Topics in Natural Language Processing

Shay Cohen

Institute for Language, Cognition and Computation

University of Edinburgh

Lecture 8

Estimation

$$f(w|x_1, y_1, \ldots, x_n, y_n) = \prod_{i=1}^{n} \frac{\exp\left(w^\top g(x, y)\right)}{Z(w)}$$

average

What is the log-likelihood?

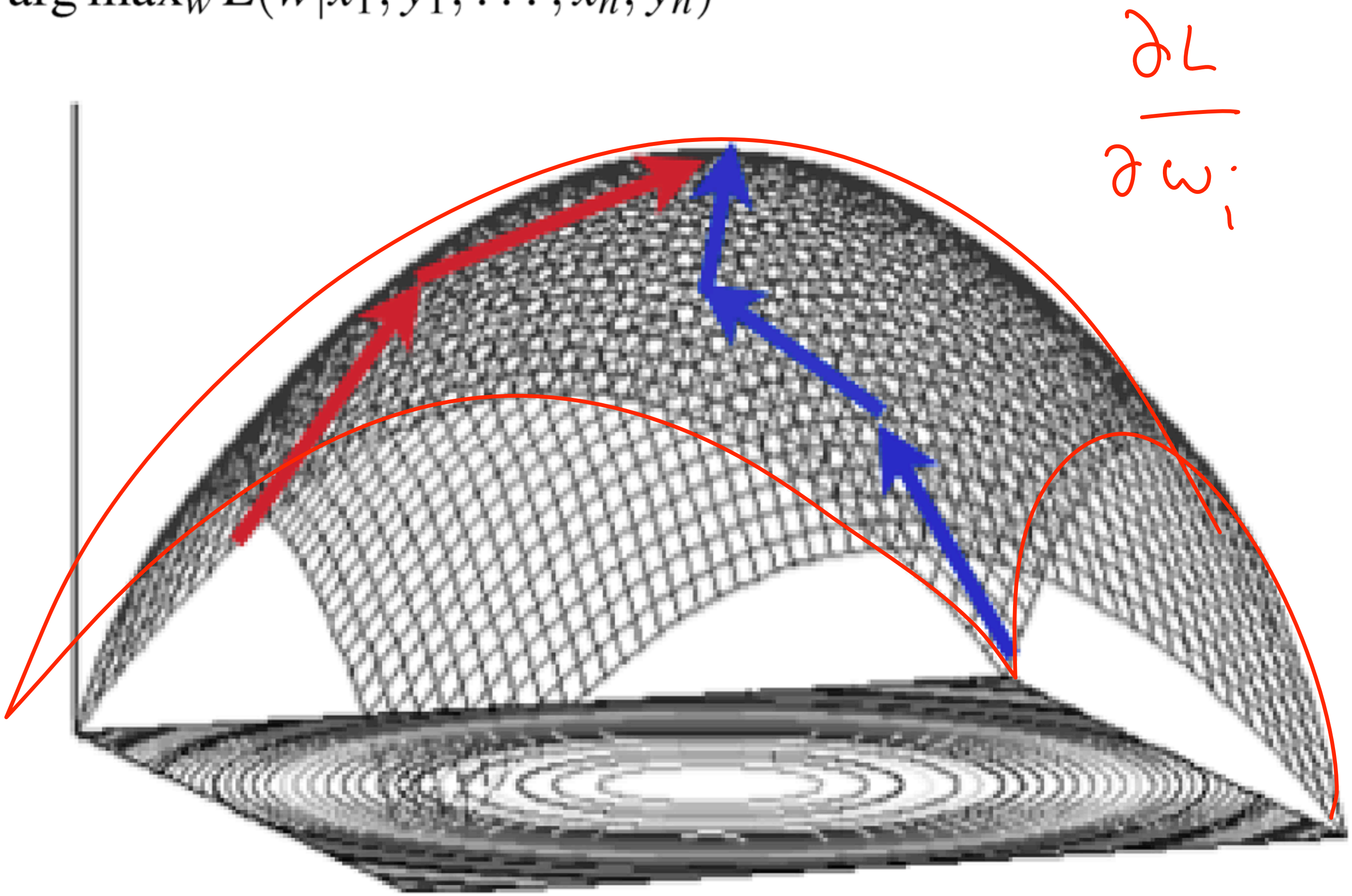
$$L(w|x_1, y_1, \ldots, x_n, y_n) = \left[\frac{1}{n} \sum_{i=1}^{n} w^T g(x_i, y_i)\right] - \log Z(w)$$

