Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Structured Prediction

Tobias Scheffer

Overview

- Multi-class classification.
- Directed and undirected graphical models.
- Factor graphs.
- Predicting sequences.
- Classification with class taxonomies.
- Predicting trees.
- Predicting graphs.

Recap: Multi-Class Classification

- Motivation: we would like to extend classification to problems with more than 2 classes.
  - $Y = \{1, \dots, k\}$
- Problem: we cannot separate k classes with a single hyperplane.
- Idea: Each class y has a separate function $f_{\boldsymbol{\theta}}(\mathbf{x}, y)$ that is used to predict how likely y is given $\mathbf{x}$.
  - Each function is modeled as linear.
  - We predict class y with the highest scoring function for $\mathbf{x}$.

Multi-Class Classification

- Decision functions:

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

- Classifier:

$$y_{\boldsymbol{\theta}}(\mathbf{x}) = \underset{y \in Y}{\operatorname{argmax}}\, f_{\boldsymbol{\theta}}(\mathbf{x}, y)$$

- Model parameters (k classes):

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}^{1} \\ \vdots \\ \boldsymbol{\theta}^{k} \end{pmatrix}$$

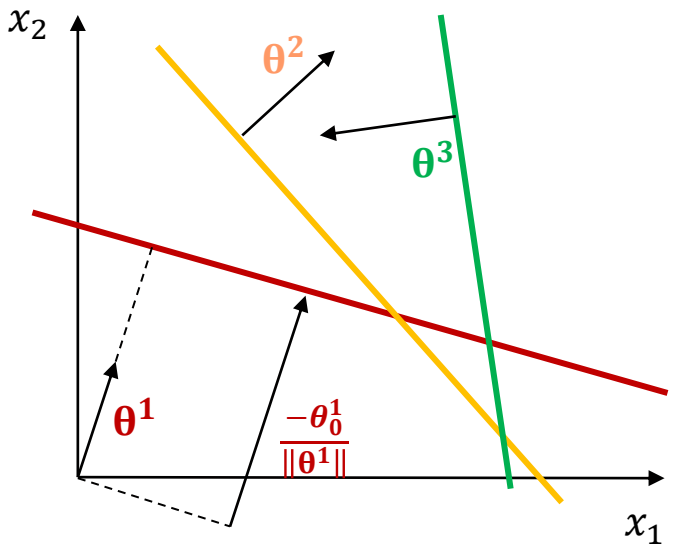
Multi-Class Classification

- Decision functions:

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

- Classifier:

$$y_{\boldsymbol{\theta}}(\mathbf{x}) = \underset{y \in Y}{\operatorname{argmax}}\, f_{\boldsymbol{\theta}}(\mathbf{x}, y)$$



Multi-Class Classification

- Decision function:

$$f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}^{y} \text{ with } \boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}^{1} \\ \vdots \\ \boldsymbol{\theta}^{k} \end{pmatrix}$$

- Decision function in terms of a joint feature mapping of input and output:

$$f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \Phi(\mathbf{x}, y)^{\mathrm{T}}\boldsymbol{\theta} \text{ with } \Phi(\mathbf{x}, y) = \begin{pmatrix} \mathbf{x}[y = 1] \\ \vdots \\ \mathbf{x}[y = k] \end{pmatrix}$$

Multi-Class Classification

- Decision function in terms of a joint feature mapping of input and output:

$$f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \Phi(\mathbf{x}, y)^{\mathrm{T}}\boldsymbol{\theta} \text{ with } \Phi(\mathbf{x}, y) = \begin{pmatrix} \mathbf{x}[y=1] \\ \vdots \\ \mathbf{x}[y=k] \end{pmatrix}$$

- Example: 3-class classification

$$f_{\boldsymbol{\theta}}(\mathbf{x}, 2) = \Phi(\mathbf{x}, 2)^{\mathrm{T}}\boldsymbol{\theta}$$

$$= (\mathbf{x}[2=1] \quad \mathbf{x}[2=2] \quad \mathbf{x}[2=3]) \begin{pmatrix} \boldsymbol{\theta}^1 \\ \boldsymbol{\theta}^2 \\ \boldsymbol{\theta}^3 \end{pmatrix}$$

$$= (0 \quad \mathbf{x} \quad 0) \begin{pmatrix} \boldsymbol{\theta}^1 \\ \boldsymbol{\theta}^2 \\ \boldsymbol{\theta}^3 \end{pmatrix} = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}^2$$

Multi-Class Classification

- Decision function in terms of a joint feature mapping of input and output:
  $f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \Phi(\mathbf{x}, y)^{\mathrm{T}}\boldsymbol{\theta}$ with $\Phi(\mathbf{x}, y) = \Phi(\mathbf{x}) \times \Lambda(y)$,
  $\Phi(\mathbf{x}) = \mathbf{x}$, and $\Lambda(y) = \begin{pmatrix} [y = 1] \\ \vdots \\ [y = k] \end{pmatrix}$
- Example: 3-class classification

