as follows:

```
scfg-toolkit/genFig -F 0 -c 1000  -l 2 -L 10 > DATA/Tr-right
```

You have to use a different seed for each training set in order to avoid a repeated sequence of training samples. Analyze the results.

Question

Apply a different classifier to this task and evaluate the results.

➢ Discriminative models: **Conditional Random Fields (CRF)**

`[Lafferty, McCallum, Pereira, 2001].`

$$P_\theta(y|x) = \frac{\exp\{\,\phi(x,y;\theta)\,\}}{Z(x;\theta)}$$

$$= \frac{\exp\{\,\sum_{t=1}^{T}\ \sum_{k=1}^{K}\ \theta_k\ f_k(y_{t-1},\ y_t,\ x_t)\ \}}{Z(x;\theta)} \tag{10}$$

Where

$$Z(x;\theta) = \sum_{y' \in y^*}\ \exp\left\{\ \sum_{t=1}^{T}\ \sum_{k=1}^{K}\ \theta_k\ f_k(y'_{t-1},\ y'_t,\ x_t)\ \right\} \tag{11}$$

CRF: PARAMETER ESTIMATION

- Given $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$, finding the optimal parameters of a CRF:

$$\widehat{\theta} = \arg\max_{\theta} \; L_\theta(\mathcal{D})$$

- Objective function: Conditional log likelihood

$$L_\theta(\mathcal{D}) \quad = \quad \sum_{n=1}^{N} \log \; P_\theta(y^{(n)}|x^{(n)}) \tag{12}$$

- Given $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$, finding the optimal parameters of a CRF:

$$\widehat{\theta} = \arg\max_{\theta} \; L_\theta(\mathcal{D})$$

- Objective function: Conditional log likelihood

$$L_\theta(\mathcal{D}) \;\; = \;\; \sum_{n=1}^{N} \log \; P_\theta(y^{(n)}|x^{(n)}) \tag{13}$$

- Objective function: Regularized log likelihood

$$L_\theta(\mathcal{D}) \;\; = \;\; \sum_{n=1}^{N} \log \; P_\theta(y^{(n)}|x^{(n)}) \; - \; \frac{\lambda}{2} \sum_{k} \; \theta_k^2 \tag{14}$$

The second term is an additional regularization term, with $\lambda > 0$,
to control the trade-off between fitting to data and model complexity

- Substituting the definitions of CRF model (10, 11) into the expression of $L_\theta(\mathcal{D})$ (13)

$$L_\theta(\mathcal{D}) \;=\; \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k \; f_k(y_{t-1}^{(n)}, y_t^{(n)}, x_t^{(n)}) \;-\; \sum_{n=1}^{N} \log \; Z(x^{(n)}; \theta) \quad (15)$$

- Find

$$\widehat{\theta} = \arg\max_{\theta} \; L_\theta(\mathcal{D}) \;-\; \frac{\lambda}{2} \sum_{k} \theta_k^2$$

  - In general there is no analytical solution to this optimization, so we must use a numerical optimization, i.e. gradient-based optimization.
  - It can be shown that $L_\theta(\mathcal{D})$ is a concave function. It means that every local optimum is also a global optimum. Adding regularization ensures that $L_\theta(\mathcal{D})$ is strictly concave, which implies that it has exactly one global optimum [Sutton & McCallum, 2010].

➢ Gradient-based optimization

1 `Initialize:` $\theta = 0$

2 `Compute the gradient:`

$$\delta_k = \frac{\partial\ L_\theta(\mathcal{D})}{\partial \theta_k} \quad \forall k = 1 \dots K$$

3 `Calculate`

$$\widehat{\beta} = \arg\max_{\beta}\ L_{(\theta+\beta\delta)}(\mathcal{D})$$

4 `Move` θ `in steps proportional to the gradient:`

$$\theta = \theta + \widehat{\beta}\ \delta$$

5 `Repeat steps 2 to 4 until convergence`

– The step 3 can be performed using some type of linear search.

