Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Basic Models

Tobias Scheffer

Overview

- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.

Graphical Models

- Model of the joint distribution $p(X_1, \ldots, X_N)$ of a set of random variables.
- Direct dependence and independence of variables is represented in the graphical structure.
- Inference: Model allows to infer
  - All marginal probabilities $p(X_{i_1}, \ldots, X_{i_m})$,
  - All conditional probabilities $p(X_{i_1}, \ldots, X_{i_k} | X_{i_{k+1}}, \ldots, X_{i_m})$.

- Random variables can represent acoustic signals, letters, words, parts of speech, semantic labels, …

Directed Graphical Models

- Directed graphical model is a graph over
  - Nodes: set of random variables $\{X_1, \dots, X_N\}$.
  - With edges $\subset \{X_1, \dots, X_N\}^2$ that contain no cycles.
  - Edges are written $X_j \rightarrow X_i$.

- Parents $pa(X_i) = \{X_j | X_j \rightarrow X_i\}$ are the nodes which X_i is directly dependent on.

- The directed model represents a joint distribution
  - $P(X_1, \dots, X_N) = \prod_{i=1}^{N} p(X_i | pa(X_i))$.

Directed Graphical Models

- Why does the graph have to be acyclic?
- It the graph is acyclic then there is an ordering $i_1, \ldots, i_N$ of the nodes such that
  - For all i_j and for all $X_{i_k} \in pa\left(X_{i_j}\right): k < j$.
  - That is, the parents come before their children in the odering.
- For such an ordering, we can factorize
  - $P(X_1, \ldots, X_N) = \prod_{i=1}^{N} p(X_i | pa(X_i))$.

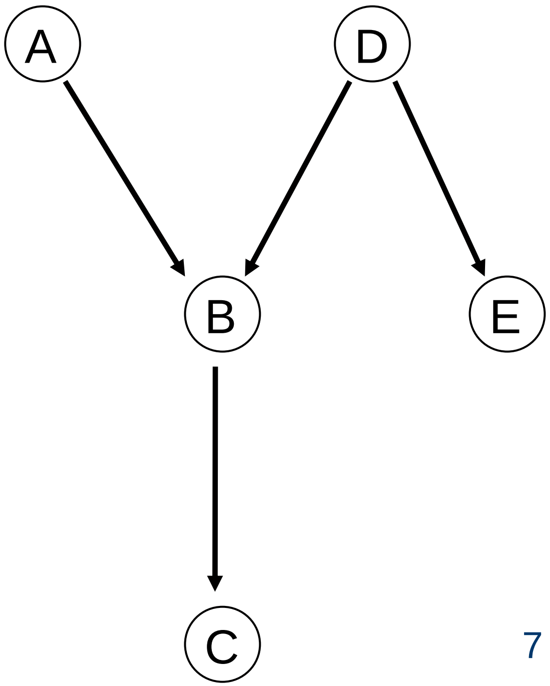
Before X_i in the ordering

Directed Graphical Models: Independence

- The graph structure of a graphical model implies (conditional) independencies between random variables.
  - $\neg(X_i \rightarrow X_j)$ implies $p(X_j \mid pa(X_j), X_i) = p(X_j \mid pa(X_j))$.
- Independences make inference easier!
  - Depending on the structure of the model, polynomial instead of exponential complexity.
- Dependences and independences represent domain knowledge and modeling assumptions.

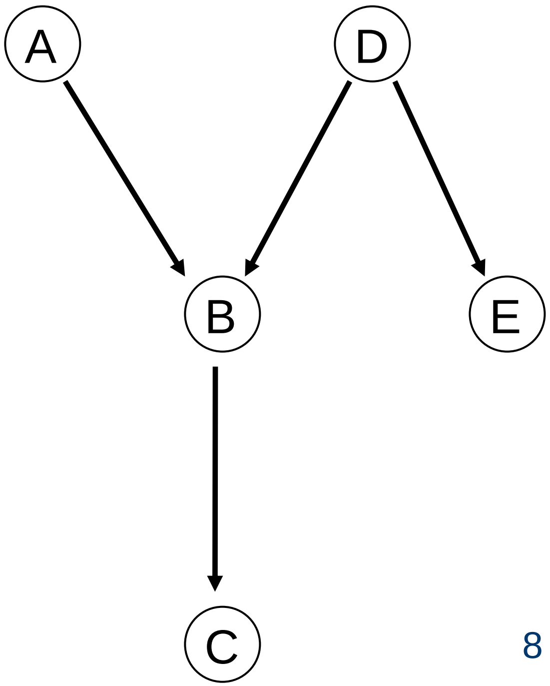
Directed Graphical Models: Example

- Ordering:



Directed Graphical Models: Example

- Ordering: A, D, B, E, C.
- $P(A, B, C, D, E) = P(A)P(D)P(B|A, D)P(E|D)P(C|B)$.

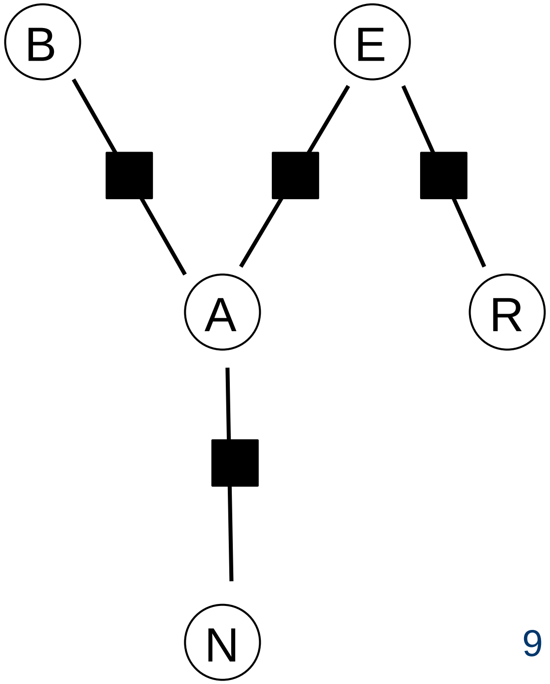


Undirected Graphical Models

- Undirected graphical model over $X_1, \ldots, X_N$:

$$P(X_1, \ldots, X_N) = \frac{1}{Z} \prod_{i=1}^{k} \Psi_i$$

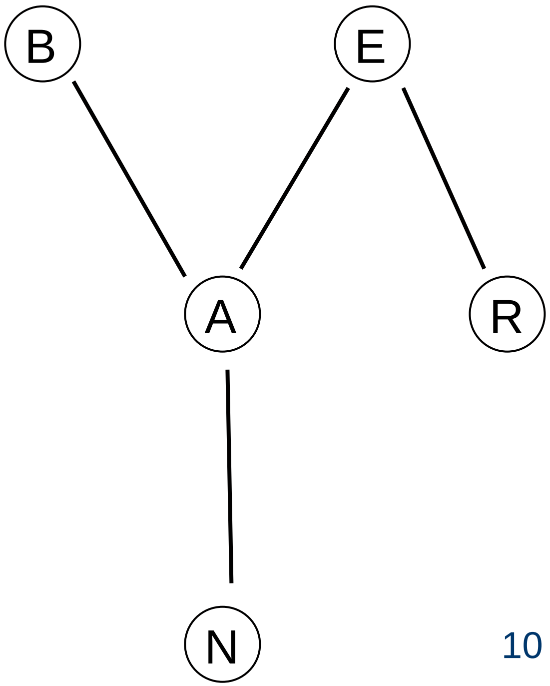
- Represented by a factor graph
- $\frac{1}{Z} P(A,B) P(A,E) P(E,R) P(A,N)$



- Solid nodes are factors.
- Each factor is joint distribution of its connected nodes.

Undirected Graphical Models

- Undirected graphical model over $X_1, \dots, X_N$:
$$P(X_1, \dots, X_N) = \frac{1}{Z} \prod_{i=1}^{k} \Psi_i$$
- Factor graph represents factorization
- Markov field: connect nodes that occur in a joint factor.
- Markov field reflects conditional independence.



Undirected Graphical Models

- Undirected graphical model over $X_1, \ldots, X_N$:

$$P(X_1, \ldots, X_N) = \frac{1}{Z} \prod_{j=1}^{k} \Psi_j$$

- With $\Psi_j = e^{\psi_j}$:

$$P(X_1, \ldots, X_N) = \frac{1}{Z} \exp\left\{\sum_{j=1}^{k} \psi_j\right\}$$

Inference in Graphical Models

- Problem setting for inference.
- Given observations for some variables $X_{i_1}, \dots, X_{i_m}$,
- Infer distribution of query variable X_q:
  - $p\left(X_q \middle| x_{i_1}, \dots, x_{i_m}\right)$.
- Cannot immediately calculate value because the values of the unobserved parents are unknown.
  - Sum over all values (summation rule).
- Notation: $\{X_1, \dots, X_N\} = \{ \underbrace{X_q}_{\text{query}}, \underbrace{X_{i_1}, \dots, X_{i_m}}_{\text{observed}}, \underbrace{X_{j_1}, \dots, X_{j_k}}_{\text{unobserved}} \}$

Graphical Models: Inference

- Notation: $\{X_1, \ldots, X_N\} = \{ \underbrace{X_q}_{\text{query}}, \underbrace{X_{i_1}, \ldots, X_{i_m}}_{\text{observed}}, \underbrace{X_{j_1}, \ldots, X_{j_k}}_{\text{unobserved}} \}$
- Inference:

$$p(X_q | x_{i_1}, \ldots, x_{i_m}) = \frac{p(X_q, x_{i_1}, \ldots, x_{i_m})}{p(x_{i_1}, \ldots, x_{i_m})}$$

$$= \frac{1}{p(x_{i_1}, \ldots, x_{i_m})} \sum_{x_{j_1}} \ldots \sum_{x_{j_k}} p(X_q, x, \ldots, x_{i_m}, x_{j_1}, \ldots, x_{j_k})$$

Number of summands is exponential
in the number of unobserved variables.

Graphical Models: Inference

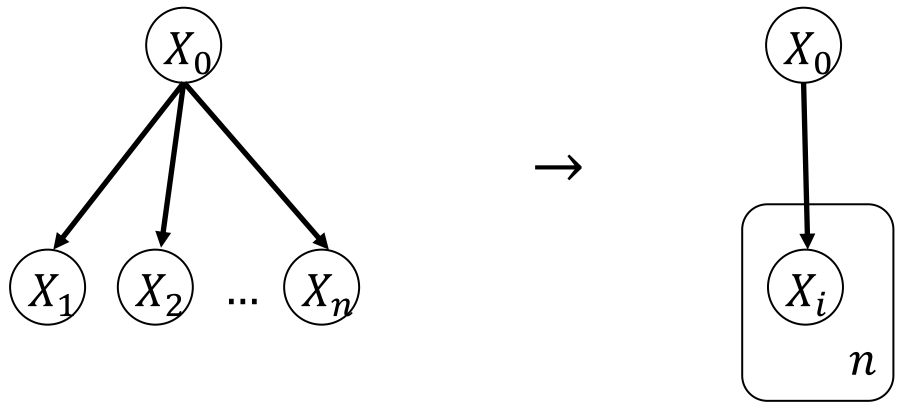
- Inference:
  $p\left(X_q \middle| x_{i_1}, \ldots, x_{i_m}\right)$
  $= \frac{1}{p(x_{i_1}, \ldots, x_{i_m})} \sum_{x_{j_1}} \ldots \sum_{x_{j_k}} p(X_q, x, \ldots, x_{i_m}, x_{j_1}, \ldots, x_{j_k})$
- Exact inference:
  - Message passing algorithm.
  - For general graph structures intractable.
  - If the model has sequential structure: quadratic (Viterbi-/forward-backward algorithm).
  - If the model has tree structure: cubic.

Graphical Models: Inference

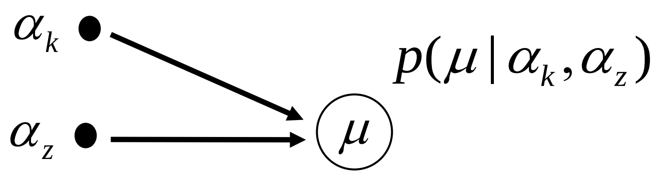
- Inference:
  $p\left(X_q \middle| x_{i_1}, \dots, x_{i_m}\right)$
  $= \frac{1}{p(x_{i_1}, \dots, x_{i_m})} \sum_{x_{j_1}} \dots \sum_{x_{j_k}} p(X_q, x, \dots, x_{i_m}, x_{j_1}, \dots, x_{j_k})$
- Approximate inference for general graph structures
  - Loopy belief propagation,
  - Variational methods.

Plate Notation

- Shorthand notation for “loops”:



- Notation for parameters of distributions:



Overview

- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.