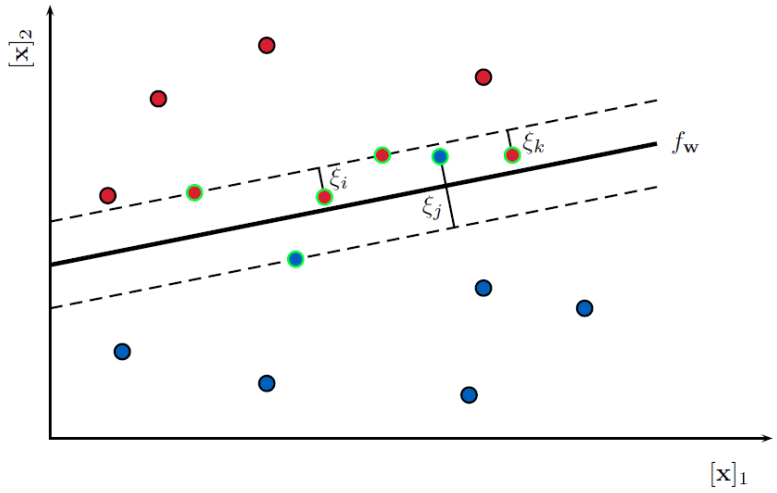
$$\Phi(\mathbf{x}, y) = \Phi(\mathbf{x}) \times \Lambda(y)$$
$$= \mathbf{x} \times \begin{pmatrix} [y = 1] \\ [y = 2] \\ [y = 3] \end{pmatrix} = \begin{pmatrix} \mathbf{x}[y = 1] \\ \mathbf{x}[y = 2] \\ \mathbf{x}[y = 3] \end{pmatrix}$$

Binary SVM: Optimization Problem

- minimize

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

- Equivalent to: minimize $\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta}$ subject to the constraints
  - $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq 1 - \xi_i$
  - $\xi_i \geq 0$

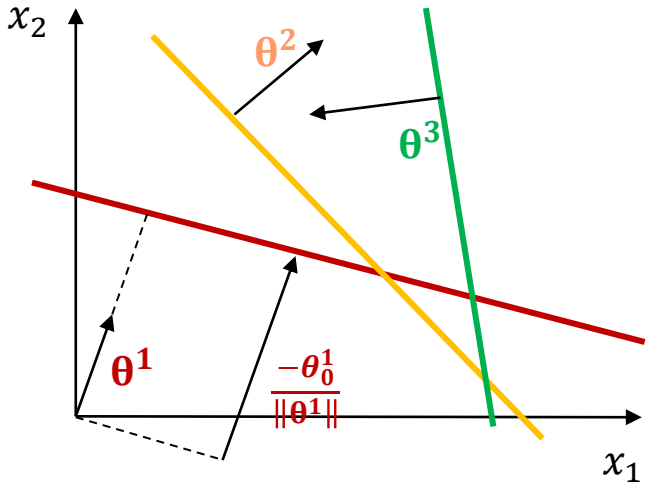


Multi-Class SVM: Optimization Problem

- minimize

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

- Minimize
  $\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta}$ subject to the constraints
  - $\forall y \neq y_i : f_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i) \geq f_{\boldsymbol{\theta}}(\mathbf{x}_i, y) + 1 - \xi_i$
  - $\xi_i \geq 0$



Joint Feature Mappings

- Input $\mathbf{x}$ can be arbitrary structure (vector, sequence, graph, …)
- Output $\mathbf{y}$ can be arbitrary structure (vector, sequence, graph, …)
- Structured prediction:

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

$$p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z} e^{-\Phi(\mathbf{x},\mathbf{y})^{\mathrm{T}} \boldsymbol{\theta}}$$

- Joint feature mapping $\Phi(\mathbf{x}, \mathbf{y})$ encodes arbitrary joint structural properties of $\mathbf{x}$ and $\mathbf{y}$.

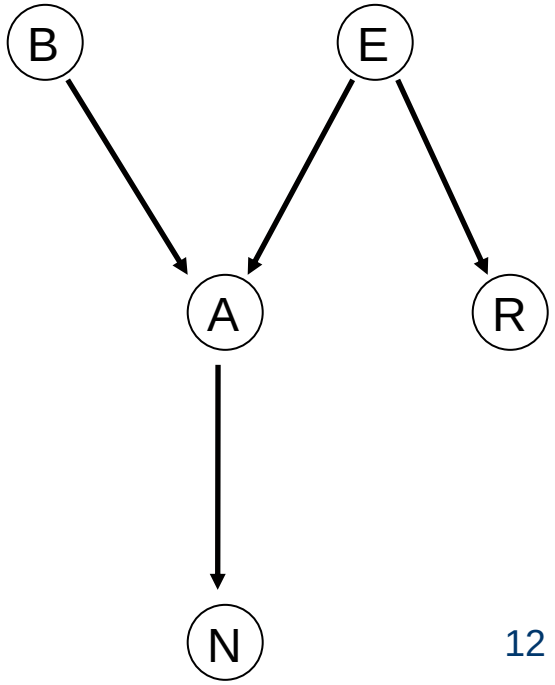
Directed Graphical Models

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

$$P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i | pa(X_i))$$

  - Graph has edges from $pa(X_i)$ to X_i.