– There are many standard packages which use more sophisticated algorithms, and which give considerably faster convergence.

$$\frac{\partial\, L(\theta)}{\partial\, \theta_k} = \sum_{n=1}^{N} \sum_{t=1}^{T} f_k(y_{t-1}^{(n)},\, y_t^{(n)},\, x_t^{(n)}) - \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{y \in \mathcal{Y}^T} f_k(y_{t-1},\, y_t,\, x_t^{(n)})\; P_\theta(y \,|\, x^{(n)}) \tag{16}$$

- The first term is simply the number of times that the feature f_k is equal to 1 on the training data. This term is easy to compute by counting.
- The second term can be interpreted as the expected number of times the feature f_k is equal to 1. This term is more involved to compute.
- For the Regularized log likelihood (14), add the term: $-\lambda\, \theta_k$

$$\sum_{t=1}^{T} \sum_{y \in \mathcal{Y}^T} f_k(y_{t-1},\, y_t,\, x_t^{(n)})\; P_\theta(y \,|\, x^{(n)}) \;=$$

$$\sum_{t=1}^{T} \sum_{s,s' \in \mathcal{Y}} f_k(y_{t-1} = s',\, y_t = s,\, x_t^{(n)}) \sum_{\substack{y \in \mathcal{Y}^T \\ y_{t-1} = s',\, y_t = s}} P_\theta(y \,|\, x^{(n)}) \quad (17)$$

From the definition of CRF (10):

$$P_\theta(y|x) \;=\; \frac{1}{Z(x;\theta)} \prod_{t=1}^{T} \Psi_t(y_{t-1}, y_t, x_t)$$

Where:

$$\Psi_t(y_{t-1}, y_t, x_t) \;=\; \exp\,\{\, \sum_{k=1}^{K} \theta_k\; f_k(y_{t-1}, y_t, x_t)\,\}$$

➢ Forward variables

$$\alpha_t(s) = \sum_{y_1^t;\ y_t=s} \prod_{j=1}^{t} \Psi_j(y_{j-1}, y_j, x_j)$$

➢ Backward variables

$$\beta_t(s) = \sum_{y_{t+1}^T;\ y_t=s} \prod_{j=t+1}^{T} \Psi_j(y_{j-1}, y_j, x_j)$$

➢ The quantity Z

$$Z = \sum_{s} \alpha_T(s)$$

Finally, the marginal distribution needed for the gradient can be computed as:

$$\sum_{\substack{y \in \mathcal{Y}^T \\ y_{t-1}=s',\ y_t=s}} P_\theta(y \,|\, x^{(n)}) \;=\; \alpha_{t-1}(s')\ \Psi_t(y_{t-1}=s', y_t=s, x_t^{(n)})\ \beta_t(s)\ Z^{-1}$$

- Given a training corpus $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$

  Goal: to learn a predictor $x \rightarrow y$ with small error on unseen inputs.

- In CRFs:

$$\begin{aligned}
\arg\max_y \; P_\theta(y|x) &= \arg\max_y \; \frac{\exp\{\sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k \; f_k(y_{t-1}, \; y_t, \; x_t)\}}{Z(x;\theta)} \\
&= \arg\max_y \; \sum_{t=1}^{T} \theta \; f\,(y_{t-1}, \; y_t, \; x_t)
\end{aligned}$$

- Can we learn θ more directly, focusing on errors?

[Collins, 2002]

- Given a training set $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$
- Set $\theta = 0$
- For each iteration $w : 1 \dots W$, and for each training sample $(x^{(n)}, y^{(n)}) : n : 1 \dots N$
  - Compute

$$z^{(n)} = \arg\max_{z} \sum_{t=1}^{T} \theta \, f\,(z_{t-1},\ z_t,\ x_t^{(n)})$$

  - if $z^{(n)} \neq y^{(n)}$

$$\theta = \theta + \sum_{t=1}^{T} f\,(y_{t-1}^{(n)},\ y_t^{(n)},\ x_t^{(n)}) - \sum_{t=1}^{T} f\,(z_{t-1}^{(n)},\ z_t^{(n)},\ x_t^{(n)})$$

- Return θ

[Collins, 2002] approximation of voted perceptron [Freund & Schapire, 1999]

- Given a training set $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{i=1}^{N}$
- Set $\theta = 0$; $\theta_a = 0$
- For each iteration $w : 1 \dots W$, and for each training sample $(x^{(n)}, y^{(n)}) : n : 1 \dots N$
  - Compute

$$z^{(n)} = \arg\max_{z} \sum_{t=1}^{T} \theta \, f\,(z_{t-1},\ z_t,\ x_t^{(n)})$$

