Why Graphical Models and Bayesian methods? – 1

- formalization of *Information Processing*
  – data is information
  – sensors give information
  – outputs/actions/decisions are *missing* information (to be 'inferred')
  – coupling between sources/points of information

$\Rightarrow$ Graphical Models formalize "networks of coupled information"

$\Rightarrow$ Information Processing can be viewed as inference or message passing in Graphical Models

probability theory

- why do we need probabilities?
  – of course, in case of random events, stochasticity...

- but also in a deterministic world!:
  – lack of knowledge!
  – hidden (latent) variables
  – expressing *uncertainty*
  – expressing *information*

- probabilities are a generic tool to express uncertainty, information, and coupling

Probability: Frequentist and Bayesian

- Frequentist probabilities are defined in the limit of an infinite number of trials
- *Example:* The probability of a particular coin landing heads up is 0.43

- Bayesian (subjective) probabilities quantify degrees of belief
- *Example:* The probability of it raining tomorrow is 0.3
- Not possible to repeat tomorrow many times

random variables

- intuitively: a *random variable* takes on *values* with a certain probability

  a bit more formally: a random variable relates a measureable space with a domain (sample space) and thereby introduces a probability measure on the domain (“assigns a probability to each possible value”)

- the *domain* dom(X) of a variable X is the set possible values of a random varible (mutually exclusive and collectively exhaustive)
  Example: a dice can take values $\{1, .., 6\}$

- we use capital letters X to denote random variables and lower case letters x to denote values that they take

- we use the P to denote the mapping to probabilites

random variables (in terms of sets)

Let X be a random variable with domain $\Omega = \text{dom}(X)$
Let $A, B \subset \Omega$ be subsets of the domain and $x \in \Omega$ a value in the domain.

- $X \in A$ or $X \in B$ or $X = x$ are called *events*
- we use the P to denote the mapping to probabilties:
  – $P(X \in A) \in \mathbb{R}$
- we require
  – $P(X \in \emptyset) = 0$ and $P(X \in \Omega) = 1$
  – if $A \cap B = \emptyset$ then $P(X \in A \cup B) = P(X \in A) + P(X \in B)$
  if the domain is discrete this implies *normalization:*
  – $\sum_{x \in \Omega} P(X = x) = 1$

probabilty distribution & tables

- for continuous domains: “probability distribution” is the integral of a “probability density function”
- for discrete domains: “probability distribution” and “probability mass function” are used synonymously

- a RV assigns a probability to each possible value
  $\rightarrow$ think of the probability distribution as a *table* of numbers:
  Example: A fair dice X, dom$(X) = \{1, 2, 3, 4, 5, 6\}$, with
  $$\forall_{x \in \text{dom}(X)} : \; P(X = x) = \frac{1}{6}$$
  corresponds to the table
  $$[\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}]$$
- in implementations we typically represent random variables by tables (arrays/vectors) of numbers

joint distributions

- assume we have two random variable X and Y. The *joint probability distribution*

  $$P(X = x, Y = y)$$

  gives the probability that $X = x$ *and* $Y = y$.
  (In logic one would perhaps write something like $X = x \wedge Y = y$. But not so in joint probability distributions.)

- *Example:* Suppose Toothache and Cavity are the variables:

	Toothache = true	Toothache = false
Cavity = true	0.04	0.06
Cavity = false	0.01	0.89

  we write

  $$P(Toothache = true, Cavity = false) = 0.01$$

joint distributions

- note, most of what'll need will be about JOINT PROBABILITY DISTRIBUTIONS
  – graphical models are nothing but descriptions of joint probability distributions!
  – correlations, interdependence, coupling are all expressed in terms of joint probability distributions
  – whenever you're confused about the "model", the "approach", the "assumptions", etc, reconsider explicitly what the joint probability distribution over all involved variables is!

joint distributions

- *definitions:*
  – the *marginal* (probability) of X given $P(X,Y)$ is

$$P(X) = \sum_Y P(X,Y)$$

  – the *conditional* (probability) of X given Y and $P(X,Y)$ is

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}$$

  defs also hold for tuples of variables, e.g., $X = (X_1,..,X_n)$, $Y = (Y_1,..,Y_m)$

- *implications:*
  – the *product rule* $P(X,Y) = P(X|Y)\,P(Y) = P(Y|X)\,P(X)$
  – the *chain rule* $P(X_1,..,X_n) = \prod_{i=1}^n P(X_i|X_1,..,X_{i\text{-}1})$
  – *Bayes Rule*

$$P(X|Y) = \frac{P(Y|X)}{P(Y)} P(X)$$

Ich gehe nach Hause....