- $P(B)P(E)P(A|B,E)P(R|E)P(N|A)$



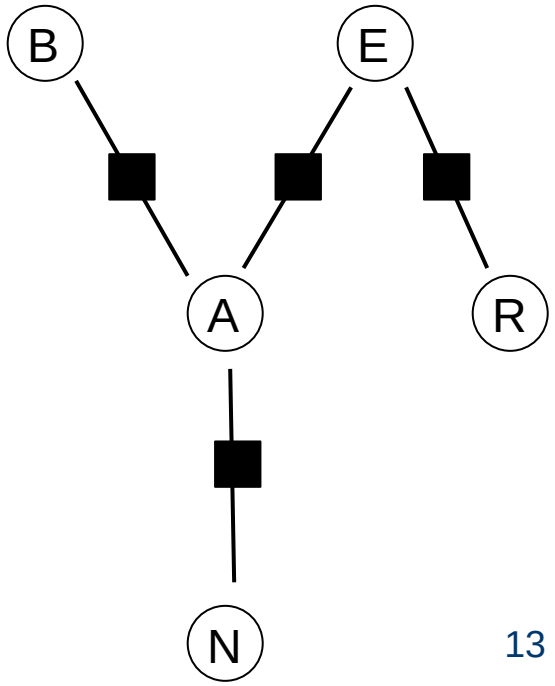
Undirected Graphical Models

- 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.

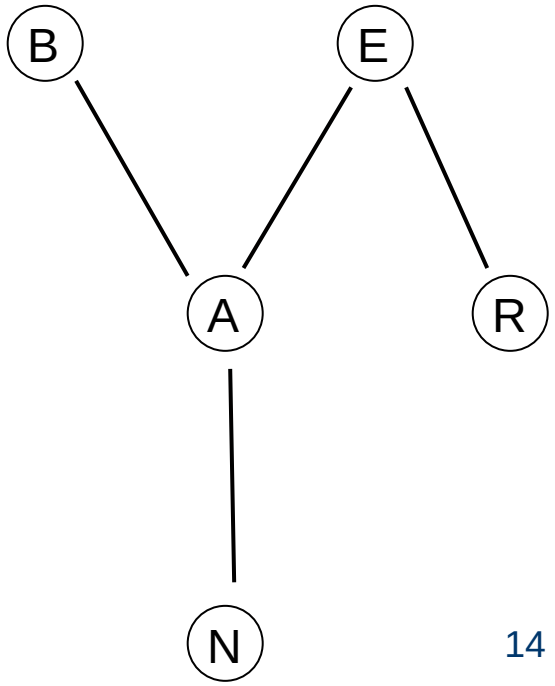


Undirected Graphical Models

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

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

- 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 $X_1, \ldots, X_N$:

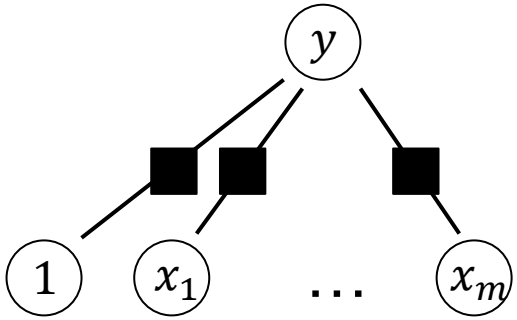
$$P(X_1, \ldots, X_N) = \frac{1}{Z} \prod_{j=1}^{k} \Psi_j$$

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

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

Factor Graph for Classification

- Inputs $\mathbf{x}$, output y.
- Factors $\Psi_j^y = p(y|x_j) = e^{\theta_j^y x_j}$.



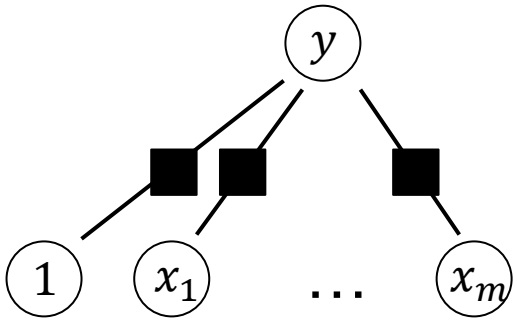
- Log-Factors $\psi_j^y = \log p(y|x_j) = \theta_j^y x_j$.
- $P(y|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{Z}\prod_{j=1}^{k} \Psi_j^y$

$$= \frac{1}{Z}\exp\left\{\sum_{j=1}^{k} \psi_j^y\right\} = \frac{1}{Z}\exp\left\{\sum_{j=1}^{N} \theta_j^y x_j\right\}$$