$$Z(w) = \sum_{x,y} \exp\left(w^T g(x, y)\right)$$

$$p(x, y)$$

Maximising the log-likelihood

$w^* = \arg\max_w L(w|x_1, y_1, \ldots, x_n, y_n)$



Maximising the log-likelihood

Many of the maximisation algorithms are a variant of the update:

$$w^{(t+1)} \leftarrow w^{(t)} + \mu v$$

where $v \in \mathbb{R}^d$ and $v_i = \frac{\partial L}{\partial w_i}\left(w^{(t)}\right)$.

Estimation

What is the average log-likelihood?

$$L(w|x_1, y_1, \ldots, x_n, y_n) = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{d} w_j g_j(x_i, y_i) - \log Z(w)\right)$$

What is the derivative?

$$\frac{\partial L}{\partial w_j} = \left[\frac{1}{n}\sum_{i=1}^{n} g_j(x_i, y_i)\right] - \frac{\partial \log Z(w)}{\partial w_j}$$

Derivative of $Z(w)$

$(e^{y})' = e^{y} y'$

$$Z(w) = \sum_{x,y} \exp\left(\sum_{j=1}^{d} w_j g_j(x,y)\right)$$

$$\frac{\partial Z}{\partial w_j}(w) = \sum_{x,y} \exp\left(\sum_{j=1}^{d} w_j g_j(x,y)\right) \cdot g_j(x,y)$$

$$\frac{\partial \log Z(w)}{\partial w_j} = \frac{1}{Z(w)} \cdot \frac{\partial Z(w)}{\partial w_j}$$

$$\frac{1}{Z(w)} \cdot \sum_{x,y} \exp\left(\sum_{j=1}^{d} w_j g_j(x,y)\right) \cdot g_j(x,y)$$

Gradient of average log-likelihood

$\sum_z p(z) f(z)$

$$\frac{\partial L}{\partial w_j} = \left(\frac{1}{n} \sum_{i=1}^{n} g_j(x_i, y_i) \right) - \sum_{x,y} \frac{\exp(\sum_{k=1}^{d} w_k g_k(x,y))}{Z(w)} g_j(x,y)$$

$$\frac{1}{n} \sum_{i=1}^{n} g_j(x_i, y_i) =$$

averge of $g_j(x,y)$ on the observed data

$p(x,y|w)$

$$\sum_{x,y} \frac{\exp(\sum_{k=1}^{d} w_k g_k(x,y))}{Z(w)} g_j(x,y) =$$

$E_w[g_j(X,Y)]$

Gradient of average log-likelihood

$$\frac{\partial L}{\partial w_j} = \left(\frac{1}{n} \sum_{i=1}^{n} g_j(x_i, y_i) \right) - \sum_{x,y} \frac{\exp(\sum_{k=1}^{d} w_k g_k(x,y))}{Z(w)} g_j(x,y)$$

$$\frac{1}{n} \sum_{i=1}^{n} g_j(x_i, y_i) =$$

$$\sum_{x,y} \frac{\exp(\sum_{k=1}^{d} w_k g_k(x,y))}{Z(w)} g_j(x,y) =$$

max $H(p)$

$E[g_j] = \text{avg}(g_j)$

according to data

Therefore, the gradient is the difference between empirical expectations and expectations under the model