P(Ich) P(gehe | Ich) P(nach | Ich ~~gehe~~)

Bayes Rule

P(Hause | ~~[illegible]~~ ~~[illegible]~~ [illegible])

- Thomas Bayes (1702–1761)
- Bayes Rule is a trivial implication of the definitions of marginal and conditional probability!

$$P(Y|X) = \frac{P(X,Y)}{P(X)}$$

- importance lies in its interpretation and use:

$$P(X|Y) = \frac{P(Y|X)}{P(Y)}\, P(X)\,, \quad \text{posterior} = \frac{\text{likelihood}}{\text{evidence}}\, \text{prior}$$

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})}{P(\text{effect})}\, P(\text{cause})$$

Example: let M be meningitis, S be stiff neck

$$P(M|S) = \frac{P(S|M)}{P(S)}\, P(M) = \frac{0.8}{0.1}\, 0.0001 = 0.0008$$

Note: posterior probability of meningitis still very small

N-Grams: (shingles)

$\uparrow$

Ich gehe nach Hause

1-Gram / unigram: $\underset{1}{\text{"ich"}}$, $\underset{2}{\text{"gehe"}}$, $\underset{3}{\text{"nach"}}$ ...

2-Gram: $\underset{1}{\text{"ich gehe"}}$, $\underset{2}{\text{"gehe nach"}}$, $\underset{3}{\text{"nach Hause"}}$

3-Gram: "Ich gehe nach", "gehe nach Hause"

inference

- we will deal with many variables
  $X = (H_1, ..H_n,\ E_1, .., E_m,\ Y_1, .., Y_k)$
  – we are given the joint probability distribution
  $P(H_1, ..H_n,\ E_1, .., E_m,\ Y_1, .., Y_k)$
  – some variables $E_1, .., E_m$ are observed (we have evidence)
  for the other variables $H_1, ..H_n\ , Y_1, .., Y_k$ we have no evidence
  we want to know the *posterior* over some variables $Y_1, .., Y_K$

$$P(Y_{1:k} \,|\, E_{1:m}) = \frac{P(Y_{1:k}, E_{1:m})}{P(E_{1:m})} \propto \sum_{H_{1:n}} P(Y_{1:k}, E_{1:m}, H_{1:n}) \quad (1)$$

- computing $P(Y_{1:k} \,|\, E_{1:m})$ is the *problem of inference*

- obvious problem: size of table $P(Y_{1:k}, E_{1:m}, H_{1:n})$ is d^{k+m+n}

summary

- focus of this lecture:

  – graphical models as a generic tool for inference with coupled random variables

  – probability theory as calculus for uncertainty, information, evidence

  – learning graphical models from data

  – using graphical models for decision making & RL

- next time:

  – naive Bayes

  – graphical models

  – inference using the elimination algorithm

cheat sheat

- a random variable X assignes probabilties $P(X\!=\!x) \in \mathbb{R}$ to values $x \in \text{dom}(x)$
- probabilty distribution $\leftrightarrow$ table (vector) of probabilties for each value (normalization: $\sum_X P(X) = 1$)
- joint distribution $P(X,Y)$ $\leftrightarrow$ table (matrix) of probabilties
- definition: marginal $P(X) = \sum_Y P(X,Y)$ (summing along columns/rows)
- definition: conditional $P(X|Y) = \frac{P(X,Y)}{P(Y)}$ (normalizing each column)
- implications:

$$P(X,Y) = P(X|Y)\;P(Y) = P(Y|X)\;P(X)$$

$$P(X_1,..,X_n) = \prod_{i=1}^n P(X_i|X_1,..,X_{i\text{-}1})$$

$$P(X|Y) = \frac{P(Y|X)}{P(Y)} P(X)\;, \quad \text{posterior} = \frac{\text{likelihood}}{\text{evidence}}\;\text{prior}$$