- Logistic regression

Factor Graph for Classification

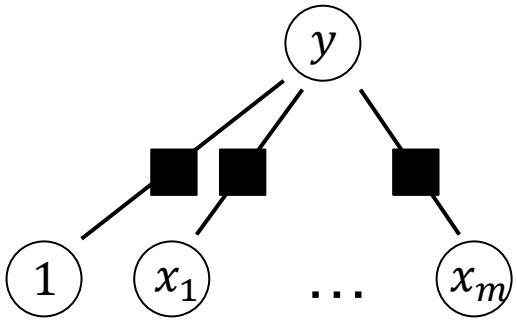
- Inputs $\mathbf{x}$, output y.
- Factors $\Psi_j^y = p(y|x_j) = e^{\theta_j^y x_j}$.
- $\Phi(\mathbf{x}, y) = \Phi(\mathbf{x}) \times \Lambda(y)$



- $\Lambda(y) = \begin{pmatrix} [y = 1] \\ \vdots \\ [y = k] \end{pmatrix}$
- $P(y|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{Z} \exp\{\boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}, y)\}$
$= \frac{1}{Z} \exp\{\boldsymbol{\theta}^{\mathrm{T}} (\Phi(\mathbf{x}) \times \Lambda(y))\}$
- Multi-class logistic regression

Factor Graph for Classification

- Inputs $\mathbf{x}$, output y.
- Factors $\Psi_j^y = p(y|x_j) = e^{\theta_j^y x_j}$.
- $\Phi(\mathbf{x}) = \mathbf{x}$

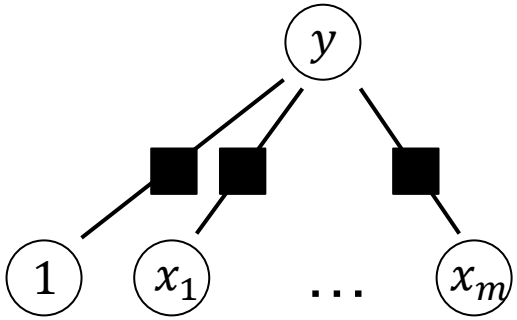


- $\Phi(\mathbf{x}) \times \Lambda(y) = \begin{pmatrix} \mathbf{x}[y=1] \\ \vdots \\ \mathbf{x}[y=k] \end{pmatrix}$
- $P(y|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{Z} \exp\{\boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}, y)\}$

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

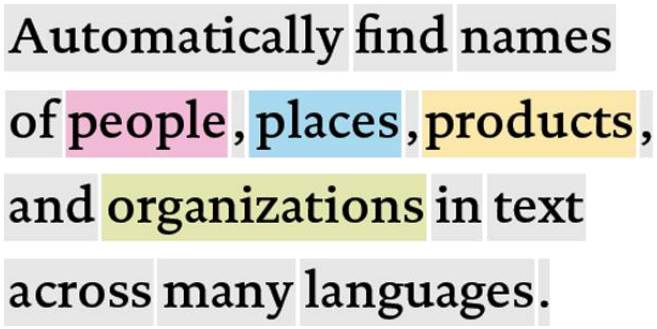
Factor Graph for Classification

- Inputs $\mathbf{x}$, output y.
- $\Phi(\mathbf{x}, y) = \Phi(\mathbf{x}) \times \Lambda(y)$
- $\Lambda(y) = \begin{pmatrix} [y = 1] \\ \vdots \\ [y = k] \end{pmatrix}$
- $f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}, y)$
  $= \boldsymbol{\theta}^{\mathrm{T}} (\Phi(\mathbf{x}) \times \Lambda(y))$



Sequence Models

- Widely used for natural language processing
  - E.g.,Part-of-speech tagging, named-entity recognition, information extraction
- Directed models: hidden Markov models
  - Generative graphical models
  - Perform less well at prediction
- Undirected graphical models:
  - conditional random fields: discriminative graphical model
  - Hidden-Markov-SVM: large-margin method



Sequence Models: Examples

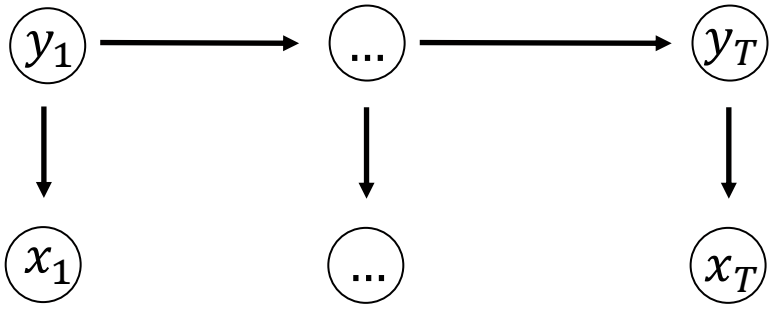
- Part-of-speech tagging
  - **x**= “Curiosity kills the cat.” → **y**= <Noun, Verb, Determiner, Noun>
- Named-entity recognition
  - **x**= “Barbie meets Ken.” → **y**= <Person, -, Person>