  - if $z^{(n)} \neq y^{(n)}$

$$\theta = \theta + \sum_{t=1}^{T} f\,(y_{t-1}^{(n)},\ y_t^{(n)},\ x_t^{(n)}) - \sum_{t=1}^{T} f\,(z_{t-1}^{(n)},\ z_t^{(n)},\ x_t^{(n)})$$

  - $\theta_a = \theta_a + \theta$
- Return $\theta_a \ / \ N \cdot W$

PROPERTIES OF THE PERCEPTRON

- If the data is separable, Perceptron algorithm converges.
  Also there are partial solutions for non-separable cases.
- It is a very simple framework that can work with many structured problems and that works very well:
  - all you need is (fast) 1-best inference (Viterbi);
  - typically reaches a good solution after only a few iterations;
  - can be applied to parsing, translation, etc.

References

- J.Lafferty, A.McCallum and F.Pereira: *Conditional Random Fields: Probabilistic Models for Segmentating and Labeling Sequence Data*. ICML, pp. 282–289, 2001.
- C.Sutton and A.McCallum. *An Introduction to Conditional Random Fields*. Machine Learning, pp. 1–35, 2010.
- M. Collins: *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms*. EMNLP, pp. 1–8, 2012.
- J.Turian and I.D.Melamed: *Advances in Discriminative Parsing*. COLING/ACL, pp. 873–880, 2006.
- J.R.Finkel, A.Kleeman and C.D.Manning: *Efficient, Feature-based, Conditional Random Field Parsing*. ACL/HLT, pp. 959–967, 2008.

Problem definition [Liang 09]:

➢ Given a set of unlabeled examples $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$, maximizing the log-likelihood:

$$l(\theta) = \sum_{n=1}^{N} \log\ P_\theta(\boldsymbol{x}^{(n)})$$

Problem definition [Liang 09]:

- Given a set of unlabeled examples $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$, maximizing the log-likelihood:

$$l(\theta) = \sum_{n=1}^{N} \log\ P_\theta(\boldsymbol{x}^{(n)})$$

- Training algorithm: EM algorithm [Dempster 77, Neal 98, Cappé 09]

$$\overline{\theta_{ij}} = \frac{\sum_{n=1}^{N} \frac{1}{P_\theta(\boldsymbol{x}^{(n)})} \sum_{\boldsymbol{t} \in \theta(\boldsymbol{x}^{(n)})} \mathcal{N}(\theta_{ij}, \boldsymbol{t})\ P_\theta(\boldsymbol{x}^{(n)}, \boldsymbol{t})}{\sum_{n=1}^{N} \frac{1}{P_\theta(\boldsymbol{x}^{(n)})} \sum_{\boldsymbol{t} \in \theta(\boldsymbol{x}^{(n)})} \mathcal{N}(\theta_{i}, \boldsymbol{t})\ P_\theta(\boldsymbol{x}^{(n)}, \boldsymbol{t})} \qquad \forall \theta_{ij}$$

Problem definition [Liang 09]:

- Given a set of unlabeled examples $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$, maximizing the log-likelihood:

$$l(\theta) = \sum_{n=1}^{N} \log\ P_\theta(\boldsymbol{x}^{(n)})$$

- Training algorithm: EM algorithm [Dempster 77, Neal 98, Cappé 09]

$$\overline{\theta_{ij}} = \frac{\sum_{n=1}^{N} \frac{1}{P_\theta(\boldsymbol{x}^{(n)})} \sum_{\boldsymbol{t} \in \theta(\boldsymbol{x}^{(n)})} \mathcal{N}(\theta_{ij}, \boldsymbol{t})\ P_\theta(\boldsymbol{x}^{(n)}, \boldsymbol{t})}{\sum_{n=1}^{N} \frac{1}{P_\theta(\boldsymbol{x}^{(n)})} \sum_{\boldsymbol{t} \in \theta(\boldsymbol{x}^{(n)})} \mathcal{N}(\theta_{i}, \boldsymbol{t})\ P_\theta(\boldsymbol{x}^{(n)}, \boldsymbol{t})} \qquad \forall \theta_{ij}$$

Rewriting the numerator for all parameters:

$$\sum_{n=1}^{N} \sum_{\boldsymbol{t} \in \theta(\boldsymbol{x}^{(n)})} \boldsymbol{\phi}(\boldsymbol{x}^{(n)}, \boldsymbol{t})\ P_\theta(\boldsymbol{t}|\boldsymbol{x}^{(n)})$$

