Introduction to Graphical Models

lecture 5 - Learning

Ulf Brefeld
TU Berlin

– parameter estimation for graphical models
– maximum likelihood

Overview

- graphical models
  - Bayesian networks
  - Markov random fields

- inference
  - belief propagation
  - loopy belief propagation

- assumption:
  - graph structure is known
  - probability tables are known
  - realistic?

Learning

- Nomenclature
  - – Input variables / observations: x
  - – Output variables / targets: y

- Recall: $P(y|X=x) = P(x|y)P(y)/P(x)$

- Model:
  - – choose a parametric model $P(x|y;\theta)$
  - – adapt parameters θ to data
  - – How can we choose θ to best approximate the true density $p(x)$

Supervised vs. Unsupervised Settings

- Task: estimate parameters $\boldsymbol{\theta}$

- supervised learning problems
  – given n input-output pairs $(x_1, y_1), \ldots, (x_n, y_n)$
  – $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
  – maximum likelihood (ML)

- unsupervised learning problems
  – only n observations are given: $x_1, x_2, \ldots, x_n \in \mathcal{X}$
  – (later in this lecture)

Maximum Likelihood

- For points generated independently and identically distributed (iid) from $p(X = x|Y = y)$, the likelihood of the data is

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} p(x_i|y;\theta)$$

- Often convenient to take logs,

$$L(\theta) = \log \mathcal{L}(\theta) = \sum_{i=1}^{n} \log p(x_i|y;\theta)$$

- Maximum likelihood chooses θ to maximize $\mathcal{L}$ (and thus L)

Example: multinomial distribution

- Consider an experiment with n independent trials
- Each trial can result in any of r possible outcomes (e.g., a die)
- p_i denotes the probability of outcome i, $\sum_{i=1}^{r} p_i = 1$
- n_i denotes the number of trials resulting in outcome i, $\sum_{i=1}^{r} n_i = n$
- The likelihood is given by

$$\mathcal{L}(p_1, \ldots, p_r) = \prod_{i=1}^{r} p_i^{n_i}$$

- Show that the maximum likelihood estimate for p_i is $\hat{p}_i = \frac{n_i}{n}$
  – proof in Davis & Jones, ML Estimation for the Multinomial Distribution, Teaching Statistics 14(3), 1992

Applications

- part-of-speech tagging

  – input: sentence (=observation)

  – output: sequence of part-of-speech tags (= latent variables)

- named entity recognition (NER)

  – input: sentence (=observation)

  – output: sequence of named entites (time, person, location, organization, ...)

- protein secondary structure prediction

  – input: primary structure

  – output: secondary structure

Example: Natural Language Processing

- Part-of-speech tagging:
  – input: ”Curiosity kills the cat.“
  – output: $<$noun, verb, determiner, noun$>$

- named entity recognition (NER)
  – input: ”Robert Enke was born in August 1977 in Jena.“
  – output: $<$ person, person, o, o, o, date, date, o, location$>$

- NER also relevant in biomedical applications: gene/protein detection

Protein Secondary Structure Prediction

- example:

```
KVFGRCELAA AMKRHGLDNY RGYSLGNWVC
 B  HHHHHH HHHHTT TTB TTB HHHHHH

AAKFESNFNT QATNRNTDGS TDYGILQINS
HHHHHHTTBT T EEE TTS  EEETTTTEET

RWWCNDGRTP GSRNLCNIPC SALLSSDITA
TTB B S  T T  BTT SBG GGGGSSS HH

SVNCAKKIVS DGNGMNAWVA WRNRCKGTDV
HHHHHHHHHT SSSGGGGSHH HHHHTTTS G

QAWIRGCRL
GGGTTT
```

Label Sequence Learning

- formalization:
  - input: sequence $\mathbf{x} = x_1, x_2, \ldots, x_T$
  - output: sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$
  - elements in $\mathbf{x}$ and $\mathbf{y}$ are not iid!