The n-Gram Model

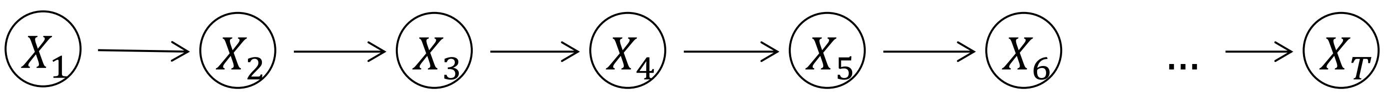
- Basic tool for language modeling.
- Unigram model ($n = 1$)
- Based on the Markov assumption of order 0:
  - $p(X_t | X_{t-1}, \ldots, X_1) = p(X_t)$

- $p(X_1, \ldots, X_T) = p(X_1)p(X_2)p(X_3)p(X_4)p(X_5)p(X_6) \ldots p(X_T)$

$$= \prod_{i=1}^{T} p(X_i)$$

The n-Gram Model

- Basic tool for language modeling.
- Bigram model ($n = 2$)
- Based on the Markov assumption of order 1:
  - $p(X_t | X_{t-1}, \ldots, X_1) = p(X_t | X_{t-1})$

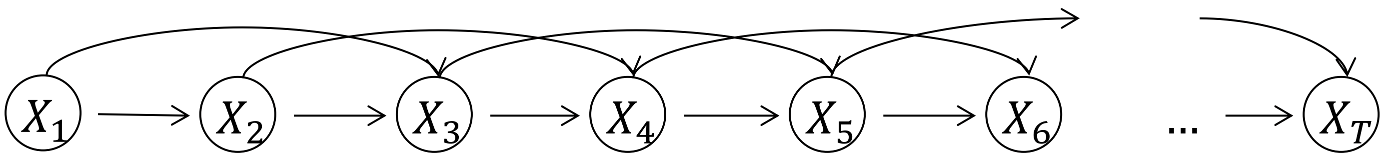


- $p(X_1, \ldots, X_T) = p(X_1)p(X_2|X_1)p(X_3|X_2) \ldots p(X_T|X_{T-1})$

$$= P(X_1) \prod_{t=2}^{T} p(X_t | X_{t-1})$$

The n-Gram Model

- Basic tool for language modeling.
- Trigram model ($n = 3$)
- Based on the Markov assumption of order 3:
  - $p(X_t | X_{t-1}, \dots, X_1) = p(X_t | X_{t-1}, X_{t-2})$



- $p(X_1, \dots, X_T)$
  $= p(X_1) p(X_2 | X_1) p(X_3 | X_2, X_1) \dots p(X_T | X_{T-1}, X_{T-2})$
  $= P(X_1) p(X_2 | X_1) \prod_{t=3}^{T} p(X_t | X_{t-1}, X_{t-2})$

The n-Gram Model

- Basic tool for language modeling.
- Based on the Markov assumption of order $n-1$:
  - $p(X_t|X_{t-1},\ldots,X_1) = p(X_t|X_{t-1},\ldots,X_{t-n+1})$
- n-gram model:

$$
\begin{aligned}
& p(X_1,\ldots,X_T) \\
& = p(X_1)\ldots p(X_T|X_{T-1},\ldots,X_{T-n+1}) \\
& = P(X_1)\ldots p(X_{n-1}|X_{n-2},\ldots,X_1)\prod_{t=n}^{T} p(X_t|X_{t-1},\ldots,X_{t-n+1})
\end{aligned}
$$

Categorical distributions

The n-Gram Model

- Basic tool for language modeling.
- Based on the Markov assumption of order $n-1$:
  - $p(X_t|X_{t-1},\ldots,X_1) = p(X_t|X_{t-1},\ldots,X_{t-n+1})$

- n-gram model:

$$
\begin{aligned}
& p(X_1,\ldots,X_T) \\
& = p(X_1)\ldots p(X_T|X_{T-1},\ldots,X_{T-n+1}) \\
& = P(X_1)\ldots p(X_{n-1}|X_{n-2},\ldots,X_1)\prod_{t=n}^{T} p(X_t|X_{t-1},\ldots,X_{t-n+1})
\end{aligned}
$$

Parameters of the n-gram model

Parameters of the n-Gram Model

- Parameters: values that are required to infer the likelihood of an observation in the model.
  - $p(x_n | x_{n-1}, \dots, x_1) = \frac{p(x_1, \dots, x_n)}{p(x_1, \dots, x_{n-1})}$
- For each m with $1 \leq m \leq n$ and each combination of values $x_1, \dots, x_m$:
  - $p(x_1, \dots, x_m)$ has to be known.
  - Written as $\theta_{x_1, \dots, x_m}$.
- We will infer these parameters from data (e.g., from a text corpus).

The n-Gram Model: Parameter Inference

- Given data from which we determine:
  - For all m with $1 \leq m \leq n$ and each combination of values $x_1, \dots, x_m$ we observe the number of occurances $N_{x_1,\dots,x_m}$.
  - For all m with $1 \leq m \leq n$ we observe the total number N_m of observations.

- What is the relationship between the true, unknown parameters $\theta_{x_1,\dots,x_m}$ and the observed $N_{x_1,\dots,x_m}$?

The n-Gram Model: Parameter Inference

- What is the relationship between the true, unknown parameters $\theta_{x_1,\ldots,x_m}$ and the observed $N_{x_1,\ldots,x_m}$?
  - Each $N_{x_1,\ldots,x_m}$ is a random variable.
  - N_m random experiments,
  - Each $(x_1, \ldots, x_m) \in Y^m$ is a possible outcome that occurs with probability $\theta_{x_1,\ldots,x_m}$.
  - Let $\{y_1, \ldots, y_k\} = Y^m$ be all the possible outcomes
  - What type of distribution is $p(N_{y_1}, \ldots, N_{y_k} | \theta_{y_1}, \ldots, \theta_{y_k})$?

The n-Gram Model: Parameter Inference

- What is the relationship between the true, unknown parameters $\theta_{x_1,\dots,x_m}$ and the observed $N_{x_1,\dots,x_m}$?
  - Each $N_{x_1,\dots,x_m}$ is a random variable.
  - N_m random experiments,
  - Each $(x_1, \dots, x_m) \in Y^m$ is a possible outcome that occurs with probability $\theta_{x_1,\dots,x_m}$.
  - Let $\{y_1, \dots, y_k\} = Y^m$ be all the possible outcomes
- $p(N_{y_1}, \dots, N_{y_k} | \theta_{y_1}, \dots, \theta_{y_k})$ is a multinomial distribution.
  - $p\left(N_{y_1}, \dots, N_{y_k} \middle| \theta_{y_1}, \dots, \theta_{y_k}\right) = \frac{N_m!}{N_{y_1}! \dots N_{y_k}!} \theta_{y_1}^{N_{y_1}} \cdot \dots \cdot \theta_{y_k}^{N_{y_k}}$
    with $N_m = \sum_j N_{y_j}$

The n-Gram Model: ML Parameters

- Likelihood of training data:
  - $$p\left(N_{y_1}, \ldots, N_{y_k} \middle| \theta_{y_1}, \ldots, \theta_{y_k}\right) = M\left[\theta_{y_1}, \ldots, \theta_{y_k}\right](N_{y_1}, \ldots, N_{y_k})$$
    $$= \frac{N_m!}{N_{y_1}! \ldots N_{y_k}!} \theta_{y_1}^{N_{y_1}} \cdot \ldots \cdot \theta_{y_k}^{N_{y_k}}$$
- Maximum-likelihood estimate:
  - $$\arg \max_{\theta_{y_1}, \ldots, \theta_{y_k}} M\left[\theta_{y_1}, \ldots, \theta_{y_k}\right](N_{y_1}, \ldots, N_{y_k})$$
  - $$\theta_{y_i}^{\mathrm{ML}} = \frac{N_{y_i}}{N_m} = \frac{\text{Number of occurrences of } y_i}{\text{Number of observed } n\text{−gram combinations}}$$
- Maximum-likelihood parameters generally not robust, unregularized estimates.

The n-Gram Model: MAP Parameters

- Posterior of training data:
  - $p\left(\theta_{y_1}, \dots, \theta_{y_k} \middle| N_{y_1}, \dots, N_{y_k}\right)$
    $= \frac{1}{p(N_{y_1}, \dots, N_{y_k})} p\left(N_{y_1}, \dots, N_{y_k} \middle| \theta_{y_1}, \dots, \theta_{y_k}\right) p(\theta_{y_1}, \dots, \theta_{y_k})$
- Likelihood $p\left(N_{y_1}, \dots, N_{y_k} \middle| \theta_{y_1}, \dots, \theta_{y_k}\right)$ is a multinomial distribution.
- If prior $p(\theta_{y_1}, \dots, \theta_{y_k})$ follows a Dirichlet distribution, then posterior $p\left(\theta_{y_1}, \dots, \theta_{y_k} \middle| N_{y_1}, \dots, N_{y_k}\right)$ follows a Dirichlet distribution as well.
  - Dirichlet distribution is the conjugate of the multinomial distribution.

The n-Gram Model: MAP Parameters

- Posterior of training data:
  - $p(\theta_{y_1}, \dots, \theta_{y_k} | N_{y_1}, \dots, N_{y_k})$
    $= \frac{1}{p(N_{y_1}, \dots, N_{y_k})} M[\theta_{y_1}, \dots, \theta_{y_k}](N_{y_1}, \dots, N_{y_k}) D[\alpha_{y_1}, \dots, \alpha_{y_k}] p(\theta_{y_1}, \dots, \theta_{y_k})$
    $= \frac{1}{p(N_{y_1}, \dots, N_{y_k})} D[\alpha_{y_1} + \theta_{y_1}, \dots, \alpha_{y_k} + \theta_{y_k}](N_{y_1}, \dots, N_{y_k})$
- Maximum-posterior estimate:
  - $\arg\max_{\theta_{y_1}, \dots, \theta_{y_k}} D[\alpha_{y_1} + \theta_{y_1}, \dots, \alpha_{y_k} + \theta_{y_k}](N_{y_1}, \dots, N_{y_k})$
  - $\theta_{y_i}^{\text{MAP}} = \frac{N_{y_i} + \alpha_{y_i}}{N_m + \sum_y \alpha_y} = \frac{\text{Number of occurrences of } y_i + \alpha_{y_i}}{\text{Number of observed } n\text{−gram combinations} + \sum_y \alpha_y}$
- This form of regularization is also called Laplace smoothing.

Scheffer: Language Technology

Overview

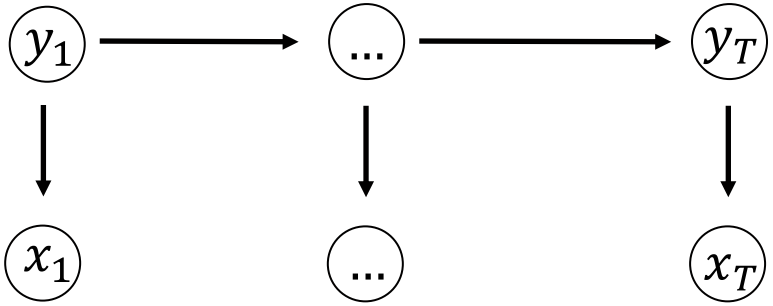
- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.

The Hidden Markov Model

- Directed, generative model.
- Joint probability of input and output:

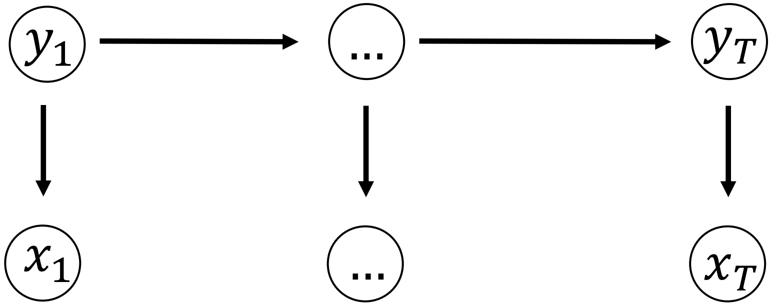
$$P(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = P(x_1, \dots, x_T, y_1, \dots, y_T|\boldsymbol{\theta})$$
$$= \prod_{t=1}^{T} P(y_t|y_{t-1}, \boldsymbol{\theta})\, P(x_t|y_t, \boldsymbol{\theta})$$

- Generative model: parameters optimized to maximize joint likelihood of input and output.