Sequences: HMM Recap

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

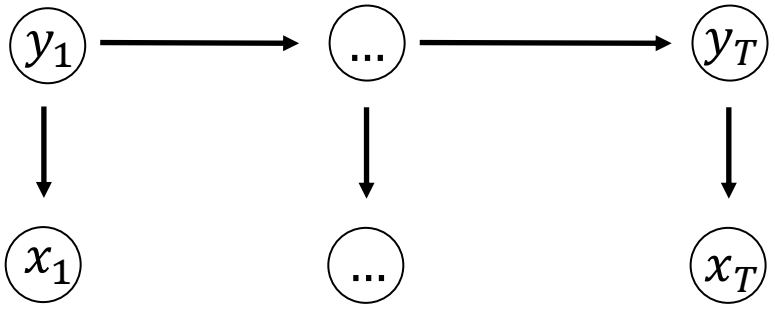
$$P(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = P(x_1, \ldots, x_T, y_1, \ldots, y_T|\boldsymbol{\theta})$$
$$= \prod_{t=1}^{T} P(y_t|y_{t-1}, \boldsymbol{\theta})\, P(x_t|y_t, \boldsymbol{\theta})$$

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



Sequences: HMM Recap

- 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:
  - $P(y_{t+1}|y_1, \dots, y_t, \boldsymbol{\theta}) = P(y_{t+1}|y_t, \boldsymbol{\theta})$
  - $P(x_t|y_1, \dots, y_T, \boldsymbol{\theta}) = P(x_t|y_t, \boldsymbol{\theta})$

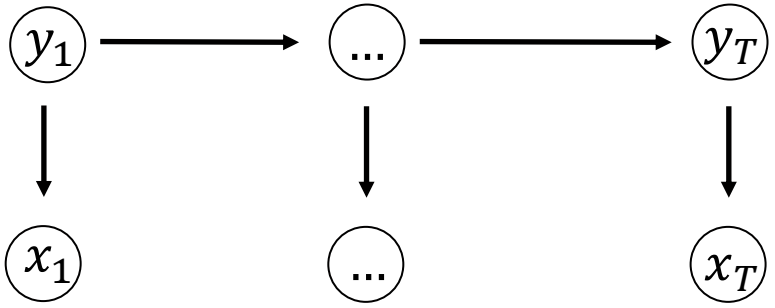


Sequences: HMM Recap

- Prediction: most likely output given input

$$\text{argmax}_{y_1,\ldots,y_T} P(y_1, \ldots, y_T | x_1, \ldots, x_T, \boldsymbol{\theta})$$

- Naive maximization over all combinations of labels; exponentially many in T.
- But can be solved in $O(T)$ with dynamic programming → Viterbi algorithm.



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:

$$\delta_{t+1}(j) = \max_{y_1,\dots,y_t} P(y_1,\dots,y_t,y_{t+1} = j, x_1,\dots,x_{t+1}|\boldsymbol{\theta})$$
$$= \max_{y_1,\dots,y_t} P(y_1,\dots,y_t,x_1,\dots,x_t|\boldsymbol{\theta})a_{ij}b_j(x_{t+1})$$
$$= \max_i \max_{y_1,\dots,y_{t-1}} P(y_1,\dots,y_t = i,x_1,\dots,x_t|\boldsymbol{\theta})a_{ij}b_j(x_{t+1})$$
$$= \max_i \delta_t(i)a_{ij}b_j(x_{t+1})$$

Viterbi Algorithm

- Initialization:
  - $\log \delta_1(i) = \log \pi_y b_y(x_1)$
  - $\psi_1(i) = 0$
- For $t = 1 \ldots T-1$, 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 $t = T-1 \ldots 1$
  - $y_t^* = \psi_{t+1}(q_{t+1}^*)$
- Return $y_1^*, \ldots, y_T^*$.

HMM Learning

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

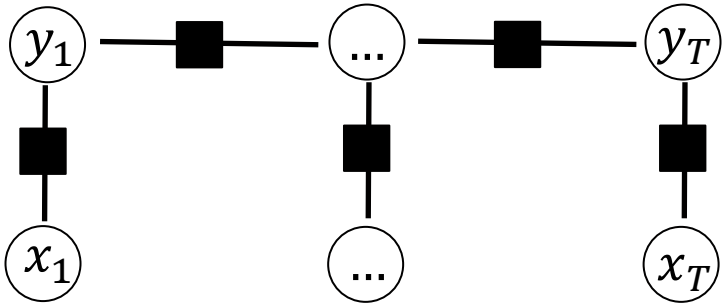
- Laplace correction (regularization): add constant to numerator, change denominator so that probabilities add to one again.