EM algorithm [Liang 09]:

Batch EM

$\theta \leftarrow$ initialization

for each iteration $w = 1, \ldots, W$:

  $\mu \leftarrow 0$

  for each example $n = 1, \ldots, N$:

    $\rho_n \leftarrow \sum_{\boldsymbol{t}} \ \phi(\boldsymbol{x}^{(n)}, \boldsymbol{t}) \ \ P_\theta(\boldsymbol{t}|\boldsymbol{x}^{(n)})$

    $\mu \leftarrow \mu + \rho_n$

  $\theta \leftarrow \boldsymbol{\theta}(\mu)$

- $\phi(\boldsymbol{x}, \boldsymbol{t})$: mapping from a labelled example $(\boldsymbol{x}, \boldsymbol{t})$ to a vector of sufficient statistics (μ)
- $\boldsymbol{\theta}(\mu)$: maximum likelihood estimate

EM algorithm [Liang 09]:

Batch EM

$\theta \leftarrow$ initialization

for each iteration $w = 1, \ldots, W$:

  $\mu \leftarrow 0$

  for each example $n = 1, \ldots, N$:

    $\rho_n \leftarrow \sum_{\boldsymbol{t}} \ \phi(\boldsymbol{x}^{(n)}, \boldsymbol{t}) \ \ P_\theta(\boldsymbol{t}|\boldsymbol{x}^{(n)})$

    $\mu \leftarrow \mu + \rho_n$

  $\theta \leftarrow \boldsymbol{\theta}(\mu)$

Stepwise EM

$\theta \leftarrow$ initialization; $\quad k = 0$

for each iteration $w = 1, \ldots, W$:

  for each example $n = 1, \ldots, N$ in random order:

    $\rho_n \leftarrow \sum_{\boldsymbol{t}} \phi(\boldsymbol{x}^{(n)}, \boldsymbol{t}) \ \ P_\theta(\boldsymbol{t}|\boldsymbol{x}^{(n)})$

    $\theta \leftarrow \boldsymbol{\theta}(\ (1 - \eta_k)\ \theta \ + \ \eta_k \ \rho_n \ )$

    $k \leftarrow k + 1$

- $\phi(\boldsymbol{x}, \boldsymbol{t})$: mapping from a labelled example $(\boldsymbol{x}, \boldsymbol{t})$ to a vector of sufficient statistics (μ)
- $\boldsymbol{\theta}(\mu)$: maximum likelihood estimate

EM algorithm [Liang 09]:

Batch EM

$\theta \leftarrow$ initialization

for each iteration $w = 1, \ldots, W$:

  $\mu \leftarrow 0$

  for each example $n = 1, \ldots, N$:

    $\rho_n \leftarrow \sum_{\boldsymbol{t}} \ \phi(\boldsymbol{x}^{(n)}, \boldsymbol{t}) \ \ P_\theta(\boldsymbol{t}|\boldsymbol{x}^{(n)})$

    $\mu \leftarrow \mu + \rho_n$

  $\theta \leftarrow \boldsymbol{\theta}(\mu)$

Stepwise EM

$\theta \leftarrow$ initialization; $\quad k = 0$

for each iteration $w = 1, \ldots, W$:

  for each example $n = 1, \ldots, N$ in random order:

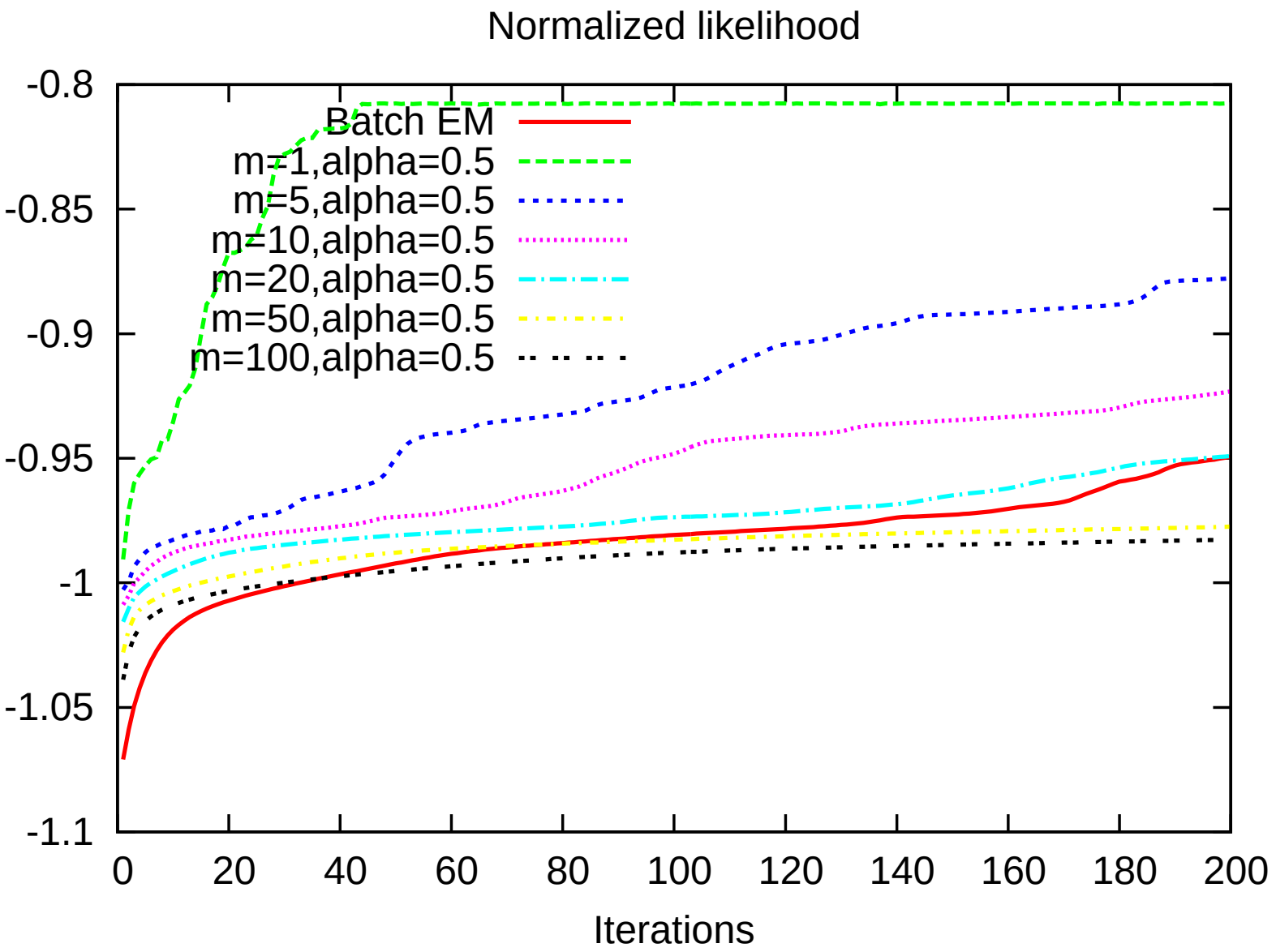
    $\rho_n \leftarrow \sum_{\boldsymbol{t}} \phi(\boldsymbol{x}^{(n)}, \boldsymbol{t}) \ \ P_\theta(\boldsymbol{t}|\boldsymbol{x}^{(n)})$

    $\theta \leftarrow \boldsymbol{\theta}(\ (1 - \eta_k)\ \theta \ + \ \eta_k \ \rho_n \ )$