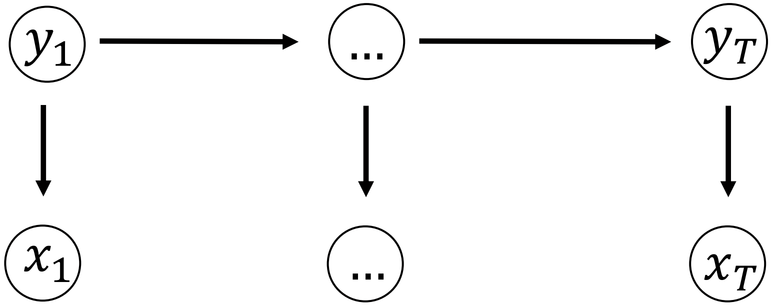
The Hidden Markov Model

- Model parameters in vector $\boldsymbol{\theta} = (\boldsymbol{\pi}, \mathbf{a}, \mathbf{b})$:
  - Starting state probabilities $\pi_i = P(y_1 = i|\boldsymbol{\theta})$
  - Transition probabilities $a_{ij} = P(y_{t+1} = j|y_t = i, \boldsymbol{\theta})$
  - Observation probabilities $b_i(o) = P(x_t = o|y_t = i, \boldsymbol{\theta})$
- Markov assumptions:
  - $P(y_{t+1}|y_1, \ldots, y_t, \boldsymbol{\theta}) = P(y_{t+1}|y_t, \boldsymbol{\theta})$
  - $P(x_t|y_1, \ldots, y_T, \boldsymbol{\theta}) = P(x_t|y_t, \boldsymbol{\theta})$



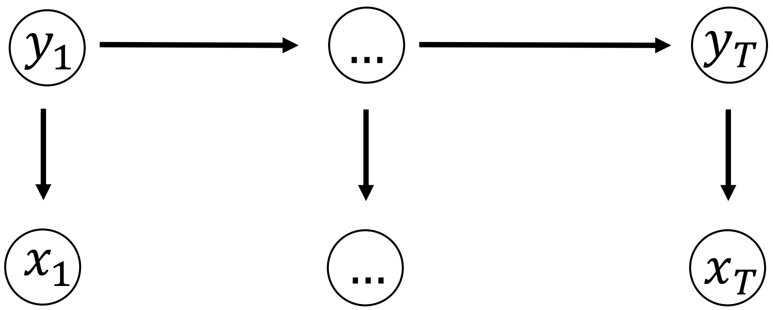
The Hidden Markov Model

- Prediction, inference: most likely output given input
  $$\text{argmax}_{y_1,\dots,y_T} P(y_1, \dots, y_T | x_1, \dots, x_T, \boldsymbol{\theta})$$
- Naive maximization over all combinations of labels; exponentially many in T.
- But can be solved in $O(T)$ with dynamic programming → Viterbi algorithm.



HMM: Inference Tasks

- Find most likely state sequence given output:
  - $\text{argmax}_{y_1,\dots,y_T} P(y_1, \dots, y_T | x_1, \dots, x_T, \boldsymbol{\theta})$
  - Solved in $O(T)$ by the Viterbi algorithm.
- Likelihood of an observation sequence:
  - $P(x_1, \dots, x_T | \boldsymbol{\theta})$
  - Solved in $O(T)$ by the forward-backward algorithm.
- Find most likely state at t given output sequence:
  - $\text{argmax}_{y_t} P(y_t | x_1, \dots, x_T, \boldsymbol{\theta})$
  - Solved in $O(T)$ by the forward-backward algorithm.



Viterbi Algorithm

- Definition:
  $\delta_t(i) = \max_{y_1,\dots,y_{t-1}} P(y_1, \dots, y_{t-1}, y_t = i, x_1, \dots, x_t | \boldsymbol{\theta})$
- It follows that $\sum_i \delta_T(i) = P(\mathbf{x}, \mathbf{y} | \boldsymbol{\theta})$
- Since $\mathbf{x}$ is constant, maximizing $\sum_i \delta_T(i)$ over $\mathbf{y}$ maximizes $P(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta})$.

Viterbi Algorithm

- Definition:
  $$\delta_t(i) = \max_{y_1,\dots,y_{t-1}} P(y_1,\dots,y_{t-1},y_t = i, x_1,\dots,x_t|\boldsymbol{\theta})$$
- Theorem: can be calculated recursively
  $$\delta_1(i) = p(y_1 = i|\boldsymbol{\theta})P(x_1|y_1 = i,\boldsymbol{\theta})$$
  $$\delta_{t+1}(j) = \left(\max_i \delta_t(i)a_{ij}\right)b_j(x_{t+1})$$
- Proof:
  $$\delta_{t+1}(j) = \max_{y_1,\dots,y_t} P(y_1,\dots,y_t,y_{t+1} = j, x_1,\dots,x_{t+1}|\boldsymbol{\theta})$$
  $$= \max_{y_1,\dots,y_t} P(y_1,\dots,y_t,x_1,\dots,x_t|\boldsymbol{\theta})a_{ij}b_j(x_{t+1})$$
  $$= \max_i \max_{y_1,\dots,y_{t-1}} P(y_1,\dots,y_t = i, x_1,\dots,x_t|\boldsymbol{\theta})a_{ij}b_j(x_{t+1})$$
  $$= \max_i \delta_t(i)a_{ij}b_j(x_{t+1})$$

Scheffer: Language Technology

Viterbi Algorithm

- Initialization:
  - $\log \delta_1 (i) = \log \pi_i b_i(x_1)$
  - $\psi_1(i) = 0$
- For $t = 1 \dots T - 1$, for all y':
  - $\log \delta_{t+1}(j) = \left(\max_i \log \delta_t(i) + \log a_{ij}\right) + \log b_j(x_{t+1})$
  - $\psi_{t+1}(j) = \left(\arg\max_i \log \delta_t(i) + a_{ij}\right)$
- Termination: $y_T^* = \arg\max_y \delta_T(y)$
- For $t = T - 1 \dots 1$
  - $y_t^* = \psi_{t+1}(q_{t+1}^*)$
- Return $y_1^*, \dots, y_T^*$.

Forward-Backward Algorithm

- Definitions:
  - $\alpha_t(i) = P(O_1, \dots, O_t, q_t = i|\theta)$
  - $\beta_t(i) = P(O_{t+1}, \dots, O_T|q_t = i, \theta)$
  - $\gamma_t(i) = P(q_t = i|O_1, \dots, O_T, \theta)$
- Relation to inference problems:
  - Likelihood of observation sequence: $P(O_1, \dots, O_T\ |\theta) = \sum_i \alpha_T(i)$.
  - Most likely state given observation sequence: $P(q_t = i|O_1, \dots, O_T, \theta) = \gamma_t(i)$.

Forward Step

- Likelihood of an initial observation sequence:
  - $\alpha_t(i) = P(O_1, \dots, O_t, q_t = i|\theta)$
- Theorem:
  - $\alpha_1(i) = \pi_i b_i(O_1)$
  - $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1})$
- Forward algorithm:
  - `For all i, let` $\alpha_1(i) = \pi_i b_i(O_1)$
  - `For` $t = 1 \dots T-1$`: let`
    $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1})$

Forward Step: Proof

- Proof by induction: base case
  - $\alpha_1(i) = P(O_1, q_1 = i \mid \lambda)$
    $= P(q_1 = i \mid \lambda) P(O_1 \mid q_1 = i, \lambda) = \pi_i b_i(O_1)$
- Induction: $t \rightarrow t+1$

$$\alpha_{t+1}(j) = P(O_1, \ldots, O_{t+1}, q_{t+1} = j \mid \lambda) = \sum_{i=1}^{N} P(O_1, \ldots, O_{t+1}, q_t = i, q_{t+1} = j \mid \lambda)$$

$$= \sum_{i=1}^{N} P(O_1, \ldots, O_t, q_t = i \mid \lambda) P(q_{t+1} = j \mid q_t = i, O_1, \ldots, O_t, \lambda)$$

$$P(O_{t+1} \mid q_{t+1} = j, q_t = i, O_1, \ldots, O_t, \lambda)$$

$$= \sum_{i=1}^{N} P(O_1, \ldots, O_t, q_t = i \mid \lambda) P(q_{t+1} = j \mid q_t = i, \lambda) P(O_{t+1} \mid q_{t+1} = j, \lambda)$$

$$= \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(O_{t+1})$$

Backward Step

- Likelihood of rest of observation sequence:
  - $\beta_t(i) = P(O_{t+1}, \ldots, O_T | q_t = i, \theta)$
- Theorem:
  - $\beta_T(i) = 1$
  - $\beta_t(i) = \left(\sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j)\right)$

- Backward algorithm:
  - `For all i, let` $\beta_T(i) = 1$
  - `For` $t = T-1 \ldots 1$, `for all i: let` $\beta_t(i) = \left(\sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j)\right)$

Backward Step: Proof

- Induction step $t+1 \rightarrow t$

$$\beta_t(i) = P(O_{t+1},\ldots,O_T \mid q_t = i,\lambda)$$

$$= \sum_j P(O_{t+1},\ldots,O_T, q_{t+1} = j \mid q_t = i,\lambda)$$

$$= \sum_j P(O_{t+1},\ldots,O_T \mid q_{t+1} = j, q_t = i,\lambda) P(q_{t+1} = j \mid q_t = i,\lambda)$$

$$= \sum_j P(O_{t+2},\ldots,O_T \mid q_{t+1} = j, q_t = i,\lambda) P(q_{t+1} = j \mid q_t = i,\lambda)$$

$$P(O_{t+1} \mid q_{t+1} = j, q_t = i,\lambda)$$

$$= \sum_j P(O_{t+2},\ldots,O_T \mid q_{t+1} = j,\lambda) P(q_{t+1} = j \mid q_t = i,\lambda) P(O_{t+1} \mid q_{t+1} = j,\lambda)$$

$$= \sum_j \beta_{t+1}(j) a_{ij} b_j(O_{t+1})$$

Forward-Backward Algorithm

- Probability of a state at time t:
  - $\gamma_t(i) = P(q_t = i | O_1, \ldots, O_T, \theta)$
    $= \frac{P(q_t = i, O_1, \ldots, O_T | \theta)}{P(O_1, \ldots, O_T | \theta)}$
    $= \frac{P(q_t = i, O_1, \ldots, O_t, O_{t+1}, O_T | \theta)}{P(O_1, \ldots, O_T | \theta)}$
    $= \frac{P(q_t = i, O_1, \ldots, O_t) P(O_{t+1}, O_T | q_t = i, \theta)}{P(O_1, \ldots, O_T | \theta)}$
    $= \frac{\alpha_t(i) \beta_t(i)}{\sum_i \alpha_T(i)}$

Forward-Backward Algorithm

- Forward pass:
  - For all i, let $\alpha_1(i) = \pi_i b_i(O_1)$
  - For $t = 1 \ldots T-1$, for all i:
    - Let $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1})$
- Backward pass:
  - For all i, let $\beta_T(i) = 1$
  - For $t = T-1 \ldots 1$, for all i:
    - Let $\beta_t(i) = \left(\sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j)\right)$
    - Let $\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_T(i)}$

Learning in Hidden Markov Models

- Focus on discrete observations O_t first.

1. Hidden states are visible during training:
   1. Training data are labeled with states.
   2. For instance, in part-of-speech tagging.
2. Hidden states are not visible during training:
   1. States have to be inferred.
   2. For instance, in speech recognition.
   3. Training by Baum-Welch algorithm.

HMM Learning with Visible States

- Maximum-likelihood parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \mathbf{a}, \mathbf{b})$:
  - $\pi_i = P(y_1 = i|\boldsymbol{\theta}) = \frac{\#\text{ sequences that start in state } i}{\#\text{sequences}}$
  - $a_{ij} = P(y_{t+1} = j|y_t = i, \boldsymbol{\theta}) = \frac{\#\text{ state transitions from } i \text{ to } j}{\#\text{state transitions from } i}$
  - $b_i(o) = P(x_t = o|y_t = i, \boldsymbol{\theta}) = \frac{\#\text{ of times } x_t{=}o \text{ in state } i}{\#\text{ of times in state } i}$

- Laplace correction (regularization): add constant to numerator, change denominator so that probabilities add to one again.

HMM Learning with Invisible States

- Start with initial, random model
- Iteratively:
  - use forward backward to estimate state probabilities
  - Update model parameters based on estimated states.

- Baum-Welch algorithm is variant of the EM algorithm for the hidden markov model.