- definition: *inference* is the problem to compute

$$P(Y_{1:k} \,|\, E_{1:m}) = \frac{P(Y_{1:k}, E_{1:m})}{P(E_{1:m})} \propto \sum_{H_{1:n}} P(Y_{1:k}, E_{1:m}, H_{1:n})$$

- web links:

  Bayes Rule:

  `http://www.cs.ubc.ca/~murphyk/Bayes/bayesrule.html`

  Kevin's lecture:

  `http://www.cs.ubc.ca/~murphyk/Teaching/CS532c_Fall04/Lectures/index.html`

  `http://www.cs.ubc.ca/~murphyk/Bayes/bnsoft.html`

  `site:http://www.cs.ubc.ca/~murphyk/Bayes`

Overview

- graphical models
  - Bayesian networks
  - Markov random fields
- inference
  - ~~belief propagation~~
  - ~~loopy belief propagation~~
- assumption:
  - graph structure is known
  - probability tables are known
  - realistic?

Klaus	chase	Karl-Heinz
N	V	N
P	∅	P

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

- 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

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

$$D = \{(x_n, y_n)\}_{n=1..N}$$

$$p(D|\theta)$$

- 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

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

- 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

$|Y| = 20$

- Part-of-speech tagging:
  - input: *Curiosity kills the cat.* 1 2 3 4

    4^{20}
  - 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>

    B-P I-P B-D, I-D B-L
- NER also relevant in biomedical applications: gene/protein detection

  ~~FA~~ (

Protein Secondary Structure Prediction

$f(x) = w^T\phi(x) + b$

- example:

$\phi(x)$

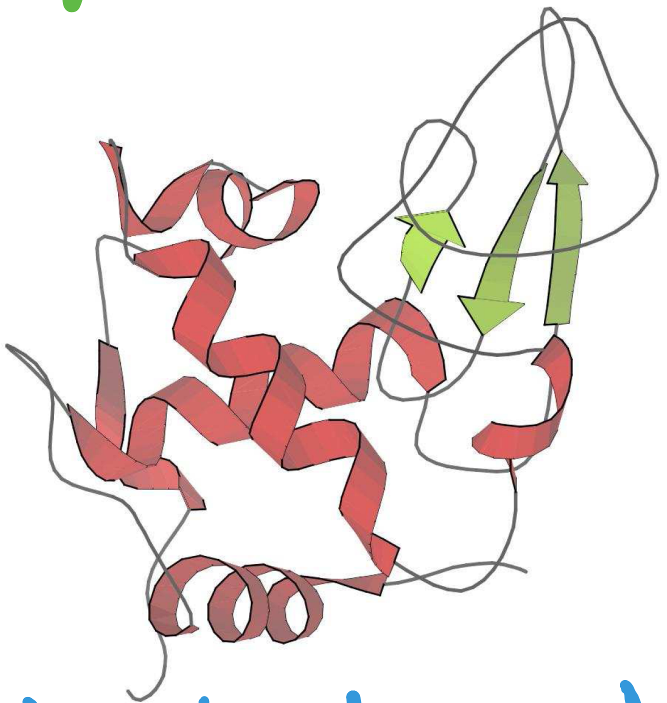
```
KVFGRCELAA AMKRHGLDNY RGYSLGNWVC
 B  HHHHHH HHHHTT TTB TTB HHHHHH

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
```



NLP: $P(y_1|K)\,P(y_2|V,K)\,P(y_3|F,V,K)\ldots$

Naive Bayes

$$P(y|x) = \frac{P(x|y)\,P(y)}{P(x)}$$

$$P(x|y)$$

$$\Rightarrow P(x_1|y)\,P(x_2|y)\ldots$$

$$\prod_{d=1}^{D} P(x_d|y)$$

Label Sequence Learning

- formalization:
  - input: sequence $\xi = x_1, x_2, \ldots, x_T$
  - output: sequence $\uparrow = y_1, y_2, \ldots, y_T$
  - elements in ξ and are not iid!