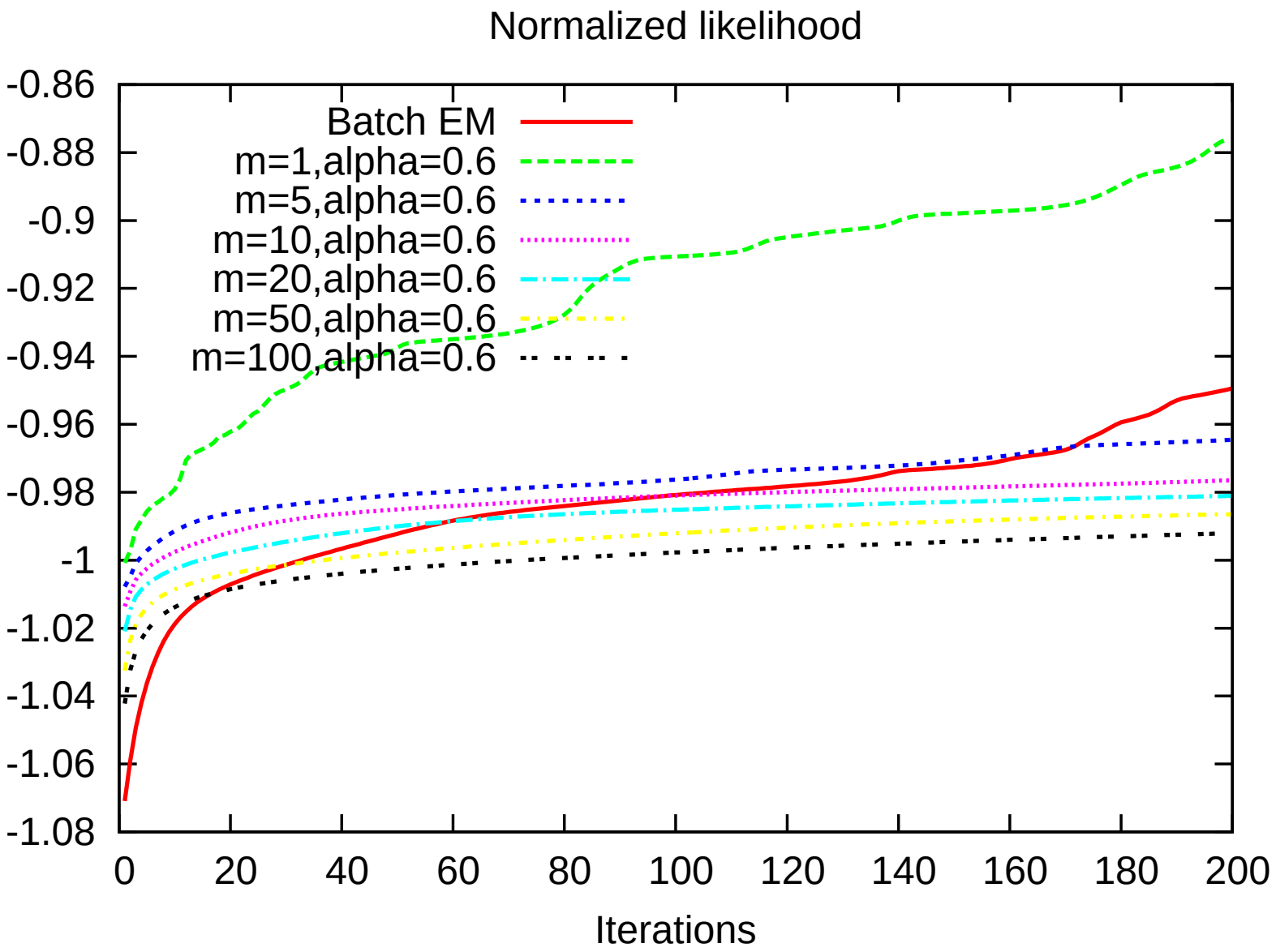
    $k \leftarrow k + 1$

- $\phi(\boldsymbol{x}, \boldsymbol{t})$: mapping from a labelled example $(\boldsymbol{x}, \boldsymbol{t})$ to a vector of sufficient statistics (μ)
- $\boldsymbol{\theta}(\mu)$: maximum likelihood estimate
- Stepwise EM: convergence is guaranteed if $\sum_{k=0}^{\infty} \eta_k = \infty$ and $\sum_{k=0}^{\infty} \eta_k^2 < \infty$
  - $\eta_k = (k + 2)^{-\alpha}$ with $0.5 \leq \alpha \leq 1$
  - Approach: take m examples at once

Palindrome language (15 random initializations, $\alpha = 0.5$)



Palindrome language (15 random initializations, $\alpha = 0.6$)



Palindrome language

[% of palindromes generated in average by the estimated grammar (50K sentences)]

- Batch EM $21.9\% \pm 10.1$
- Online EM: from random strings

$\alpha \backslash m$	1	5	10	20	50
0.5	46.7 ± 20.8	35.1 ± 21.0	29.5 ± 18.3	21.8 ± 10.2	16.6 ± 2.3
0.6	34.1 ± 20.2	20.3 ± 11.3	16.8 ± 2.4	15.8 ± 2.4	14.9 ± 2.8
0.7	17.2 ± 3.1	15.1 ± 2.7	14.9 ± 2.9	14.6 ± 3.3	14.9 ± 3.4