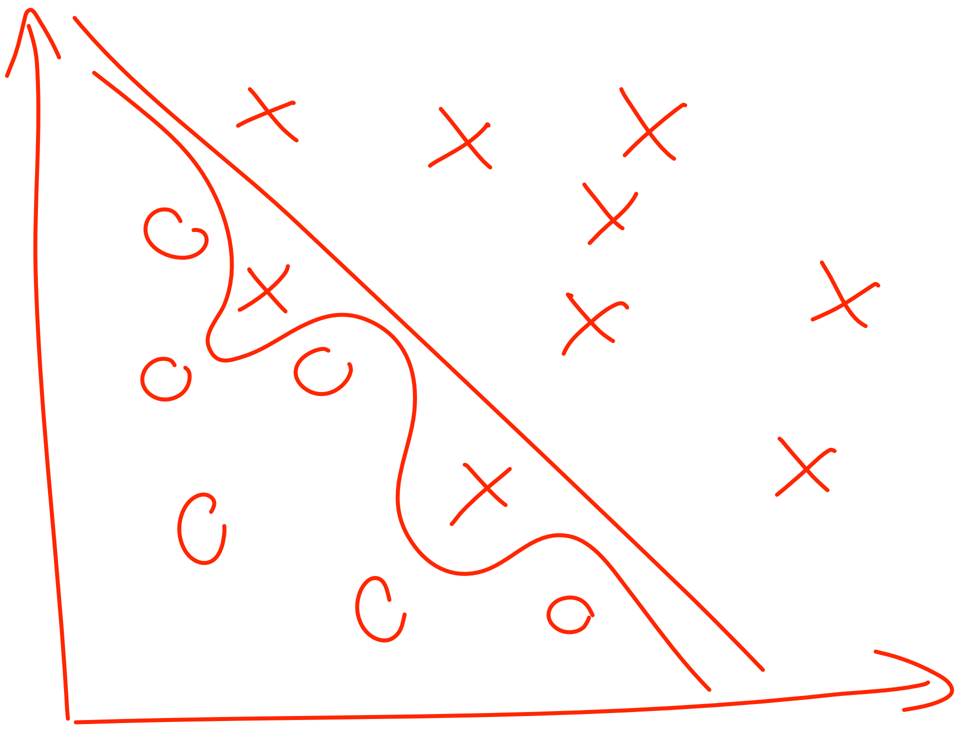
Overfitting

The advantage of log-linear models: can have arbitrary features

The problem: too many features lead to overfitting

Regularisation

What is overfitting?



L_2 Regularisation

New objective:

$$G(w|x_1, y_1, \ldots, x_n, y_n) = L(w|x_1, y_1, \ldots, x_n, y_n) - \lambda ||w||_2^2$$

regularizer

where $||w||_2^2 = \sum_{i=1}^{d} w_i^2$

Partial derivatives:

$$\frac{\partial G}{\partial w_j} = \frac{\partial L}{\partial w_j} - 2\lambda \cdot w_j$$

L_1 Regularisation

New objective:

$$G(w|x_1, y_1, \ldots, x_n, y_n) = L(w|x_1, y_1, \ldots, x_n, y_n) - \lambda||w||_1^2$$

where $||w||_1^2 = \sum_{i=1}^{d} |w_i|$

Encourages sparse solutions, such that most of w_i are exactly 0

Bayesian interpretation to regularlisation

$$G(w|x_1, y_1, \ldots, x_n, y_n) = L(w|x_1, y_1, \ldots, x_n, y_n) - \frac{\lambda}{2}||w||_2^2$$

Could the answer be a MAP estimate for some prior?

$$G(w|x_1, y_1, \ldots, x_n, y_n) \propto \log p(x_1, y_1, \ldots, x_n, y_n|w) + \log p(w)$$

$$p(w) \propto \exp\left(-\frac{\lambda}{2}\sum_i w_i^2\right)$$

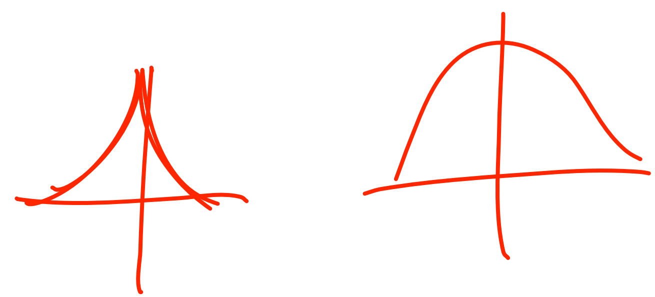
Bayesian interpretation to regularlisation

$$G(w|x_1, y_1, \ldots, x_n, y_n) = L(w|x_1, y_1, \ldots, x_n, y_n) - \frac{\lambda}{2}||w||_2^2$$

Could the answer be a MAP estimate for some prior?

$$G(w|x_1, y_1, \ldots, x_n, y_n) \propto \log p(x_1, y_1, \ldots, x_n, y_n|w) + \log p(w)$$

$p(w) \propto$



This means that $p(w)$ is a Gaussian distribution with mean 0 and variance $1/\lambda$