- Structure is determined by length of input sequence

- goal:
  - prediction model: ~~$P(|\xi;)$~~ $P(y_1, y_2, \ldots, y_T \mid x_1, \ldots, x_T)$
  - given a new sentence ξ', compute prediction $\hat{}$:

    $\arg\max_{y_1 \ldots y_T} P(y_1 \ldots y_T \mid x_1 \ldots x_T)$ $\hat{} =$ ~~$P(|\xi';)$~~
  - capture dependencies between neighboring words

Standard

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

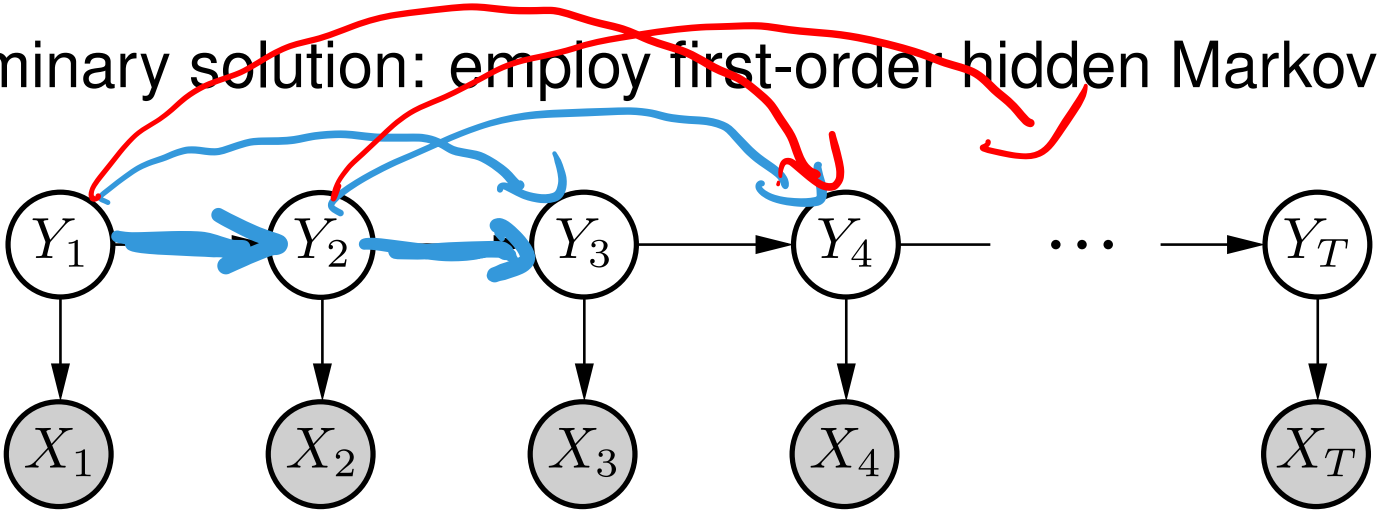
$P(Y_1)P(Y_2|Y_1)P(Y_3|Y_2)$

$\cdots P(X_1|Y_1)$

$P(X_2|Y_2)$

$\vdots$

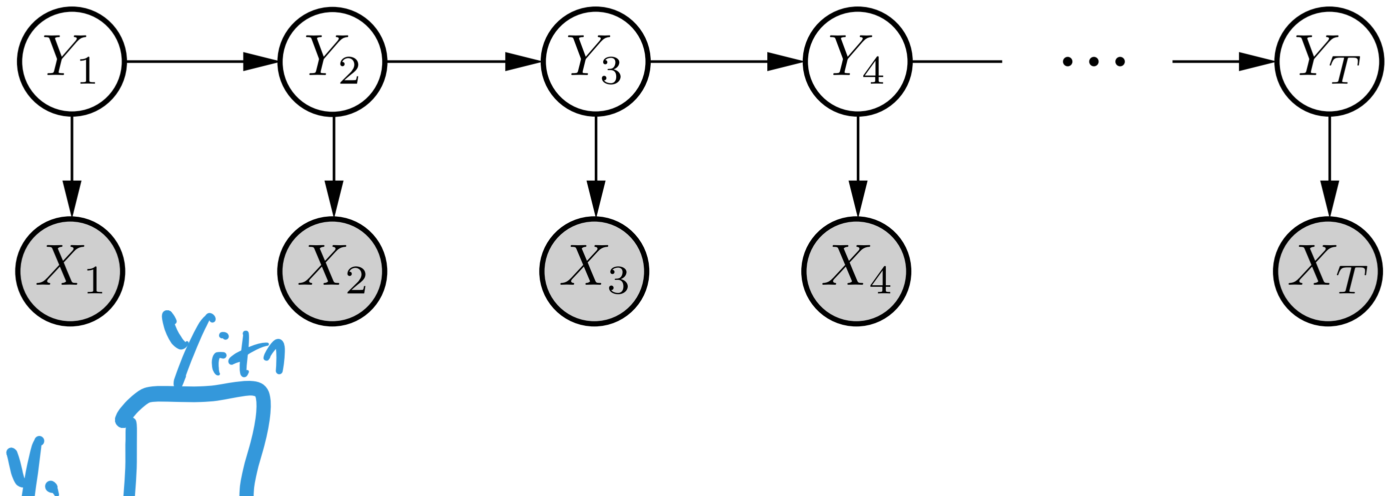
- flat appraoches w/ sliding windows
  - capture dependencies within window
  - long-range dependencies are not detected
- Preliminary solution: employ first-order hidden Markov model:



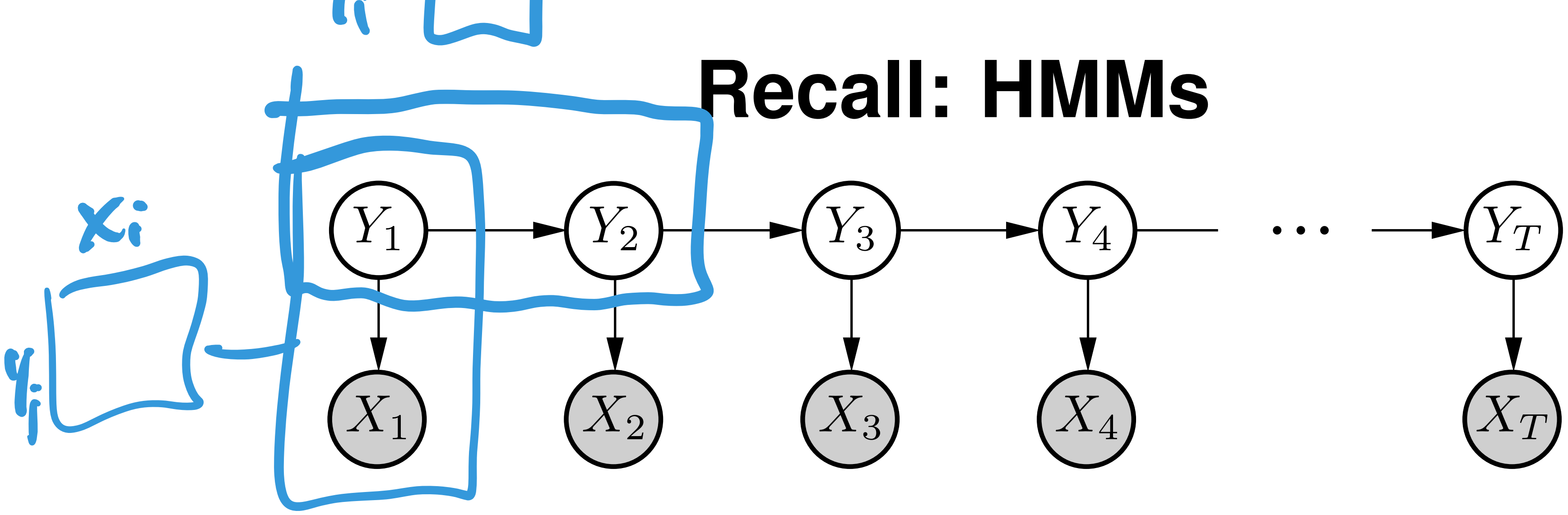
Part-of-Speech Tagging

- Given:
  - given n pairs $(\xi_1, {}_1), \ldots, (\xi_n, {}_n)$
  - $\xi_i = x_{i1}, \ldots, x_{iT_i}$ is the i-th input sequence
  - ${}_i = y_{i1}, \ldots, y_{iT_i}$ is the i-th annotation
  - $dom(x_{ij}) = \{$Aachen, Aar, $\ldots$, ZZ-top$\}$