Palindrome language

[% of palindromes generated in average by the estimated grammar (50K sentences)]

- Batch EM $21.9\% \pm 10.1$
- Online EM: from random strings

$\alpha \backslash m$	1	5	10	20	50
0.5	46.7 ± 20.8	35.1 ± 21.0	29.5 ± 18.3	21.8 ± 10.2	16.6 ± 2.3
0.6	34.1 ± 20.2	20.3 ± 11.3	16.8 ± 2.4	15.8 ± 2.4	14.9 ± 2.8
0.7	17.2 ± 3.1	15.1 ± 2.7	14.9 ± 2.9	14.6 ± 3.3	14.9 ± 3.4

- Online EM: from short to long strings

$\alpha \backslash m$	1	5	10	20	50
0.5	82.4 ± 21.6	94.2 ± 11.6	89.6 ± 19.6	92.5 ± 12.4	65.1 ± 34.9
0.6	86.8 ± 14.2	90.8 ± 13.1	92.7 ± 12.2	84.7 ± 21.7	50.7 ± 29.3
0.7	88.8 ± 14.1	81.9 ± 19.6	66.6 ± 27.1	54.5 ± 27.0	33.6 ± 16.7

Palindrome language

[% of palindromes generated in average by the estimated grammar (50K sentences)]

- Batch EM $21.9\% \pm 10.1$
- Online EM: from random strings

$\alpha \backslash m$	1	5	10	20	50
0.5	46.7 ± 20.8	35.1 ± 21.0	29.5 ± 18.3	21.8 ± 10.2	16.6 ± 2.3
0.6	34.1 ± 20.2	20.3 ± 11.3	16.8 ± 2.4	15.8 ± 2.4	14.9 ± 2.8
0.7	17.2 ± 3.1	15.1 ± 2.7	14.9 ± 2.9	14.6 ± 3.3	14.9 ± 3.4

- Online EM: from short to long strings

$\alpha \backslash m$	1	5	10	20	50
0.5	82.4 ± 21.6	94.2 ± 11.6	89.6 ± 19.6	92.5 ± 12.4	65.1 ± 34.9
0.6	86.8 ± 14.2	90.8 ± 13.1	92.7 ± 12.2	84.7 ± 21.7	50.7 ± 29.3
0.7	88.8 ± 14.1	81.9 ± 19.6	66.6 ± 27.1	54.5 ± 27.0	33.6 ± 16.7

- Online EM: from long to short strings

$\alpha \backslash m$	1	5	10	20	50
0.5	30.6 ± 2.0	83.4 ± 11.9	84.6 ± 19.4	71.0 ± 27.7	33.2 ± 26.9
0.6	79.6 ± 13.1	70.2 ± 28.6	42.4 ± 30.6	26.0 ± 18.6	16.8 ± 3.9
0.7	52.1 ± 28.3	19.0 ± 6.5	17.0 ± 4.3	16.1 ± 4.0	15.9 ± 3.7

Motivation: Supervised learning: (x, y) x: input data (sentence) y: label

- Problem: to annotate data is slow and expensive
- **Active learning: to annotate just the necessary data**

Pool-based active learning algorithm [Settles 2008, Settles 2010]:

Given: Labeled set $\mathcal{L}$, unlabeled pool $\mathcal{U}$,
query strategy $\phi(\dot{})$, query batch size B

repeat

// learn a model using the current $\mathcal{L}$

$\theta = \text{train}(\mathcal{L})$

for $b = 1$ **to** B **do**

// query the most informative instance

$\mathbf{x}_b^* = \arg\max_{\mathbf{x} \in \mathcal{U}} \ \phi(\mathbf{x})$

// move the labeled query from $\mathcal{U}$ to $\mathcal{L}$

$\mathcal{L} = \mathcal{L} \cup \langle \mathbf{x}_b^*, \text{label}(\mathbf{x}_b^*) \rangle$

$\mathcal{U} = \mathcal{U} - \mathbf{x}_b^*$

end

until *some stopping criterion*

- **Uncertainty sampling** We select the sample that it has most uncertainty about their label
  - **Least confidence (LC)**:

$$\phi^{LC}(\boldsymbol{x}) \;=\; 1 \;-\; P_\theta(\widehat{\boldsymbol{t}}\,|\,\boldsymbol{x})$$