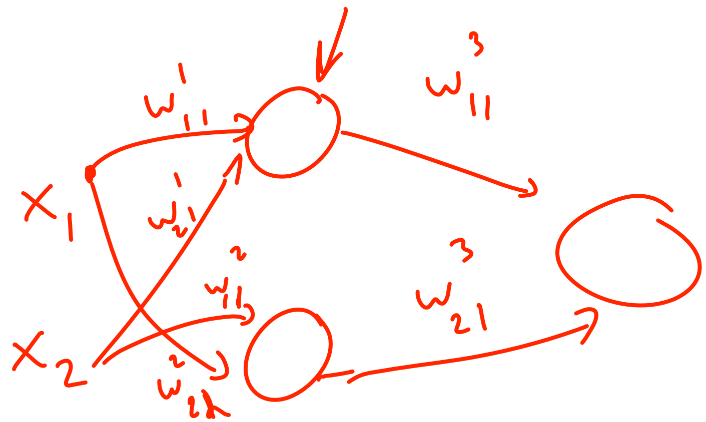
MLE with L_2-regularisation is MAP estimate with Gaussian prior

Dimensionality Reduction

- Data can be more efficiently processed
- Easier to visualize data
- Gives a low-dimensional representations for the data that can be used in other NLP problems.
- Recent example for representation learning: neural networks

Neural Networks

Example of a neural network:



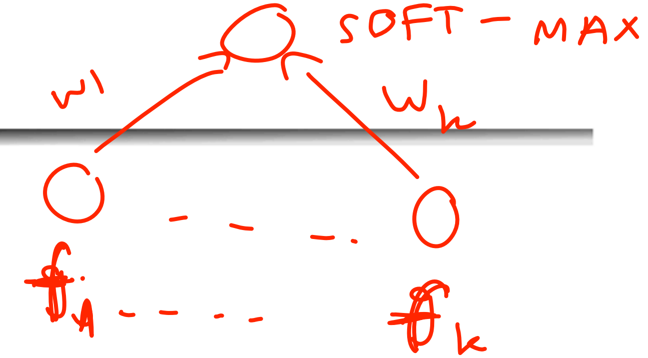
The general case:

$$g(w^1_{11} x_1 + w^1_{21} x_2)$$

↑ activation function

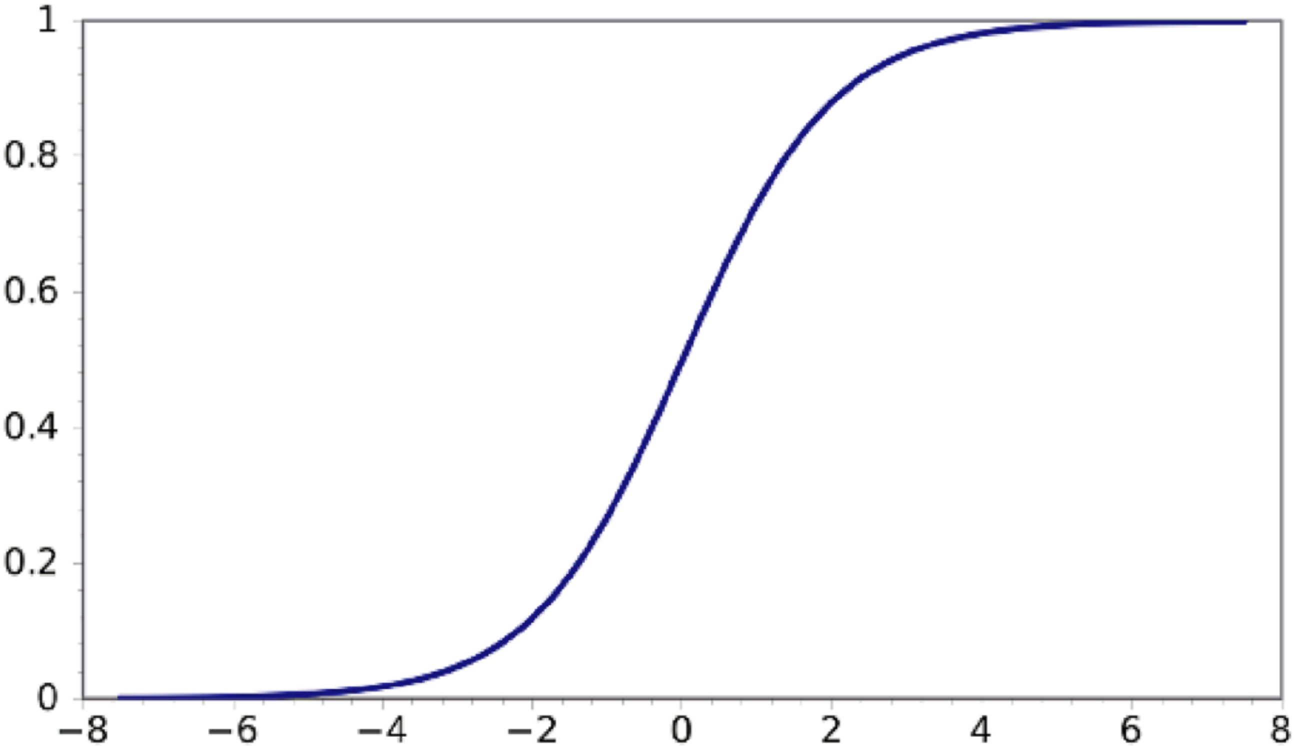
w^k_{ij} – that's the weight connecting neuron i to neuron j in layer k