Baum-Welch Algorithm

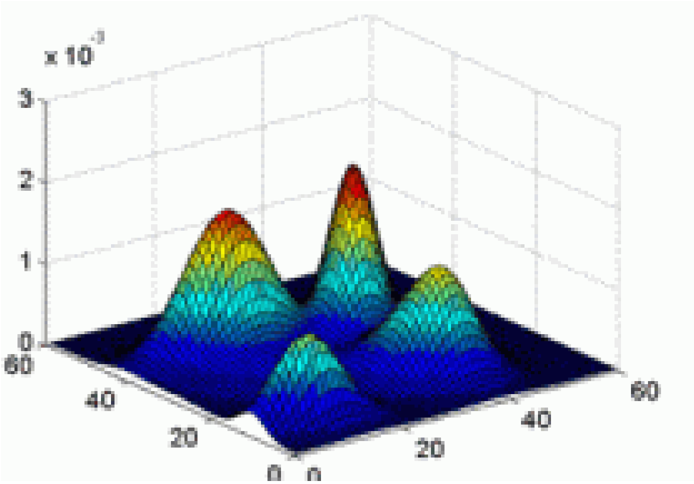
- Definition: probability of a transition from state i to state j at time t:
  - $\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O_1, \dots, O_T, \theta)$
- Can be calculated as:
  - $$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O_1, \dots, O_T, \theta) = \frac{\alpha_t(i) a_{ij} \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_T(i)}$$

Baum-Welch Algorithm

- Input: Set of observation sequences $\{(O_1^1, \dots, O_T^1), \dots, (O_1^m, \dots, O_T^m)$
- Initialize θ at random
- Repeat until convergence:
  - Run forward-backward algorithm to infer all $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$.
  - For all t, i, j: infer $\xi_t(i,j)$
  - For all i: let $\pi_i = \gamma_1(i)$
  - For all i, j: let $a_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}$
  - For all i, O: let $b_i(O) = \frac{\sum_{t:O_t=O} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$

Continuous Hidden Markov Models

- So far, $b_i(O_t)$ has been a multinomial distribution over a set of discrete observations.
- In speech processing, O_t is a vector of continuous attributes.
- Modeling assumption: mixture of Gaussian distributions.
  - $b_i(\mathbf{x}_t) = \sum_{k=1}^{M} c_{ik}\, N[\mu_{ik}, \Sigma_{ik}](\mathbf{x}_t) = \sum_{k=1}^{M} c_{ik}\, b_{ik}(\mathbf{x}_t)$



Continuous Hidden Markov Models

- Modeling assumption: mixture of Gaussian distributions.
  - $b_i(\mathbf{x}_t) = \sum_{k=1}^{M} c_{ik}\, N[\mu_{ik}, \Sigma_{ik}](\mathbf{x}_t) = \sum_{k=1}^{M} c_{ik}\, b_{ik}(\mathbf{x}_t)$
- Parameters of multivariate Gaussian distributions $b_{ik}(\mathbf{x}_t)$ have to be estimated from data.
- Mixing coefficients c_{ik} are unknown; estimated using EM algorithm.
- Each iteration of the Baum-Welch algoríthm requires an execution of the EM algorithm to estimate mixing coefficients c_{ik}.

The Hidden Markov Model

- Generative sequence model.
- Observed output variable, latent state variable.
- Latent states, state sequences can be inferred by Viterbi-/forward-backward-algorithm.
- States visible in labeled training data? Model parameters estimated by “regularized counting”.
- States invisible in labeled training data? Model parameters estimated by Baum-Welch algorithm.
- Continuous observations usually modeled as (mixture of) multivariate Gaussian distributions.

Overview

- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.

Classification

- Input: an instance $\mathbf{x} \in X$
  - E.g., X can be a vector space over *attributes*
  - The Instance is then an assignment of attributes.
  - $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ is a feature vector
- Output: Class $y \in Y$; where Y is a finite set.
  - The class is also referred to as the *target* attribute
  - y is also referred to as the (class) *label*

$$\mathbf{x} \longrightarrow \textbf{classifier} \longrightarrow y$$

Classification: Example

- Input: Instance $\mathbf{x} \in X$
  - X : the set of all possible combinations of regiment of medication

Attribute	Instance $\mathbf{x}$
Medication #1 included?	0
	1
⋮	1
	0
	1
Medication #6 included?	0

Attribute values
Feature vector

Medication combination

- Output: $y \in Y = \{\text{toxic}, \text{ok}\}$  / 

 $\longrightarrow$ **classifier** $\longrightarrow$ 

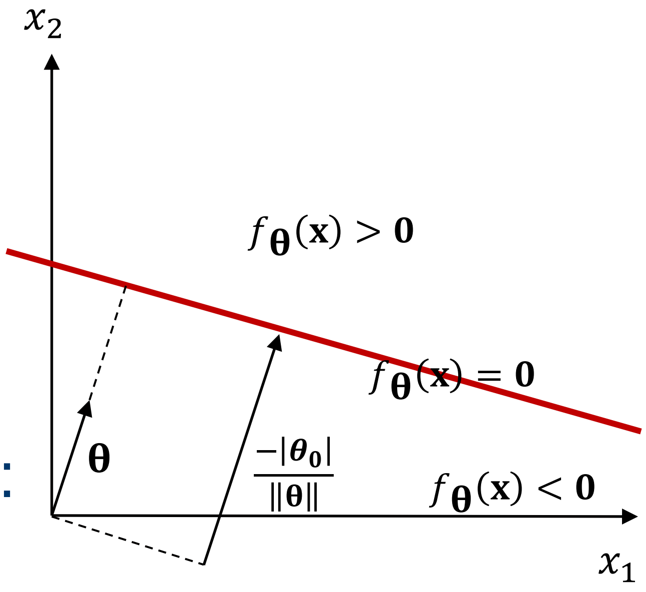
Linear Classification Models

- Hyperplane given by normal vector & displacement:
$$H_{\boldsymbol{\theta}} = \{\mathbf{x} | f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \theta_0 = 0\}$$

- Example: $X = \mathbb{R}^2$

- Decision function:
$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \theta_0$

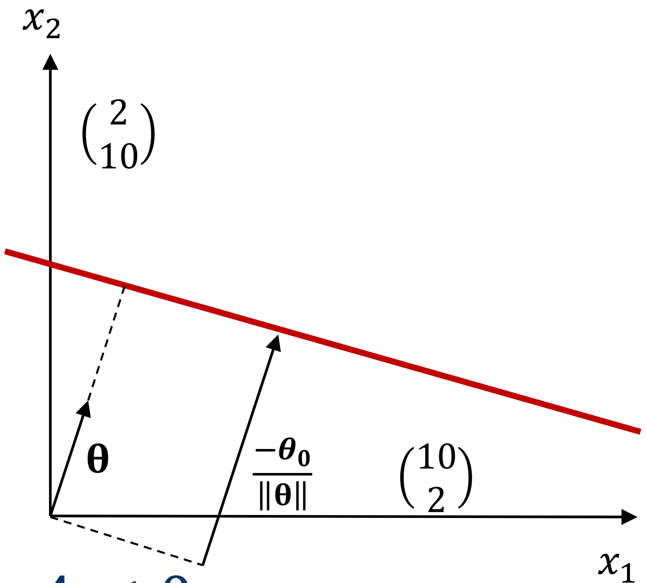
- Binary classifier, $y \in \{+1, -1\}$:
$y_{\boldsymbol{\theta}}(\mathbf{x}) = \mathrm{sign}\big(f_{\boldsymbol{\theta}}(\mathbf{x})\big)$



Linear Classification Models

- Hyperplane given by normal vector & displacement:

$$H_{\boldsymbol{\theta}} = \{\mathbf{x} | f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \theta_0 = 0\}$$



- Example: $X = \mathbb{R}^2$
- Decision function:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\begin{pmatrix}1\\3\end{pmatrix} - 20$$

- Example points:

$$f_{\boldsymbol{\theta}}\begin{pmatrix}10\\2\end{pmatrix} = (10 \quad 2)\begin{pmatrix}1\\3\end{pmatrix} - 20 = -4 < 0$$

$$f_{\boldsymbol{\theta}}\begin{pmatrix}2\\10\end{pmatrix} = (2 \quad 10)\begin{pmatrix}1\\3\end{pmatrix} - 20 = 12 > 0$$

Linear Classification Model

- Offset can “disappear” into parameter vector.
- Example
  - Before: $f_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 1 \\ 3 \end{pmatrix} - 20$
  - After: $f_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -20 \\ 1 \\ 3 \end{pmatrix}$
- New constant attribute $x_0 = 1$ added to all instances
- Offset θ_0 integrated into $\boldsymbol{\theta}$.

Learning Linear Classifiers

- Input to the Learner: Training data T_n.
  - $\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$  
  - $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$  
- Training Data:
  $T_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- Output: a Model
  -  $\mapsto$ 
    $y_{\boldsymbol{\theta}} : X \to Y$
  - Linear classifier:
    $y_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{cases} \text{[sad]} & \text{if } \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} \geq 0 \\ \text{[happy]} & otherwise \end{cases}$  

Linear classifier with parameter vector $\boldsymbol{\theta}$.

Regularized Empirical Risk Minimization

- Solve

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Loss function $\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$: cost of the model's output $f_{\boldsymbol{\theta}}(\mathbf{x})$ when the true value is y.
  - The empirical risk is $\hat{R}_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$
- Regularizer $\Omega(\boldsymbol{\theta})$ & trade-off parameter $\lambda \geq 0$:
  - Background information about preferred solutions
  - Provides numerical stability (Tikhonov-Regularizer)
  - allows for tighter error bounds (PAC-Theory)

Regularized Empirical Risk Minimization

- Solve

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(f_{\mathbf{\theta}}(\mathbf{x}_i), y_i) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Linear model:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(\mathbf{x}_i{}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

Regularized Empirical Risk Minimization

- Linear model: solve

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}} \boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- How to find solution:
  - Classification: No analytic solution but numeric solutions (gradient descent, cutting plane, interior point method)

Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Gradient:
  - Vector of the derivatives with respect to each individual parameter
  - Direction of the steepest increase of the function $L(\boldsymbol{\theta})$.

$$\nabla L(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_m} \end{pmatrix}$$

Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Gradient descent method:

RegERM(*Data:* $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$)

  Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$

  DO

    Compute gradient $\nabla L(\boldsymbol{\theta}^t)$

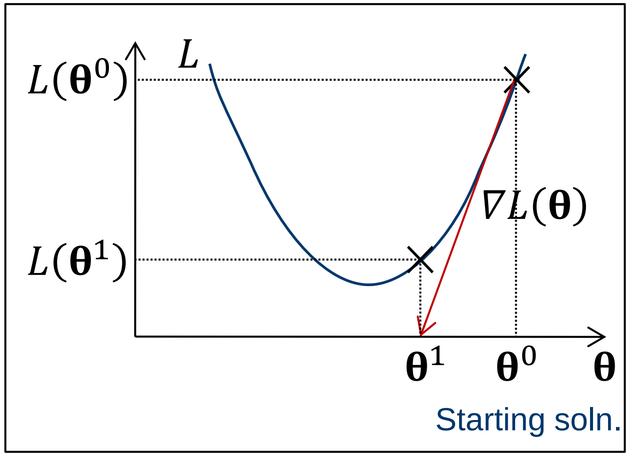
    Compute step size α^t

    Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla L(\boldsymbol{\theta}^t)$

    Set $t = t + 1$

  WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$

RETURN $\boldsymbol{\theta}^t$



Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

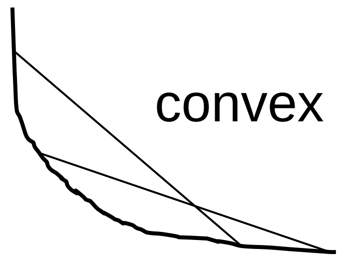
- Gradient descent method:

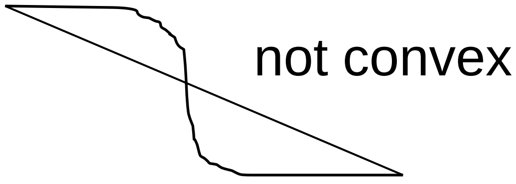
RegERM(*Data:* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$)
    Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$
    DO
        Compute gradient $\nabla L(\boldsymbol{\theta}^t)$
        Compute step size α^t
        Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla L(\boldsymbol{\theta}^t)$
        Set $t = t + 1$
    WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$
RETURN $\boldsymbol{\theta}^t$

- The step size α^t can be determined through
  - Line search
  - Barzilai-Borwein method
  - …

Regularized Empirical Risk Minimization

- Properties of the gradient descent method:
  - Optimization criterion improved with every step.
  - Converges to the global minimum of the optimization criterion when this criterion is convex.
- The sum of convex functions is convex.
- Therefore, optimization criterion is convex if
  - Loss function is convex and
  - Regularizer is convex





ERM: Stochastic Gradient Method

- Idea: Determine the gradient for a random subset of the samples (e.g., a single instance).
- Less computation per optimization step, but only approximate descent direction.
- Optimization criterion with regularizer in sum:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \frac{\lambda}{n} \Omega(\boldsymbol{\theta}) \right]$$

- Stochastic gradient for a single instance:

$$\nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \frac{\lambda}{n} \frac{\partial}{\partial \boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$$

ERM: Stochastic Gradient Method

- Approximate gradient using single examples.