  - $dom(y_{ij}) = \{$noun, verb, 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: ${}_{y_1,\ldots,y_T} P(y_1, \ldots, y_T | x_1, \ldots, x_T)$

- 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 $t+1$

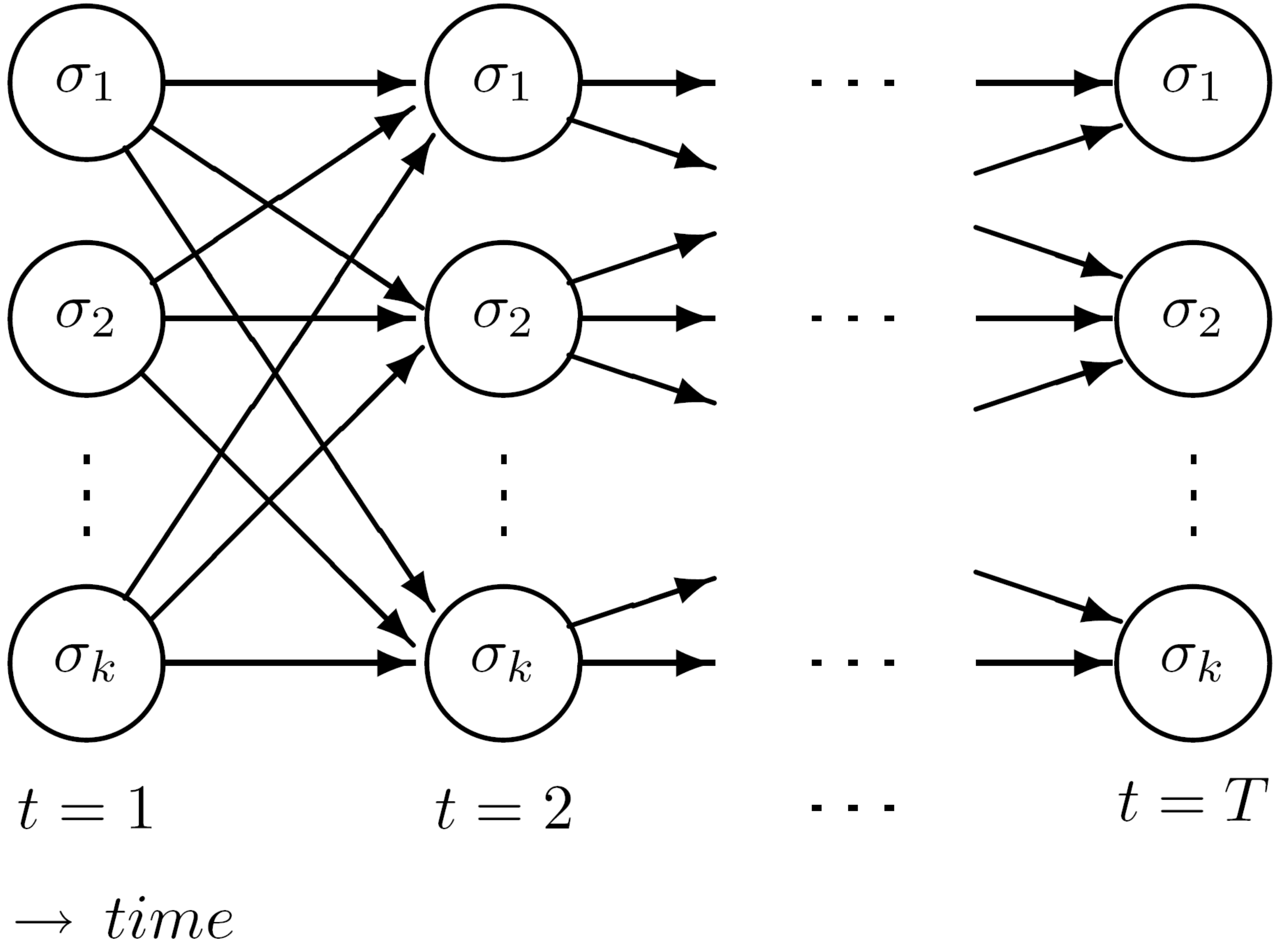
  – apply $\delta_{t+1}(\sigma_i)$ recursively, similar to forward-backward algorithm (except that a max than sum operation is used)