Undirected Sequence Models

- Factorization:

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

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



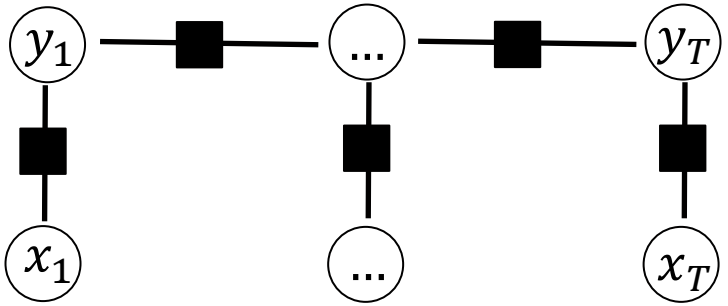
Undirected Sequence Models

- 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:

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

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

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

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

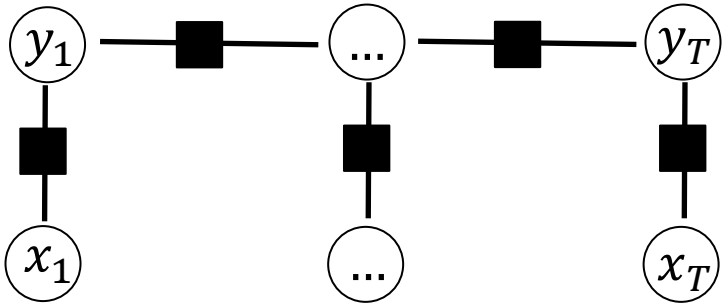
Undirected Sequence Models

- Factorization:

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

- With

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



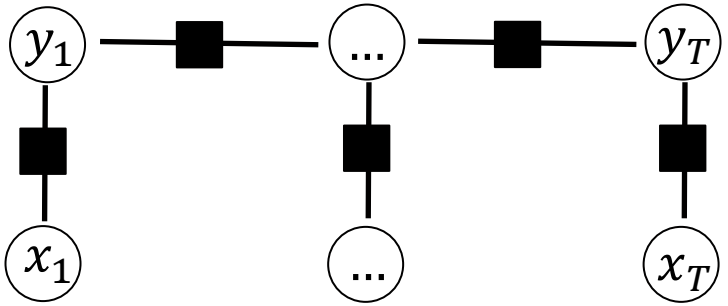
Undirected Sequence Models

- Decision function:

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

- With

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



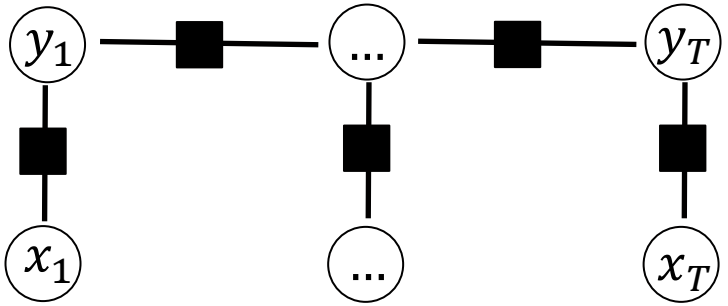
Undirected Sequence Models: Prediction

- Conditional random field: most likely output

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

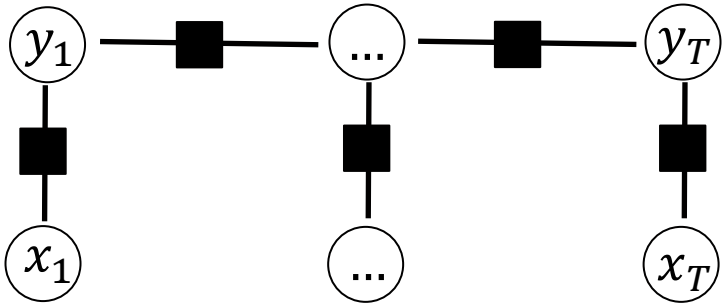
- Decision function:

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



Undirected Sequence Models: Prediction

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

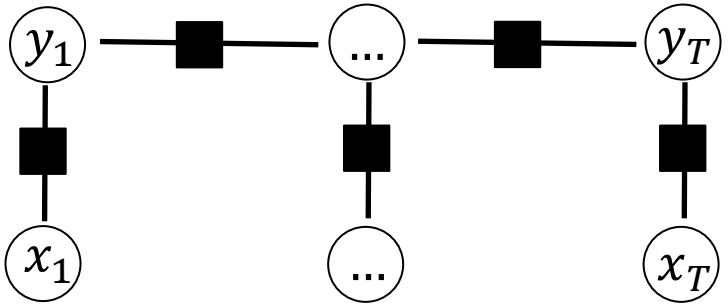


Conditional Random Fields: Inference

- 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.

SVM-Struct: Learning

- Large-margin optimization criterion: minimize

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

- Equivalent to minimize $\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta}$ subject to the constraints
  - $\forall \mathbf{y} \neq \mathbf{y}_i : f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i) \geq f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}) + 1 - \xi_i$
  - $\xi_i \geq 0$

SVM-Struct: Learning

- Large-margin optimization criterion: minimize

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

- Equivalent to minimize $\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta}$ subject to the constraints
  - $\forall \mathbf{y} \neq \mathbf{y}_i : f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i) \geq f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}) + 1 - \xi_i$
  - $\xi_i \geq 0$

- $\forall \mathbf{y} \neq \mathbf{y}_i$: number of constraints is exponential in T.
- Iterative training: explicit training constraint is added when some $\mathbf{y} \neq \mathbf{y}_i$ violates margin during training.