RegERM-Stoch(*Data:* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$)

Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$

DO

Shuffle data randomly

FOR $i = 1, \ldots, n$

Compute subset gradient $\nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}^t)$

Compute step size α^t

Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}^t)$

Set $t = t + 1$

END

WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$

RETURN $\boldsymbol{\theta}^t$

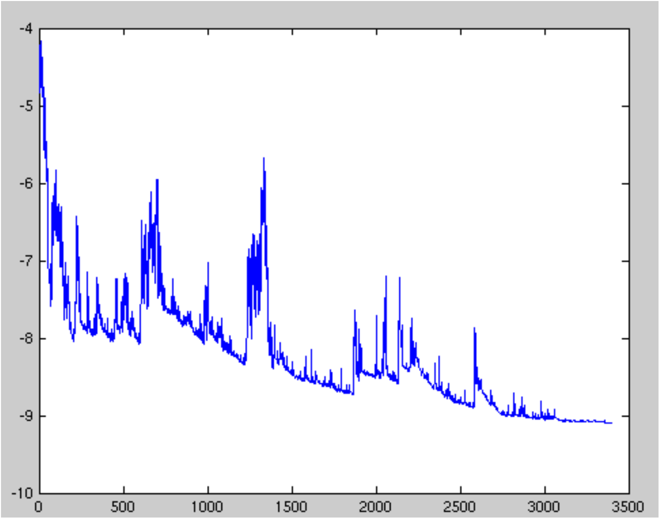
ERM: Stochastic Gradient Method

- In every step only one summand of the optimization criterion is improved.
- The total optimization criterion can be worsened by these individual steps.

- Converges to the optimum if the step sizes satisfy:

  $\sum_{t=1}^{\infty} \alpha^t = \infty$ and $\sum_{t=1}^{\infty} (\alpha^t)^2 < \infty$

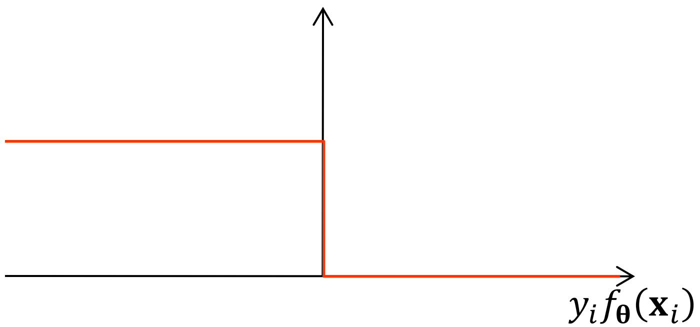
  (Robbins & Monro, 1951)



ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & \overbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ 0 & \underbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$

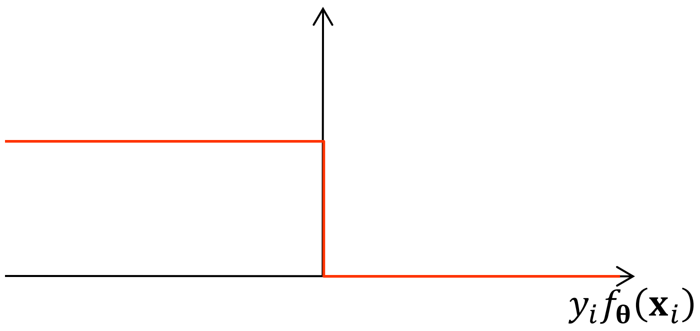


ERM: Loss Functions for Classification

Zero-one loss is not convex
⇒ difficult to minimize!

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & \overbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ 0 & \underbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$



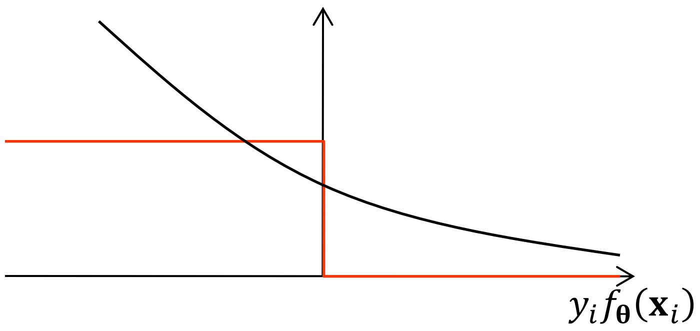
ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & \overbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ 0 & \underbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$

- Logistic loss:

$$\ell_{log}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)}\right)$$



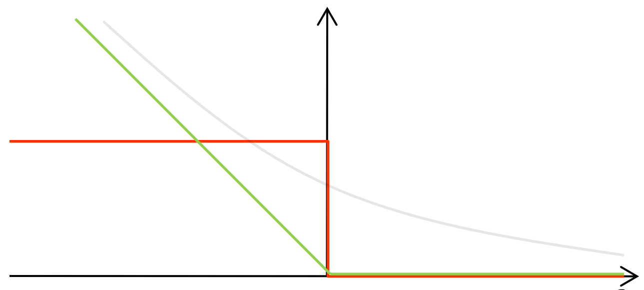
ERM: Loss Functions for Classification

- Zero-one loss:

$\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i$

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases}$$

$\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i$



- Logistic loss:

$$\ell_{log}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)}\right)$$

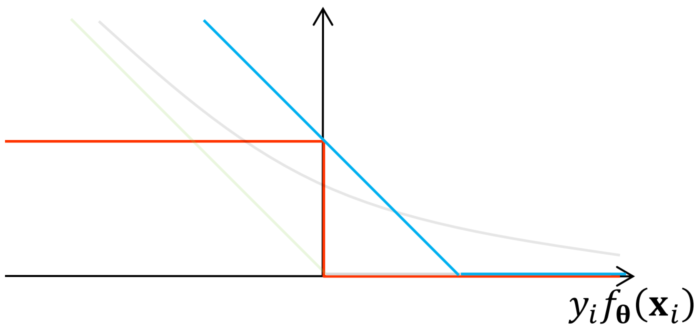
- Perceptron loss:

$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$

ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & \overbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ 0 & \underbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\text{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$



- Logistic loss:

$$\ell_{log}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)}\right)$$

- Perceptron loss:

$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$

- Hinge loss:

$$\ell_h(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$

ERM: Regularizers for Classification

- Idea: use as few attributes as possible:
  - $\Omega_0(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_0 =$ number of j with $\theta_j \neq 0$

Ω_0 is not convex $\Rightarrow$ difficult to minimize!

- Manhattan norm (encourages scarcity):

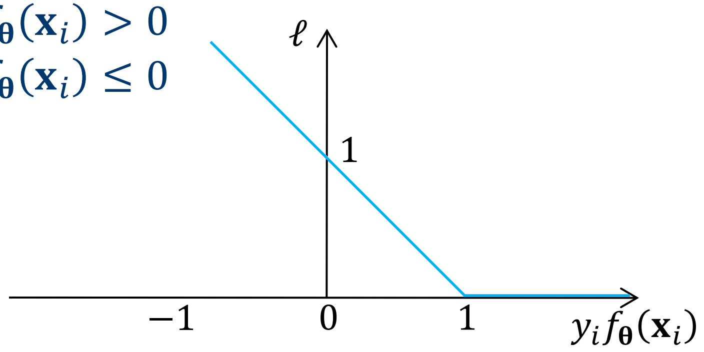
  $\Omega_1(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{m} |\theta_j|$

- Squared Euclidean norm (encourages small weights):
  - $\Omega_2(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_2^2 = \sum_{j=1}^{m} \theta_j{}^2$

Scheffer: Language Technology

ERM: Support Vector Machine (SVM)

- Class $y \in \{-1, +1\}$
- Loss function:
  - $\ell_h(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & \text{if } 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & \text{if } 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases}$
    $= \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$



- Regularizer:
  - $\Omega_2(\boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} = \sum_{j=1}^{m} |\theta_j|^2 = \|\boldsymbol{\theta}\|_2^2$

Support Vector Machine (SVM)

- Loss function is 0, if…

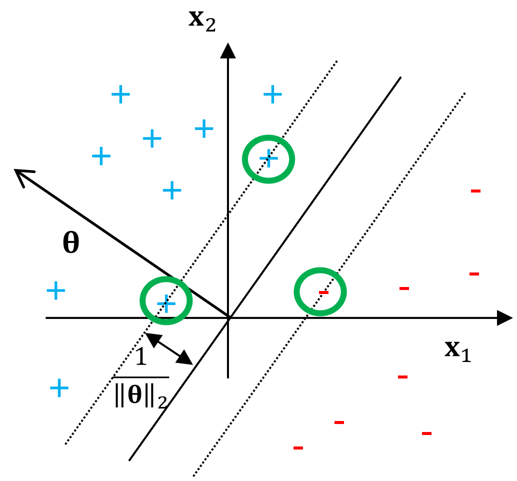
$$\sum_{i=1}^{n} \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) = 0$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq 1$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i {\mathbf{x}_i}^{\mathrm{T}} \boldsymbol{\theta} \geq 1$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i {\mathbf{x}_i}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2} \geq \frac{1}{\|\boldsymbol{\theta}\|_2}$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: {\mathbf{x}_i}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2} \begin{cases} \geq \frac{1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = +1 \\ < \frac{-1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = -1 \end{cases}$$

Hessian normal form: normal vector has length 1

Support Vector Machine (SVM)

- Loss function is 0, if…

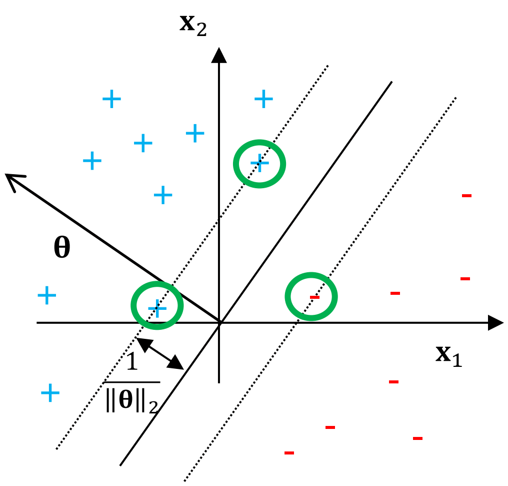
$$\sum_{i=1}^{n} \max\left(0,1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) = 0$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq 1$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i {\mathbf{x}_i}^{\mathrm{T}} \boldsymbol{\theta} \geq 1$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i {\mathbf{x}_i}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2} \geq \frac{1}{\|\boldsymbol{\theta}\|_2}$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: {\mathbf{x}_i}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2} \begin{cases} \geq \frac{1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = +1 \\ < \frac{-1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = -1 \end{cases}$$



Distance of closest point to plane (margin)

- For loss to be 0, all training samples must
  - lie on the correct side of separating plane,
  - and have a minimal margin (gap) of $\frac{1}{\|\boldsymbol{\theta}\|_2}$ to the plane.

Support Vector Machine (SVM)

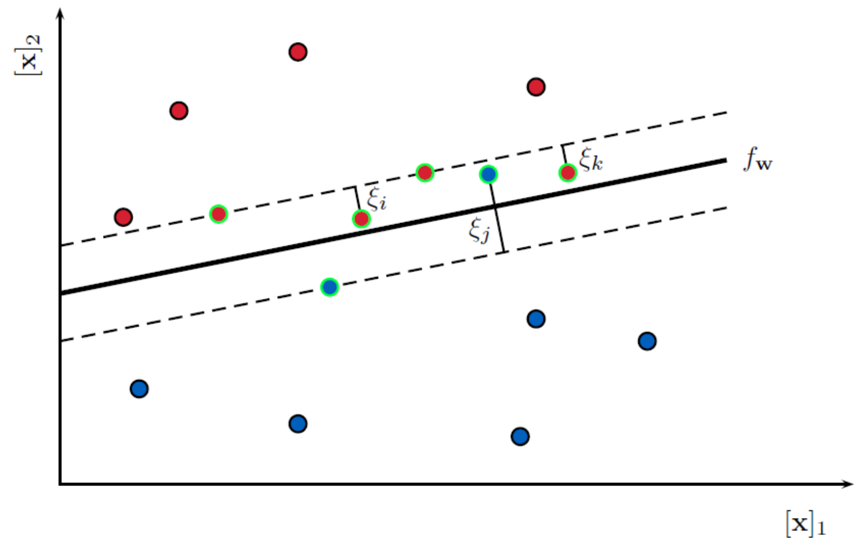


- Loss function is 0 if all training samples have a margin of at least $\frac{1}{\|\boldsymbol{\theta}\|_2}$.
  - Points that lie $\frac{1}{\|\boldsymbol{\theta}\|_2}$ from the plane are *support vectors*
- Regularizer
  - $\Omega_2(\boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} = \|\boldsymbol{\theta}\|_2^2$; is zero only if $\boldsymbol{\theta} = \mathbf{0}$
  - Minimizing $\Omega_2(\boldsymbol{\theta}) \Leftrightarrow$ maximizing margin $\frac{1}{\|\boldsymbol{\theta}\|_2}$
- SVM is also referred to as a *large margin classifier* because its optimization criterion is minimized by the plane with the largest margin from any sample.