  – 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 $j = 1, \dots, |\Sigma|$ and $t = 1, \dots, T-1$:
  - $\delta_{t+1}(\sigma_j) = \big[\max_i \delta_t(i)P(y_{t+1} = \sigma_j|y_t = \sigma_i)\big]P(x_{t+1}|y_{t+1} = \sigma_j)$
  - $\psi_{t+1}(\sigma_j) = \big[_t\delta_t(i)P(y_{t+1} = \sigma_j|y_t = \sigma_i)\big]P(x_{t+1}|y_{t+1} = \sigma_j)$
- termination: $y^*_T =_i \delta_T(\sigma_i)$
- loop $t = T-1, \dots, 1$
  - $y^*_t = \psi_{t+1}(y^*_{t+1})$

Trellis


σ_1
σ_2
$\vdots$
σ_k
σ_1
σ_2
$\vdots$
σ_k
$\cdots$
σ_1
σ_2
$\vdots$
σ_k
$t = 1$
$t = 2$
$\cdots$
$t = T$
$\rightarrow time$

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:

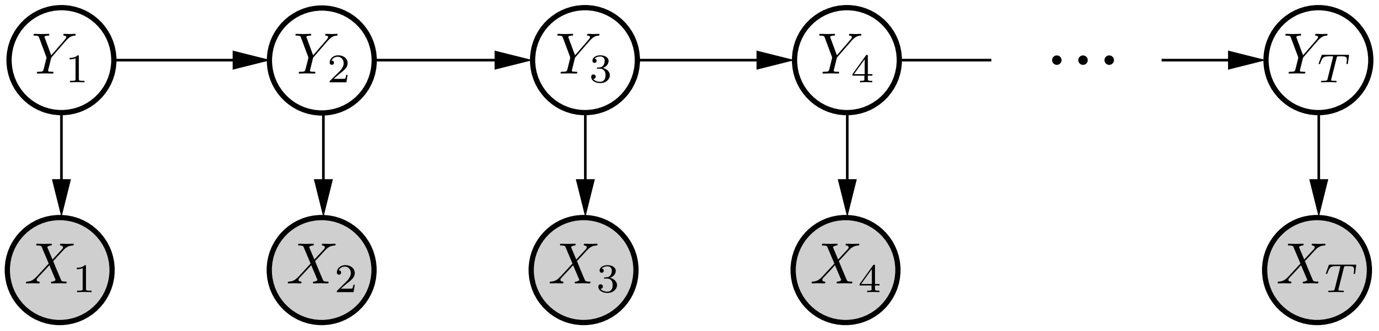
$$P(x_t = w | y_t = \sigma_i) = \frac{1 + \sum_{k=1}^{n} \sum_{p=1}^{T_k} [[y_{kp} == \sigma_i \wedge x_{kp} == w]]}{|dom(x_t)| + \sum_{k=1}^{n} \sum_{p=1}^{T_k} [[y_k == \sigma_i]]}$$

More Severe Limitations of HMMs

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

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

Recall: HMMs


$$\begin{tikzcd}
Y_1 \arrow[r] \arrow[d] & Y_2 \arrow[r] \arrow[d] & Y_3 \arrow[r] \arrow[d] & Y_4 \arrow[r, dash] \arrow[d] & \cdots \arrow[r] & Y_T \arrow[d] \\
X_1 & X_2 & X_3 & X_4 & & X_T
\end{tikzcd}$$