Activation Functions



The sigmoid function $g(x) = \dfrac{1}{1+e^{-x}}$

$$g = \frac{\exp\left(\sum w_i f_i(x)\right)}{\sum_{x'} \exp\left(\sum w_i f_i(x')\right)}$$

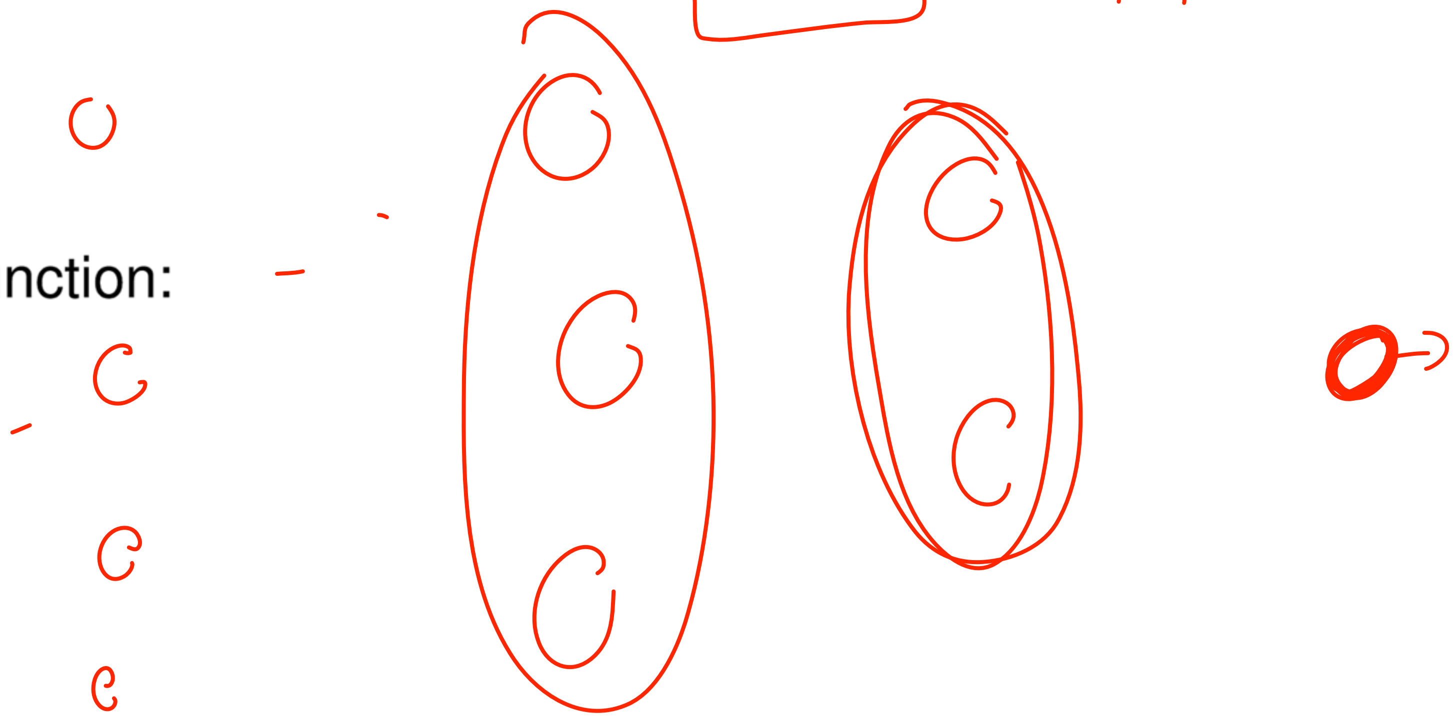


Two important points: (1) $g(x)$ is differentiable
(2) $g'(x) = g(x)(1 - g(x))$

Learning Problem

$$\frac{\partial f}{\partial z} = \sum_{i=1}^{n} \frac{\partial f}{\partial v_i} \cdot \frac{\partial v_i}{\partial z}$$