Support Vector Machine (SVM)

- If loss function >0, some instances violate margin.
- Loss function as a sum of slack terms
  - $\sum_{i=1}^{n} \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) = \sum_{i=1}^{n} \xi_i$
    $\xi_i = \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$

Slack term or margin violation



- Points with non-zero slack are *support vectors*

Support Vector Machine (SVM)

- Minimize hinge loss and L2-norm $\|\boldsymbol{\theta}\|_2^2 = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}$ of the parameter vector.
- Hinge loss is positive for a sample if the sample has a distance (margin) of less than $\frac{1}{\|\boldsymbol{\theta}\|_2}$ to the separating hyperplane.
- SVM thereby finds the hyperplane with the greatest margin that separates the most possible samples. It trades off between
  - The size of the margin $\frac{1}{\|\boldsymbol{\theta}\|_2}$
  - And the sum of the slack errors $\sum_{i=1}^{n} \xi_i$

Support Vector Machine (SVM)

- Linear classification model: minimize

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\max\left(0, 1 - y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta}\right) + \frac{\lambda}{n} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta} \right]$$

- Gradient:

$$\nabla L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \nabla_{\mathbf{x}_i} L(\boldsymbol{\theta})$$

- Stochastic gradient for $\mathbf{x}_i$:

$$\nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}) = \left\{ \right.$$

Support Vector Machine (SVM)

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\max\left(0, 1 - y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta}\right) + \frac{\lambda}{n} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta} \right]$$

- Gradient:

$$\boldsymbol{\nabla L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \boldsymbol{\nabla}_{\mathbf{x}_i} \boldsymbol{L}(\boldsymbol{\theta})$$

- Stochastic gradient for $\mathbf{x}_i$:

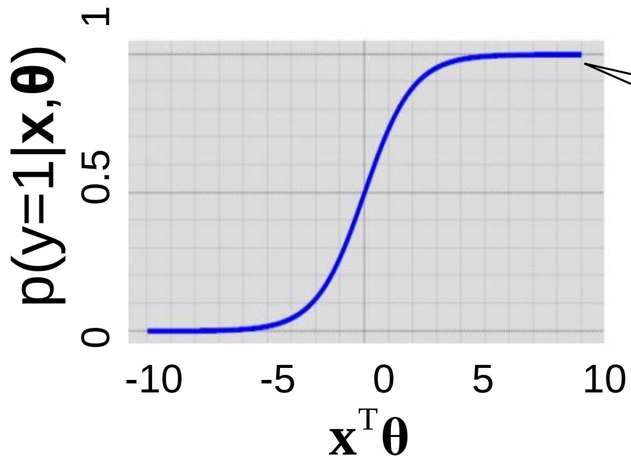
$$\boldsymbol{\nabla}_{\mathbf{x}_i} \boldsymbol{L}(\boldsymbol{\theta}) = \begin{cases} \dfrac{2\lambda}{n} \boldsymbol{\theta} & \text{if } y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta} > 1 \\ \dfrac{2\lambda}{n} \boldsymbol{\theta} - y_i \mathbf{x}_i & \text{if } y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta} < 1 \end{cases}$$

Support Vector Machine (SVM)

- $\boldsymbol{L}(\boldsymbol{\theta})$ can be minimized using stochastic gradient descent method (“Pegasos”)
  - Very fast, often used in practice
- $\boldsymbol{L}(\boldsymbol{\theta})$ can be minimized using gradient descent method (“Primal SVM”)

Logistic Regression

- „Logistic regression“ is a model for classification!
- For now, binary classification with $y_i \in \{-1,1\}$.
- Need: model for $p(y \mid \mathbf{x}, \boldsymbol{\theta})$
  - Model defines probability $p(y=1 \mid \mathbf{x}, \boldsymbol{\theta})$.
  - Probability $p(y=-1 \mid \mathbf{x}, \boldsymbol{\theta}) = 1 - p(y=1 \mid \mathbf{x}, \boldsymbol{\theta})$.
- Idea: transformation of a linear model $\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}$.

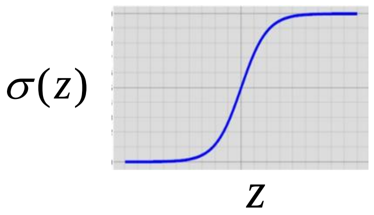


Sigmoid function („Squashing function“) maps interval $[-\infty, \infty]$ to [0,1].

Logistic Regression

- Model logistic regression
  - Given by parameter vector $\boldsymbol{\theta} \in \mathbb{R}^m$.
  - Defines conditional distribution $p(y \mid \mathbf{x}, \boldsymbol{\theta})$ by

$$p(y=1 \mid \mathbf{x}, \boldsymbol{\theta}) = \sigma(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}) = \frac{1}{1+\exp(-\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta})}$$



$$p(y=-1 \mid \mathbf{x}, \boldsymbol{\theta}) = 1 - p(y=1 \mid \mathbf{x}, \boldsymbol{\theta})$$

- Prediction function $f_{\boldsymbol{\theta}} : \mathbb{R}^m \rightarrow \{0,1\}$:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{cases} 1: & \sigma(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}) \geq 0.5 \\ 0: & \text{sonst} \end{cases}$$

Learning Logistic Regression Models

- MAP model: minimize regularized loss.

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

$$= \arg\min_{\boldsymbol{\theta}} \underbrace{\sum_{i=1}^{n} \log\left(1 + \exp(-y_i \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\theta})\right)}_{\text{loss function}} + \underbrace{\frac{1}{2\sigma_p^2} |\boldsymbol{\theta}|^2}_{\text{regularizer}}$$

- Convex optimization problem, global minimum.
- Compare earlier lecture on „Linear models“.

Overview

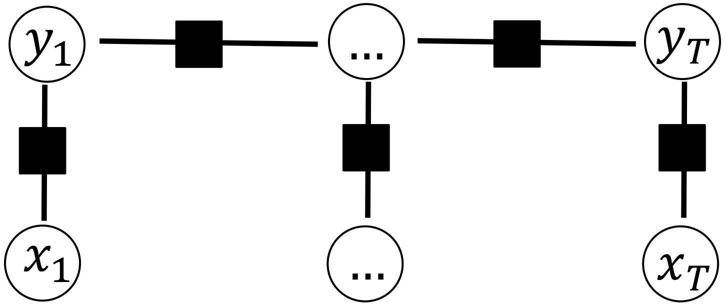
- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.

Undirected Sequence Models

- Factorization:

$$P(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{t=1}^{T} \exp\{\psi(y_t, y_{t+1}) + \psi(y_t, x_t)\}$$

$$= \frac{1}{Z}\prod_{t=1}^{T} \exp\left\{ \begin{array}{l} \sum_{i,j} \theta_{ij}[y_t = i, y_{t+1} = j] \\ + \sum_{i,o} \theta_{io}[y_t = i, x_t = o] \end{array} \right\}$$



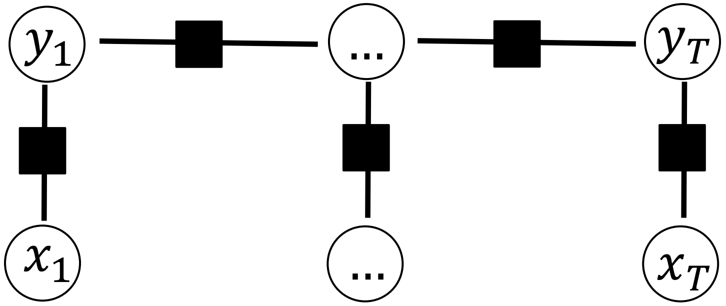
Undirected Sequence Models

- Factorization:

$$P(\mathbf{x},\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z}\prod_{t=1}^{T}\exp\{\psi(y_t,y_{t+1}) + \psi(y_t,x_t)\}$$

$$= \frac{1}{Z}\prod_{t=1}^{T}\exp\left\{\begin{array}{l}\sum_{i,j}\theta_{ij}[y_t = i, y_{t+1} = j] \\ +\sum_{i,o}\theta_{io}[y_t = i, x_t = o]\end{array}\right\}$$

$$= \frac{1}{Z}\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y})\}$$



Undirected Sequence Models

- Conditional random field:

$$P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \frac{P(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})}{\sum_{\mathbf{y}'} P(\mathbf{x},\mathbf{y}'|\boldsymbol{\theta})}$$

$$= \frac{1}{Z}\prod_{t=1}^{T} \exp\{\psi(y_t, y_{t+1}) + \psi(y_t, x_t)\}$$

$$= \frac{1}{Z}\prod_{t=1}^{T} \exp\left\{\begin{array}{c}\sum_{i,j} \theta_{ij}[y_t = i, y_{t+1} = j] \\ + \sum_{i,o} [y_t = i, x_t = o]\end{array}\right\}$$

$$= \frac{\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y})\}}{\sum_{\mathbf{y}'} \exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y}')\}}$$

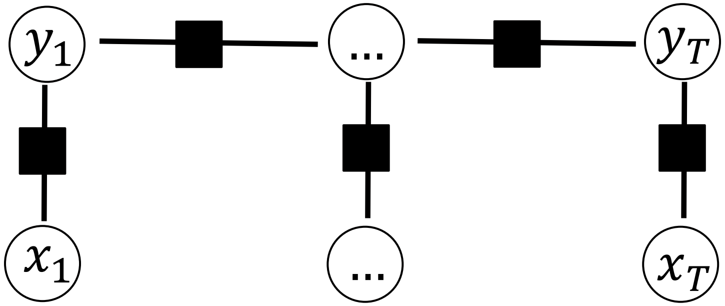
Undirected Sequence Models

- Factorization:

$$P(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z}\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x}, \mathbf{y})\}$$

- With

$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \sum_t [y_t = 1, y_{t+1} = 1] \\ \vdots \\ \sum_t [y_t = N, y_{t+1} = N] \\ \sum_t [x_t = o_1, y_t = 1] \\ \vdots \\ \sum_t [x_t = o_M, y_t = N] \end{pmatrix}$$



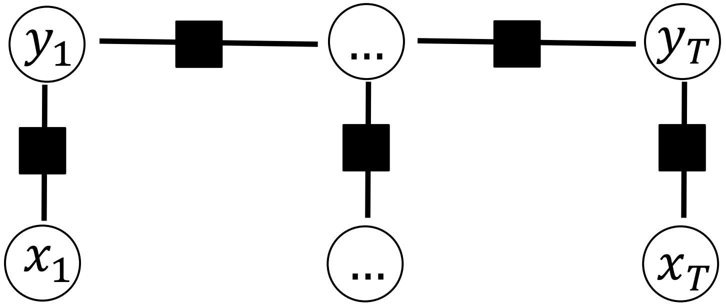
Undirected Sequence Models

- Decision function:

$$f_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}, \mathbf{y})$$

- With

$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \sum_t [y_t = 1, y_{t+1} = 1] \\ \vdots \\ \sum_t [y_t = N, y_{t+1} = N] \\ \sum_t [x_t = o_1, y_t = 1] \\ \vdots \\ \sum_t [x_t = o_M, y_t = N] \end{pmatrix}$$



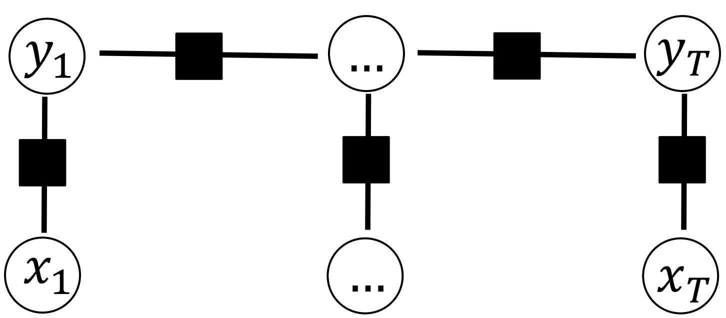
Undirected Sequence Models: Prediction

- Conditional random field: most likely output

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \arg\max_{\mathbf{y}} \frac{1}{Z}\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y})\}$$
$$= \arg\max_{\mathbf{y}} \boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y})$$

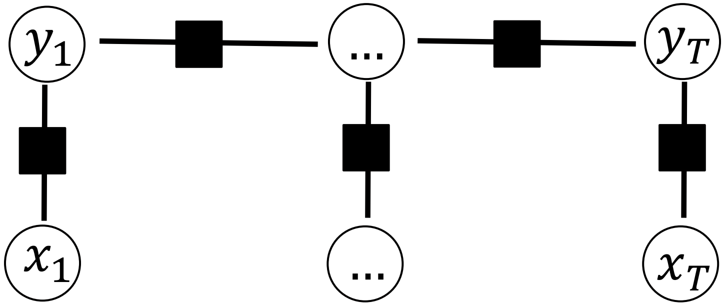
- Decision function:

$$\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}} f_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) = \boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y})$$



Undirected Sequence Models: Prediction

- $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}} f_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}, \mathbf{y})$
- Maximization with variant of Viterbi allgorithm.
  - Scores instead of log-probabilities

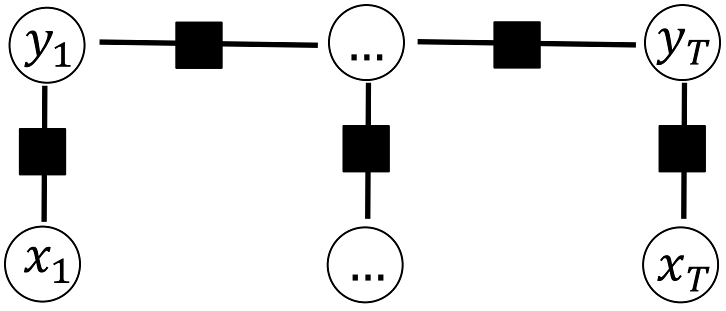


Conditional Random Fields: Inference

- Conditional random field: probability

$$P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \frac{\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y})\}}{\sum_{\mathbf{y}'}\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y}')\}}$$

- Denominator $\sum_{\mathbf{y}'}\exp\{\boldsymbol{\theta}^{\mathrm{T}}\Phi(\mathbf{x},\mathbf{y}')\}$ inferred by variant of the Viterbi algorithm.