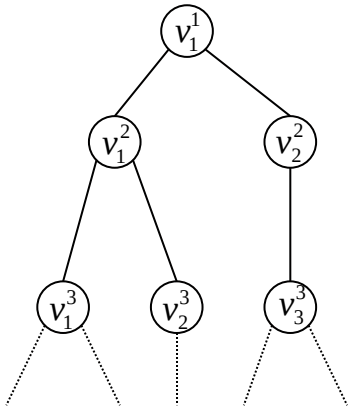
SVM-Struct: Learning Algorithm

- Minimize
  $\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta}$ subject to the constraints
  - $\forall \mathbf{y} \neq y_i : f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i) \geq f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}) + 1 - \xi_i$
  - $\xi_i \geq 0$

- Start with empty working set of constraints
- Iterate over training instances
  - Find $\arg\max_{\bar{\mathbf{y}} \neq \mathbf{y}_i} f_{\boldsymbol{\theta}}(\mathbf{x}_i, \bar{\mathbf{y}})$
  - While $f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i) < f_{\boldsymbol{\theta}}(\mathbf{x}_i, \bar{\mathbf{y}}) + 1 - \xi_i$, add this constraint to the working set of training constraints and solve minimization problem over current working set (e.g., using stochastic gradient descent).

Classification with Class Taxonomies

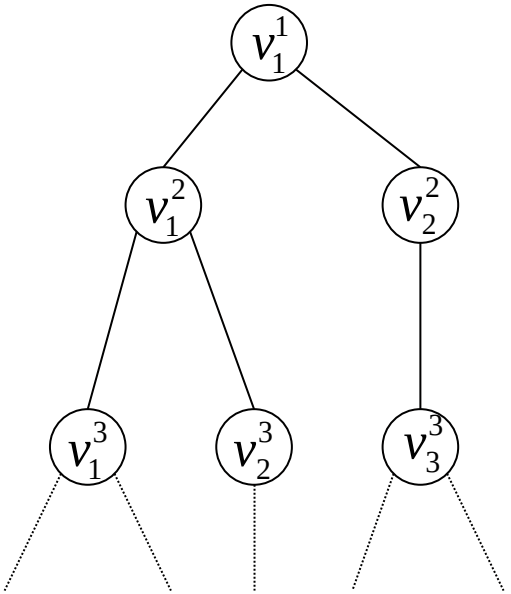
- Taxonomic tree of classes
  - $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} f_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$
  - $f_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \boldsymbol{\theta}^{\mathrm{T}} \Phi(\mathbf{x}, \mathbf{y})$
  - $\mathbf{y} = (y^1, \ldots, y^d)$
  - $\Lambda(\mathbf{y}) = \begin{pmatrix} \Lambda(y^1) \\ \vdots \\ \Lambda(y^d) \end{pmatrix}$ with $\Lambda(y^i) = \begin{pmatrix} y_i = v_1 \\ \vdots \\ y_i = v_k \end{pmatrix}$
  - $\Phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \otimes \Lambda(\mathbf{y}) = \phi(\mathbf{x}) \otimes \begin{pmatrix} \Lambda(y^1) \\ \vdots \\ \Lambda(y^d) \end{pmatrix} = \phi(\mathbf{x}) \otimes \begin{pmatrix} [\![y^1 = v_1^1]\!] \\ \vdots \\ [\![y^1 = v_{n_1}^1]\!] \\ \vdots \\ [\![y^d = v_1^d]\!] \\ \vdots \\ [\![y^d = v_{n_d}^d]\!] \end{pmatrix}$



Classification with Class Taxonomies

- Let $\mathbf{x}$ be a document
- $\mathbf{y} = \left(v_1^1, v_2^2, v_3^3\right)^T$ is a path in a subject taxonomy tree

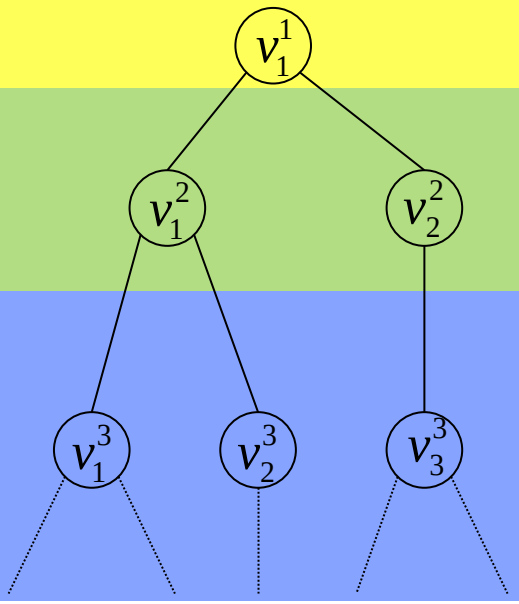
$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} [\![y^1 = v_1^1]\!]\mathbf{x} \\ [\![y^2 = v_1^2]\!]\mathbf{x} \\ [\![y^2 = v_2^2]\!]\mathbf{x} \\ [\![y^3 = v_1^3]\!]\mathbf{x} \\ [\![y^3 = v_2^3]\!]\mathbf{x} \\ [\![y^3 = v_3^3]\!]\mathbf{x} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{0} \\ \mathbf{x} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{x} \\ \vdots \end{pmatrix}$$