Training data:



Objective function:

The Backpropagation Algorithm

Learning from Incomplete Data

- Semi-supervised learning

  Small amounts of labelled data and large amounts of "unlabeled" data

- Latent variable learning

  Some information about the structure is missing

- Unsupervised learning

  Hard: given only inputs, learn a decoder

How to estimate a PCFG?

We learned how to estimate a PCFG from treebank

Reminder:

Unsupervised learning: PCFGs

How to estimate a PCFG from strings?

General case: Viterbi (or “hard”) EM

Model:

Observed Data:

Step 0:

Step 1:

Step 2:

Repeat step 1

Maximum likelihood estimation

General principle: write down the likelihood of **whatever** you observe, and then maximise with respect to parameters

Model: $p(x, y \mid \theta)$

Observed: $x_1, \ldots, x_n$

Likelihood:

$$L(x_1, \ldots, x_n \mid \theta) =$$

The EM Algorithm

- A softer version of hard EM
- Instead of identifying a single tree per sentence, identify a distribution over trees (E-step)
- Then re-estimate the parameters, with each tree for each sentence “voting” according to its probability (M-step)
- Semiring parsing: instead of CKY use the inside algorithm

EM: Main Disadvantage

Sensitivity to initialisation (finds local maximum)

Global log-likelihood optimisation in general is “hard”

Latent-variable learning

"Structure" is present

Some information is missing from model

Model: $p(x, y, h \mid \theta)$

Observed: $(x_1, y_1), \ldots, (x_n, y_n)$

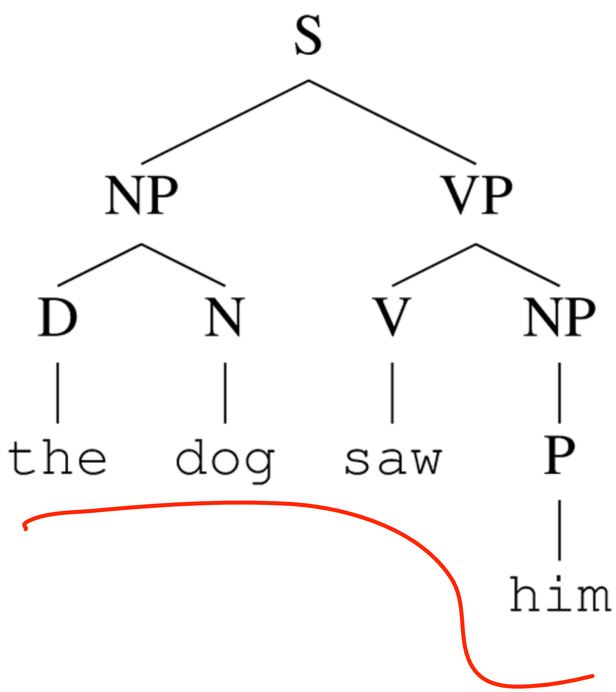
Log-likelihood:

$$L(x_1, \ldots, x_n, y_1, \ldots, y_n | \theta) =$$

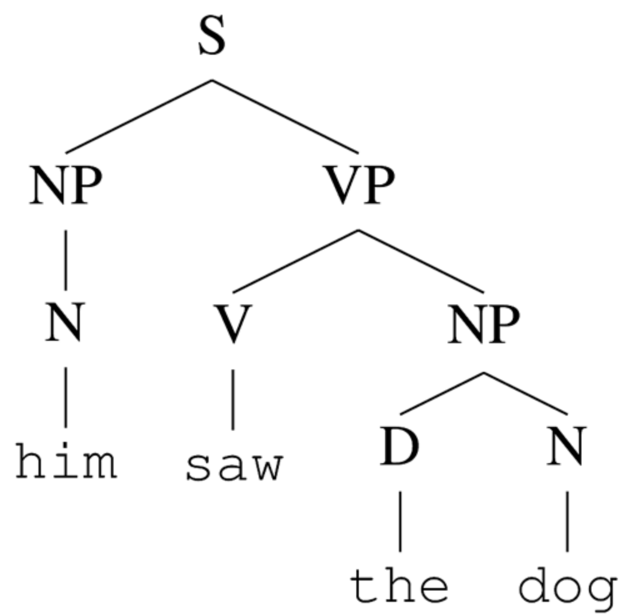
Example of Latent-Variable Use in PCFGs

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

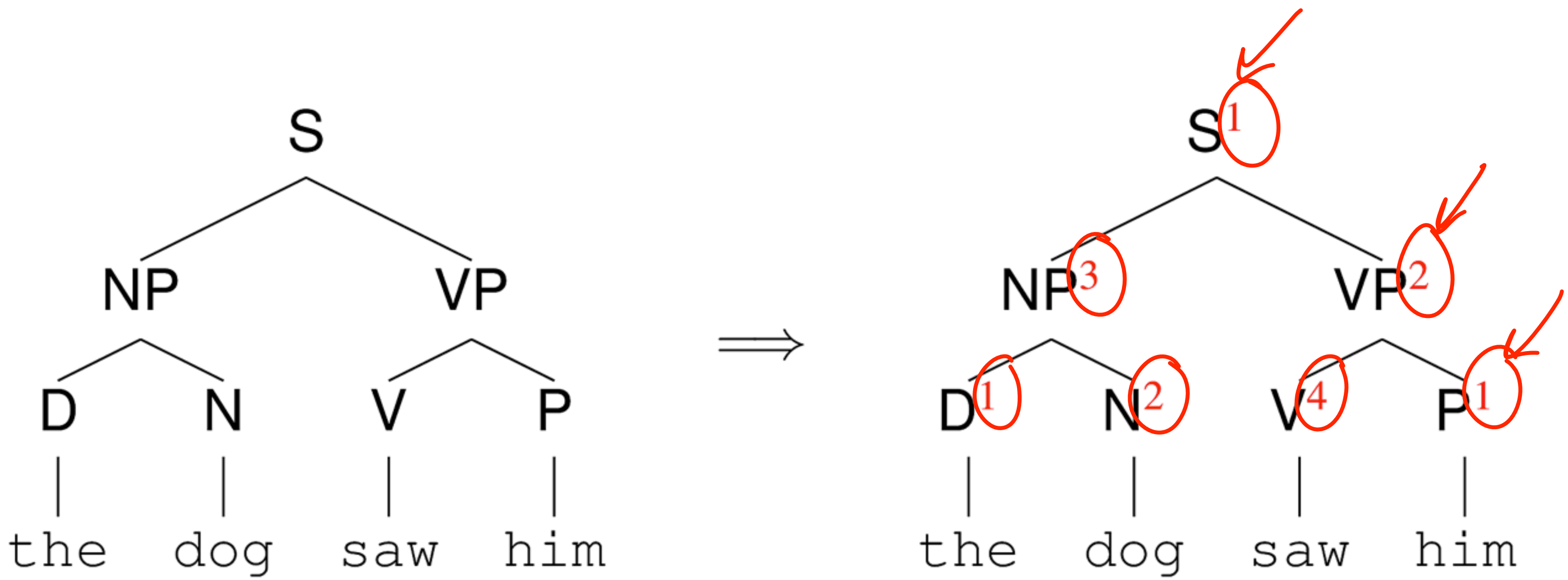
Seen in data:



Unseen in data (ungrammatical):



Latent-Variable PCFGs



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

Semi-supervised Learning

Main idea: use a relatively small amount of annotated data, and exploit also large amounts of unannotated data

The term itself is used in various ways with various methodologies

Example: Word Clusters and Embeddings

- Learn clusters of words or embed them in Euclidean space using large amounts of text
- Use these clusters/embeddings as features in a discriminative model

Semi-supervised Learning: Example 2

Combine the log-likelihood for labelled data with the log-likelihood for unlabelled data

$$L(x_1, y_1, \ldots, x_n, y_n, x'_1, \ldots, x'_m | \theta) =$$

Semi-supervised Learning: Example 3

Self-training

Semi-supervised Learning: Example 3

Self-training

Step 1:

Step 2:

Step 3:

Potentially, repeat step 2