Conditional Random Field: Learning

- Optimization criterion

$$\arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\theta})\, P(\boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}_i, \mathbf{y}_i) + \Omega(\boldsymbol{\theta})$$

- Optimization for instance by stochastic gradient descent.
- Requires labeled training data ($\mathbf{y}_i$ visible).

Overview

- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.

PCFG: Definitions

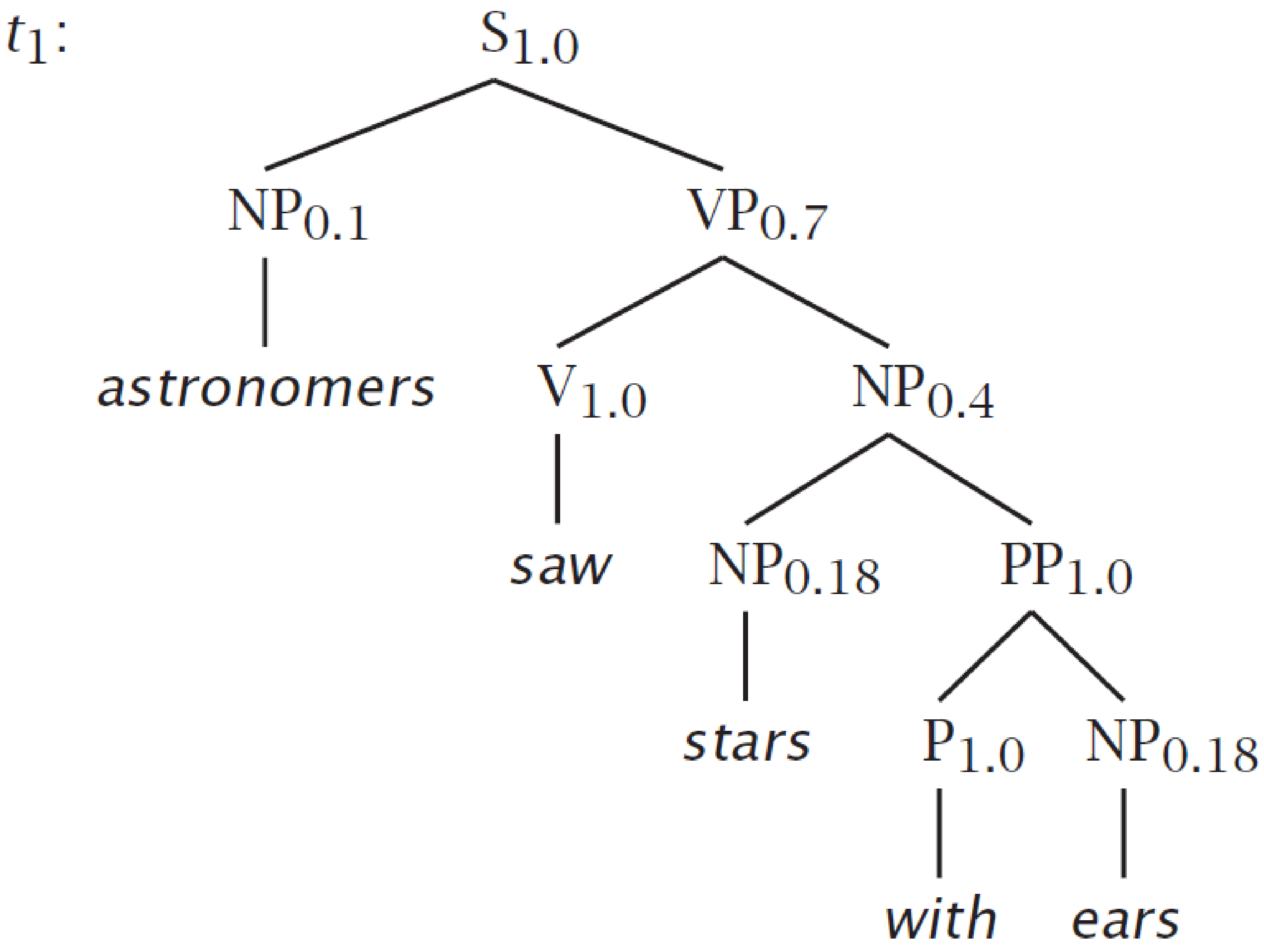
- Terminal symbols: $\{w^k\}, k = 1 \dots v$.
- Nonterminal symbols: $\{N^i\}, i = 1 \dots n$.
- Starting symbol N^1.
- Productions: $\{N^i \rightarrow \xi^j\}$, whete ξ^j is a sequence of terminals and nonterminals.
- Probabilities $P(N^i \rightarrow \xi^j)$, such that for all i: $\sum_j P(N^i \rightarrow \xi^j) = 1$.

PCFG: Definitions

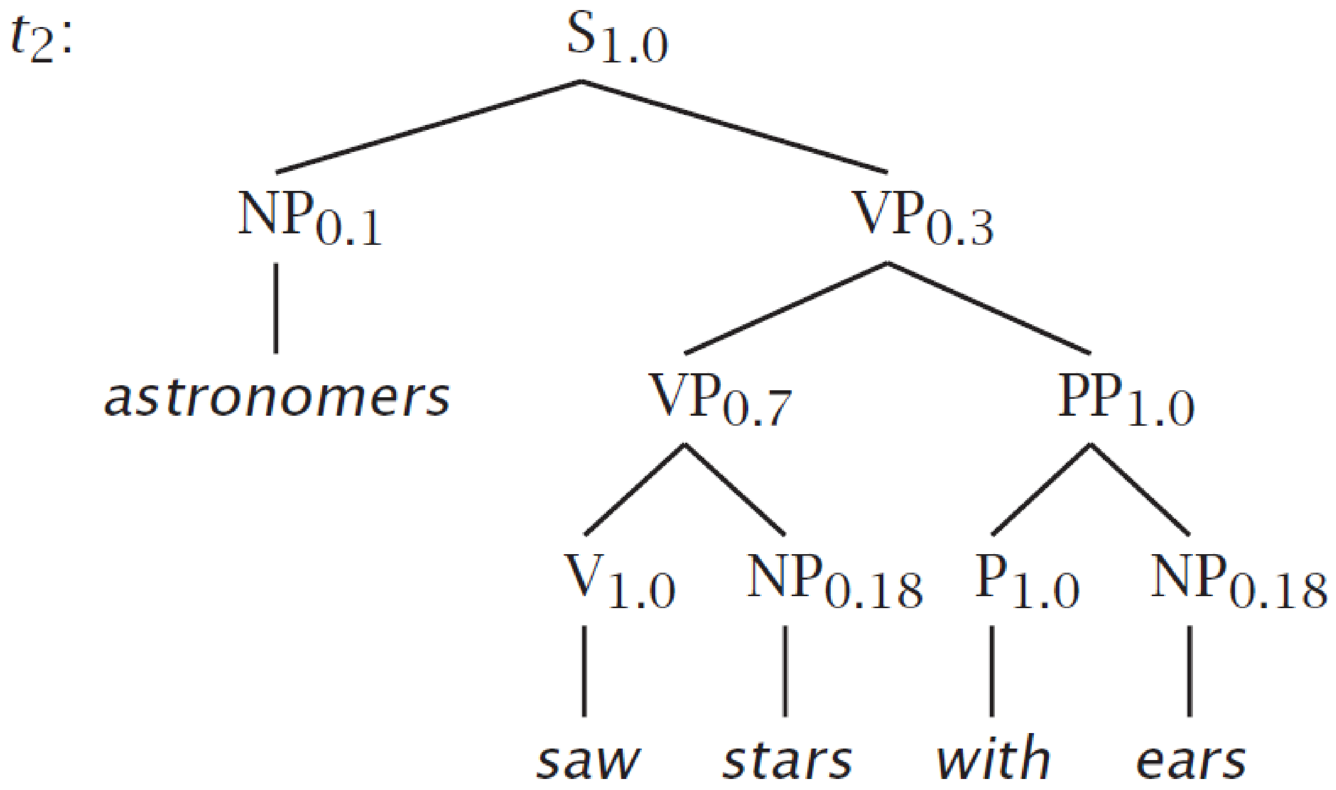
- Let $w_s, \dots, w_e$ be a sentence.
- $N^i \Rightarrow^* w_s, \dots, w_e$ if $w_s, \dots, w_e$ can be derived from N^i.
- N^i_{se}: derivation of $w_s, \dots, w_e$ from N^i.
- Example PCFG:

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

PCFG: Example Parse



PCFG: Example Parse



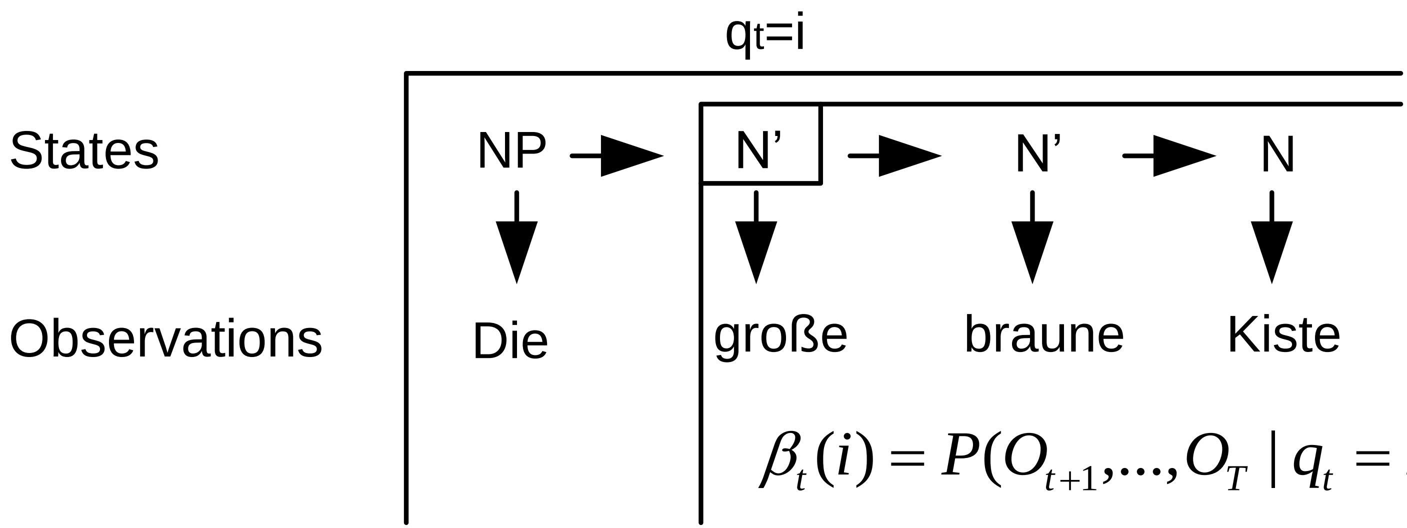
PCFG: Inference

- PCFGs answer three questions (much like HMMs):
  - What is the likelihood of a sentence given a PCFG?
  - What is the most likely parse tree?
  - What are the most likely PCFG parameters given a corpus of sentences and parse trees.

Chomsky Normal Form

- For each context-free grammar, there is a context-free grammar in Chomsky normal form.
- Chomsky normal form has two types of productions:
  - $N^i \rightarrow N^j N^k$
  - $N^i \rightarrow w^j$
- Chomsky normal form allows a simplified treatment.