Classification with Class Taxonomies

- Let $\mathbf{x}$ be a document
- $\mathbf{y} = \left(v_1^1, v_2^2, v_3^3\right)^T$ is a path in a subject taxonomy tree

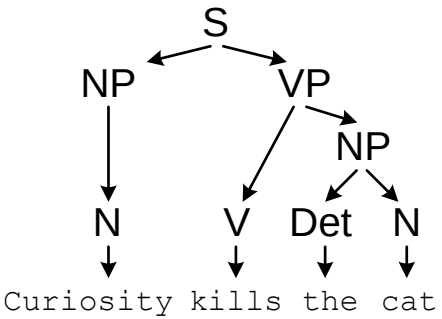
$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} [\![y^1 = v_1^1]\!]\mathbf{x} \\ [\![y^2 = v_1^2]\!]\mathbf{x} \\ [\![y^2 = v_2^2]\!]\mathbf{x} \\ [\![y^3 = v_1^3]\!]\mathbf{x} \\ [\![y^3 = v_2^3]\!]\mathbf{x} \\ [\![y^3 = v_3^3]\!]\mathbf{x} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{0} \\ \mathbf{x} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{x} \\ \vdots \end{pmatrix}$$



Prediction of Trees

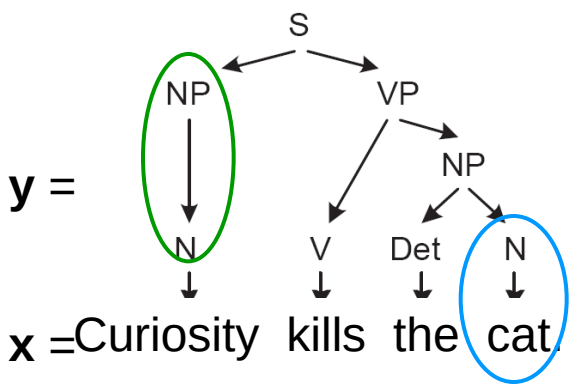
- Natural Language Parsing

x= “Curiosity kills the cat.” ↦



- Here, the entire tree (not just a node in a tree) is the output

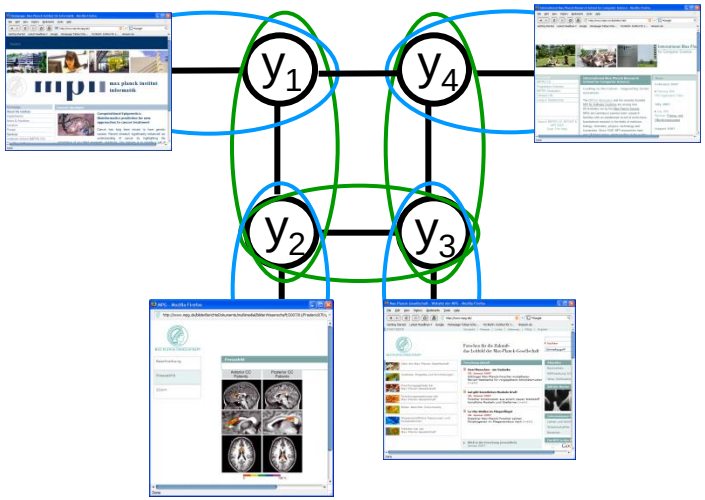
Prediction of Trees



$$\Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 & S \rightarrow NP,VP \\ 1 & NP \rightarrow N \\ 0 & VP \rightarrow V \\ 1 & VP \rightarrow V,NP \\ \vdots & \\ 0 & N \rightarrow ate \\ 1 & N \rightarrow cat \end{pmatrix}$$

- Natural Language Parsing
- Feature vector contains one feature for each production rule.
- Decoding by dynamic programming
  - CKY-Parser, $O(n^3)$

Graph Labeling (Collective Classification)



- Feature mapping contains one attribute for each pair of neighboring labels y_i, y_j
- Feature mapping contains feature for each pair of label y and observation x

- Decoder: Loopy belief propagation, approximate inference

Summary

- Directed and undirected graphical models for
  - Multi-class classification,
  - Sequences, Trees,
  - Classification with class taxonomies,
  - Prediction of trees, graphs.
- Generative models (e.g., HMMs) maximize (regularized) likelihood of input and output.
  - Not ideal for discrimination when goal is to find the right $\mathbf{y}$.
- Discriminative models find parameter that gives correct $\mathbf{y}$ for all training inputs $\mathbf{x}$.
  - Conditional random field, SVM-Struct