  - **Sequence entropy (SE)**

$$\phi^{SE}(\boldsymbol{x}) \;=\; -\sum_{\widehat{\boldsymbol{t}}} P_\theta(\widehat{\boldsymbol{t}}\,|\,\boldsymbol{x}) \;\log\; P_\theta(\widehat{\boldsymbol{t}}\,|\,\boldsymbol{x})$$

  - **N-best sequence entropy (NSE)**

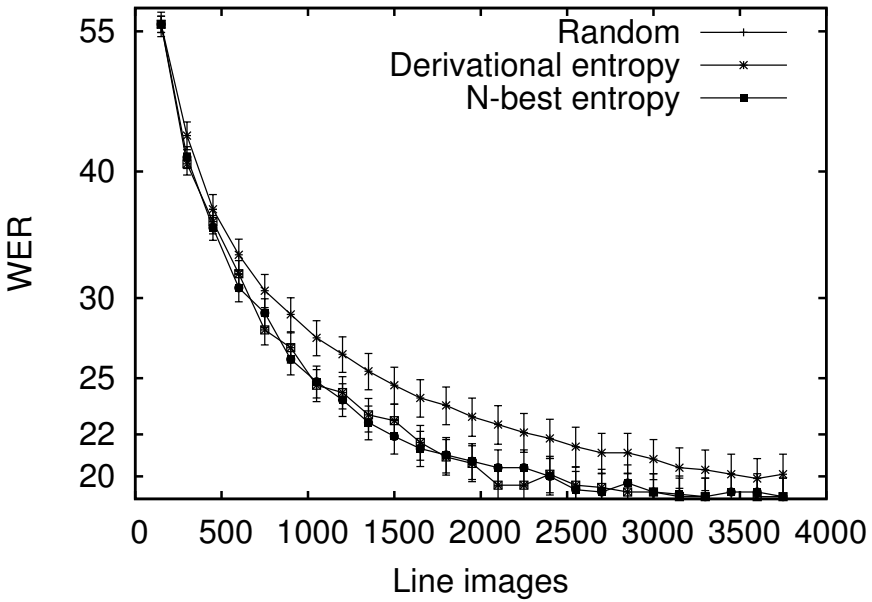
$$\phi^{NSE}(\boldsymbol{x}) \;=\; -\sum_{\widehat{\boldsymbol{t}}\in\mathcal{N}} P_\theta(\widehat{\boldsymbol{t}}\,|\,\boldsymbol{x}) \;\log\; P_\theta(\widehat{\boldsymbol{t}}\,|\,\boldsymbol{x})$$

Handwritten Text Recogntion experiment [Romero 18]:

- ESPOSALLES dataset: single writer in old Catalan in the 17th century, composed by 173 pages that contain $5,437$ lines.

	$\mathcal{L}$	$\mathcal{U}$	Test	Total
Lines	150	4,470	827	5,447
Running words	1,650	50,234	8,893	60,777
Vocabulary	397	2,914	1,119	3,500

- Initial model trained on 150 sentences randomly selected
- Batch size: 150 lines for each selection iteration (about 5 pages)



HMMs: Practice

- J.K. Baker. *The dragon system - an overview*. IEEE Trans. on Acoustics, Speech and Signal Processing, 23(1), 31-35, 1979.
- K. Lari, S.J. Young. *The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm*. Computer Speech and Language, 4, 35-56, 1990.
- H. Ney. *Dynamic Programing Parsing for Context-Free Grammars in Continous Speech Recognition*. IEEE Trans. Signal Processing, 39(2), 336-340, 1991.
- F. Pereira, Y. Schabes. *Inside-outside reestimation from partially bracketed corpora*. Proc. ACL, 128-135, 1992.
- J.A. Sánchez, J.M. Benedí. *Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(9), 1052-1055, 1997.
- V.M. Jiménez, A. Marzal. *Computation of the N Best Parse Trees for Weighted and Stochastic Context-Free Grammars*. LNCS 1876, 183-192, 2000.
- J.M. Benedí, J.A. Sánchez. *Estimation of stochastic context-free grammars and their use as language models*. Computer Speech & Language, 19(3), 249-274, 2005.
- J.A. Sánchez, V. Romero. *Computation of moments for probabilistic finite-state automata*. Information Sciences, 515, 388-400, 2020.