Forward / Backward – Inside / Outside



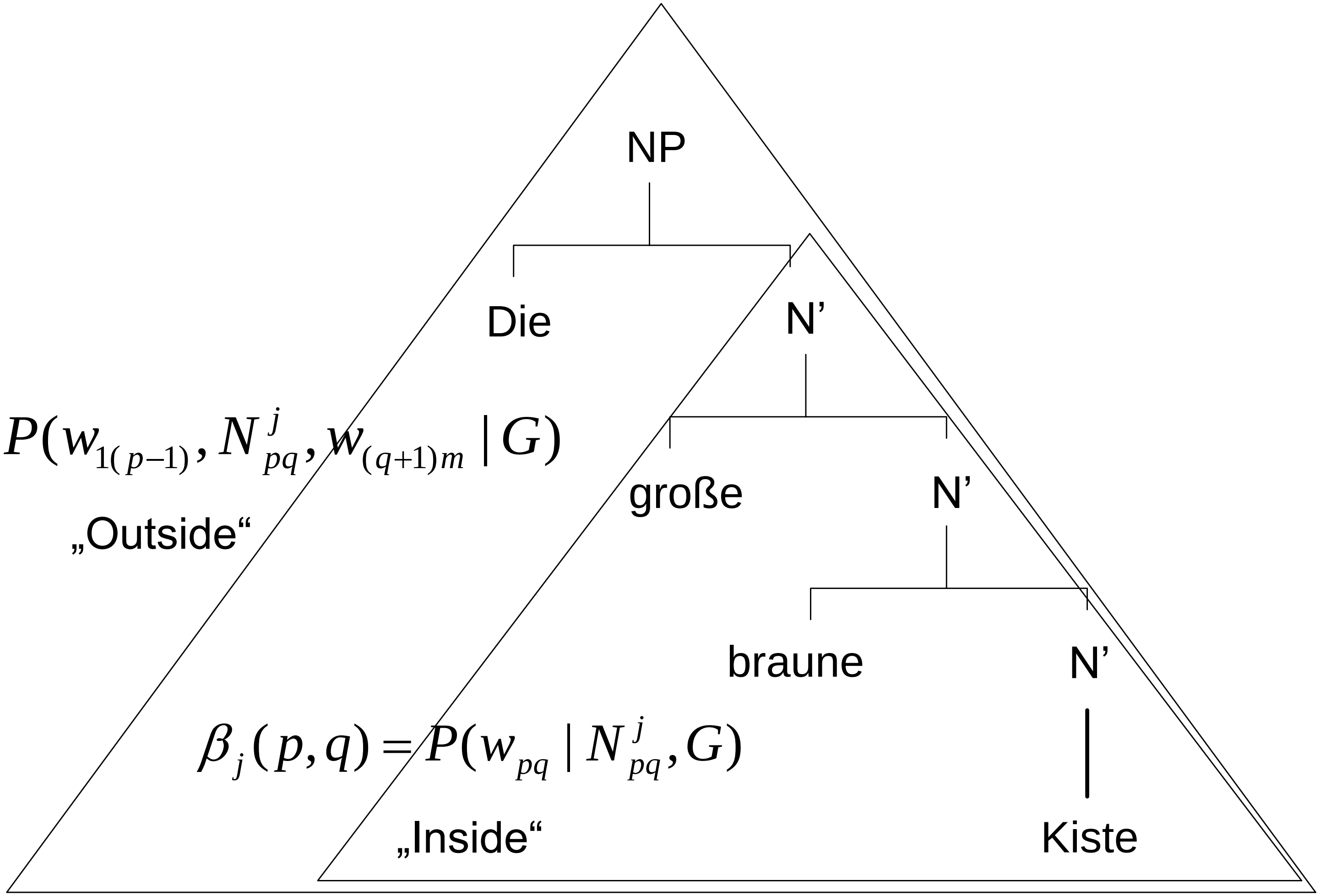
$$\beta_t(i) = P(O_{t+1},\ldots,O_T \mid q_t = i, \lambda)$$

„Backward“

$$\alpha_t(i) = P(O_1,\ldots,O_t, q_t = i \mid \lambda)$$

„Forward“

Forward / Backward – Inside / Outside



$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \mid G)$$

„Outside“

$$\beta_j(p,q) = P(w_{pq} \mid N^j_{pq}, G)$$

„Inside“

Inside / Outside

- Outside probability:

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \mid G)$$

- Inside probability

$$\beta_j(p,q) = P(w_{pq} \mid N^j_{pq}, G)$$

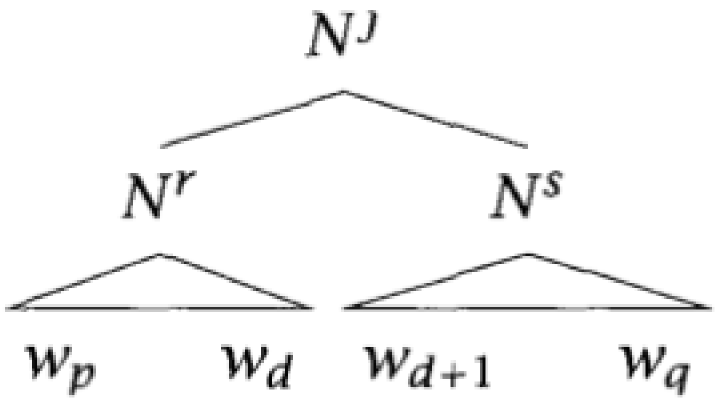
- Likelihood of a chain of terms:

$$P(w_{1T} \mid G) = P(N^1 \Rightarrow^* w_{1T}, G)$$

$$= P(w_{1T} \mid N^1_{1T}, G) = \beta_1(1,T)$$

Calculating the Inside Probability

$$\beta_j(p,q) = P(w_{pq} \mid N^j_{pq}, G)$$



Calculating the Inside Probability

- Base case:

$$\beta_j(k,k) = P(w_k \mid N^j_{kk}, G)$$
$$= P(N_j \rightarrow w_k \mid G)$$

- Inductive step:

$$\beta_j(p,q) = P(w_{pq} | N^j_{pq}, G)$$
$$= \sum_{r,s} \sum_{d=p}^{q-1} P(w_{pd}, N^r_{pd}, w_{(d+1)q}, N^s_{(d+1)q} | N^j_{pq}, G)$$
$$= \sum_{r,s} \sum_{d=p}^{q-1} P(N^r_{pd}, N^s_{(d+1)q} | N^j_{pq}, G) P(w_{pd} | N^j_{pq}, N^r_{pd}, N^s_{(d+1)q}, G)$$
$$\times P(w_{(d+1)q} | N^j_{pq}, N^r_{pd}, N^s_{(d+1)q}, w_{pd}, G)$$
$$= \sum_{r,s} \sum_{d=p}^{q-1} P(N^r_{pd}, N^s_{(d+1)q} | N^j_{pq}, G) P(w_{pd} | N^r_{pd}, G)$$
$$\times P(w_{(d+1)q} | N^s_{(d+1)q}, G)$$

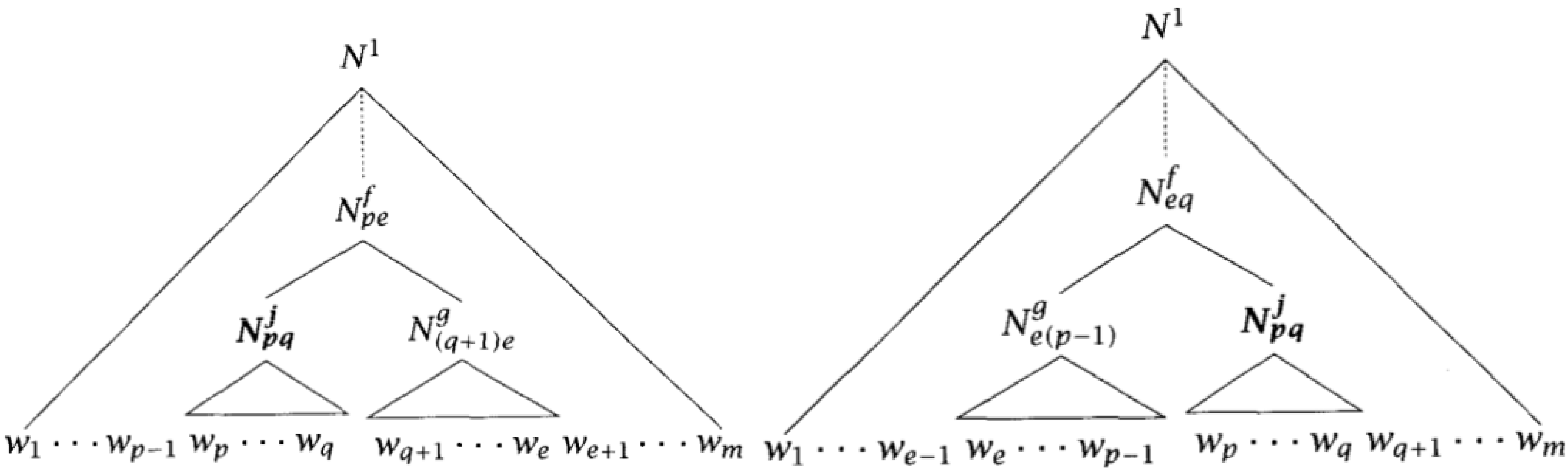
Example: Calculating Inside Probability

$S \rightarrow NP\ VP$	1.0		$NP \rightarrow NP\ PP$	0.4
$PP \rightarrow P\ NP$	1.0		$NP \rightarrow$ *astronomers*	0.1
$VP \rightarrow V\ NP$	0.7		$NP \rightarrow$ *ears*	0.18
$VP \rightarrow VP\ PP$	0.3		$NP \rightarrow$ *saw*	0.04
$P \rightarrow$ *with*	1.0		$NP \rightarrow$ *stars*	0.18
$V \rightarrow$ *saw*	1.0		$NP \rightarrow$ *telescopes*	0.1

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		$\beta_S = 0.0015876$
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	*saw*	*stars*	*with*	*ears*

Calculating the Outside Probability

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \mid G)$$



Calculating the Outside Probability

- Base case: $\alpha_1(1,T)=1; \quad \alpha_j(1,T)=0 \text{ für } j\neq 1$
- Inductive step:

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \mid G)$$

$$= \left(\sum_{f,g} \sum_{e=q+1}^{m} P(w_{1(p-1)}, w_{(q+1)m}, N^f_{pe}, N^j_{pq}, N^g_{(q+1)e}) \right)$$

$$+ \left(\sum_{f,g} \sum_{e=1}^{p-1} P(w_{1(p-1)}, w_{(q+1)m}, N^f_{eq}, N^g_{e(p-1)}, N^j_{pq}) \right)$$

$$= \left(\sum_{f,g} \sum_{e=q+1}^{m} P(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}) P(N^j_{pq} N^g_{(q+1)e} \mid N^f_{pe}) P(w_{(q+1)e} \mid N^g_{(q+1)e}) \right)$$

$$+ \left(\sum_{f,g} \sum_{e=1}^{p-1} P(w_{1(e-1)}, w_{(q+1)m}, N^f_{eq}) P(N^g_{e(p-1)}, N^j_{pq} \mid N^f_{eq}) P(w_{e(p-q)} \mid N^g_{e(p-1)}) \right)$$

$$= \left(\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) P(N^f \rightarrow N^j N^g) \beta_g(q+1,e) \right)$$

$$+ \left(\sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^g N^j) \beta_g(e,p-1) \right)$$

Most Likely Parse Tree

- Viterbi algorithm for PCFG
- $O(m^3 n^3)$
- $\delta_i(p, q)$: maximum inside probability for a parse of subtree N_{pq}^i.
- Initialization: $\delta_i(p, p) = P\left(N^i \to w_p\right)$
- Induction: $\delta_i(p,q) = \max_{j,k,p \leq r < q} P(N^i \to N^j N^k) \delta_j(p,r) \delta_k(r+1,q)$
- Save best parse tree:

  $$\psi_i(p,q) = \arg\max_{(j,k,r)} P(N^i \to N^j N^k) \delta_j(p,r) \delta_k(r+1,q)$$
- Reconstruct parse tree: if $\psi_i(p,q) = (j,k,r)$, then left branch starts with N_{pr}^j, right branch with $N_{(r+1)q}^k$

Scheffer: Language Technology

PCFG: Parameter Estimation

- Grammar has to be known; only probabilities $P(N^i \rightarrow N^j N^k)$ are estimated.

- If corpus is annotated with parse trees: count how frequently each rule is used + Laplace smoothing.
- Without parse tree annotation: use EM algorithm (Baum Welch for PCFGs).

Overview

- Graphical models.
- The n-gram model.
- Hidden Markov model.
- Linear classification models.
- Conditional random fields.
- PCFGs
- Forward- and backpropagation in neural networks.
- Recurrent neural networks.
- LSTM networks.