- Structure is determined by length of input sequence

- goal:
  - prediction model: $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$
  - given a new sentence $\mathbf{x}'$, compute prediction $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}';\boldsymbol{\theta})$$

  - capture dependencies between neighboring words

Approaches

- flat approaches (naive Bayes, SVM, ...)
  - indendence assumption on words of a sentence
  - cannot exploit dependencies

Approaches

- flat approaches (naive Bayes, SVM, ...)
  - indendence assumption on words of a sentence
  - cannot exploit dependencies

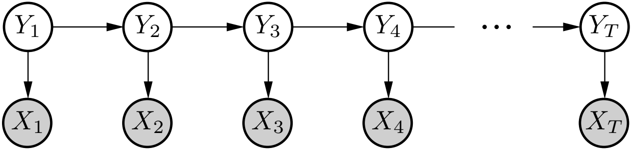
- flat appraoches w/ sliding windows
  - capture dependencies within window
  - long-range dependencies are not detected

Approaches

- flat approaches (naive Bayes, SVM, ...)
  - indendence assumption on words of a sentence
  - cannot exploit dependencies

- flat appraoches w/ sliding windows
  - capture dependencies within window
  - long-range dependencies are not detected

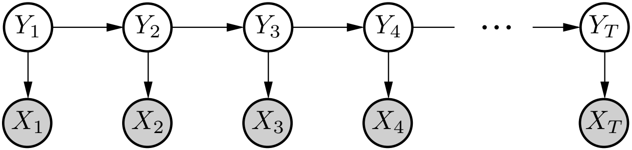
- Preliminary solution: employ first-order hidden Markov model:



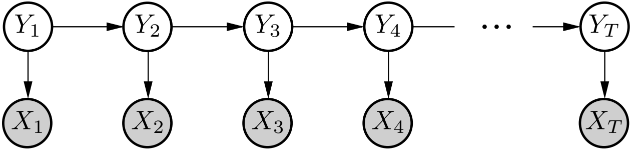
Part-of-Speech Tagging

- Given:
  - given n pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$
  - $\mathbf{x}_i = x_{i1}, \ldots, x_{iT_i}$ is the i-th input sequence
  - $\mathbf{y}_i = y_{i1}, \ldots, y_{iT_i}$ is the i-th annotation
  - $dom(x_{ij}) = \{\text{Aachen}, \text{Aar}, \ldots, \text{ZZ-top}\}$
  - $dom(y_{ij}) = \{\text{noun}, \text{verb}, \text{determiner}, \ldots\}$

- Graphical model:



Recall: HMMs



$$P(Y_1, .., Y_T, X_1, .., X_T) = P(Y_1) \left[\prod_{t=1}^{T} P(Y_t|Y_{t\text{-}1}) \right] \left[\prod_{t=1}^{T} P(X_t|Y_t) \right]$$

- multinomial distributions:
  - priors: $P(Y_1)$
  - emissions: $P(X_t|Y_t)$
  - transitions: $P(Y_t|Y_{t\text{-}1})$

Parameter Estimation

- Maximum likelihood says:

  – Priors: $\pi_i = P(y_1 = \sigma_i) = \frac{1}{n}\sum_{k=1}^{n}[[y_{k1} == \sigma_i]]$

  – emissions:

$$P(x_t = w|y_t = \sigma_i) = \frac{\sum_{k=1}^{n}\sum_{p=1}^{T_k}[[y_{kp} == \sigma_i \wedge x_{kp} == w]]}{\sum_{k=1}^{n}\sum_{p=1}^{T_k}[[y_k == \sigma_i]]}$$

  – transitions:

$$P(y_{t+1} = \sigma_j|y_t = \sigma_i) = \frac{\sum_{k=1}^{n}\sum_{p=1}^{T_k}[[y_{kp} == \sigma_i \wedge y_{k,p+1} == \sigma_j]]}{\sum_{k=1}^{n}\sum_{p=1}^{T_k}[[y_k == \sigma_i]]}$$

Applying the trained HMM

- HMM can be adapted to data with maximum likelihood

- Once the probabilities are estimated, the HMM can be used for prediction

- 2 possibilities:
  – use sum-product algorithm to optimize $P(y_t|x_1, \ldots, x_T)$
  – use max-product algorithm to optimize $P(y_1, \ldots, y_T|x_1, \ldots, x_T)$
  – max-product for first-order hidden Markov models is called Viterbi algorithm

Viterbi Algorithm

- Compute: $\mathrm{argmax}_{y_1,\ldots,y_T} P(y_1,\ldots,y_T|x_1,\ldots,x_T)$

- Define $\delta_{t+1}(\sigma_i) = \max_{y_1,\ldots,y_t} P(y_1,\ldots,y_{t+1}=\sigma_i, x_1,\ldots,x_{t+1})$

  – $\delta_{t+1}(\sigma_i)$ is the best score along a single path up to time $t+1$ which account for the first $t+1$ observations and ends in state σ_i at time $t+1$

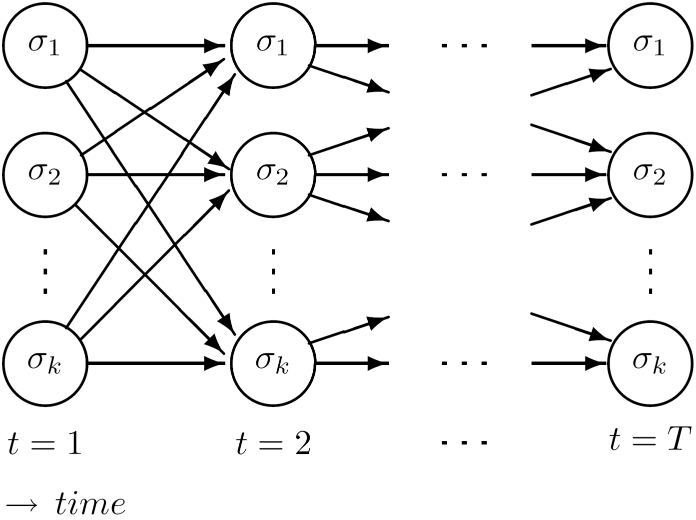
  – apply $\delta_{t+1}(\sigma_i)$ recursively, similar to forward-backward algorithm (except that a max than sum operation is used)

  – see also: Rabiner, Proc. IEEE 77(2), 1989 pp. 257-285

Viterbi Algorithm

- initialize $\delta_1(\sigma_i) = P(y_1 = \sigma_i)P(x_1|y_1 = \sigma_i)$
- initialize $\psi_1(\sigma_i) = 0$
- loop $j = 1, \ldots, |\Sigma|$ and $t = 1, \ldots, T-1$:
  - $\delta_{t+1}(\sigma_j) = \left[\max_i \delta_t(i)P(y_{t+1} = \sigma_j|y_t = \sigma_i)\right]P(x_{t+1}|y_{t+1} = \sigma_j)$
  - $\psi_{t+1}(\sigma_j) = \left[\operatorname{argmax}_t \delta_t(i)P(y_{t+1} = \sigma_j|y_t = \sigma_i)\right]P(x_{t+1}|y_{t+1} = \sigma_j)$
- termination: $y_T^* = \operatorname{argmax}_i \delta_T(\sigma_i)$
- loop $t = T-1, \ldots, 1$
  - $y_t^* = \psi_{t+1}(y_{t+1}^*)$

Trellis



Limitations of HMMs

- Long-range dependencies are not captured

  – a remedy might be higher-order HMMs

  – computationally demanding

- probabilities need to be smoothed

  – unobserved words (and sequences including them) will always have zero probability

  – a common approach that does not work very well is Laplace smoothing:

$$P(x_t = w | y_t = \sigma_i) = \frac{1 + \sum_{k=1}^{n} \sum_{p=1}^{T_k} [[y_{kp} == \sigma_i \wedge x_{kp} == w]]}{|dom(x_t)| + \sum_{k=1}^{n} \sum_{p=1}^{T_k} [[y_k == \sigma_i]]}$$

More Severe Limitations of HMMs

- HMMs are generative models
  - HMMs address the joint probability $P(\mathbf{x}, \mathbf{y})$
  - we are interested in discriminative models $P(\mathbf{y}|\mathbf{x})$
  - HMMs optimize the wrong criterion!

- Next time:
  - Use Markov random field instead of Bayesian network
  - Condition joint probability on the observations
  - Conditional random fields