- Hidden Markov models
  - generative models for sequential data
  - parameters: prior, transition, and observation probabilities
  - joint probability:

$$P(X_1, \ldots, Y_1 \ldots) = P(Y_1) \prod_{i=1}^{T} P(X_i|Y_i) \prod_{i=2}^{T} P(Y_i|Y_{i-1})$$

Learning HMMs

- given: n labeled sequences $(\xi_{1,1}), \ldots, (\xi_{n,n})$
- maximum Likelihood (ML)
  - adapt parameters of HMM to data
  - HMM: ML reduces to counting
  - efficient (one pass over data suffices)
  - easy to implement
  - exact inference (Viterbi algorithm)
- drawbacks
  - $P($unobserved token$|Y_i) = 0$ (remedy: smoothing techniques)
  - generative models optmize the wrong criterion

Today: From HMMs to CRFs

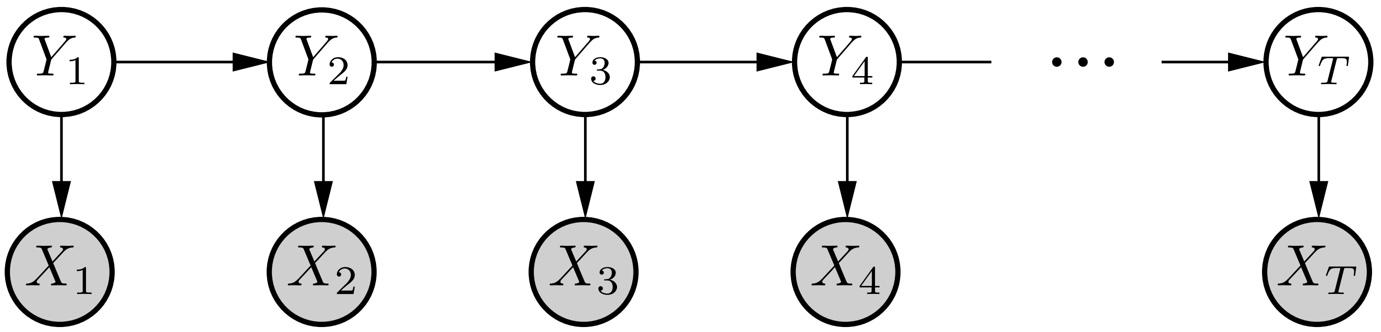
- Use undirected graphical model
  – no assumption on directions of dependencies (i.e., WWW, NLP, images, ...)
  – sequences: factor graph does not change
  – Markov random fields

- Condition joint probability of MRF on observations
  – criterion: prediction model
  – now: conditional (=discriminative) model

Conditional Random Fields

Markov Random Fields

HMM:



MRF:

- every BN can be translated into equivalent MRF (moralization)

MRF: Joint Probability Distribution

- joint probability factorizes across cliques
  - cliques between transitions and label-observation pairs

$$P(X_1, \ldots, Y_1, \ldots) = \frac{1}{Z} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i)$$

- potential functions $\psi^{trans}(Y_i, Y_{i-1})$, $\psi^{trans}(Y_i, Y_{i-1})$
- Z normalization term (partition function)

Partition Function

$$P(X_1,\ldots,Y_1,\ldots) = \frac{1}{Z}\prod_{i=1}^{T}\psi^{obs}(X_i,Y_i)\prod_{i=2}^{T}\psi^{trans}(Y_{i-1},Y_i)$$

- the partition function needs to sum over all possible assignments of input and output sequences
  – we have:

$$Z = \sum_{x_1,\ldots,x_T}\sum_{y_1,\ldots,y_T}\prod_{i=1}^{T}\psi^{obs}(X_i,Y_i)\prod_{i=2}^{T}\psi^{trans}(Y_{i-1},Y_i)$$

  – important for $P(X_1,\ldots,Y_1,\ldots)$ being a probability

Potential Functions

$$P(X_1, \ldots, Y_1, \ldots) = \frac{1}{Z} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i)$$

- potential functions ψ^{trans} (transitions), ψ^{obs} (label-observ.)
  - arbitrary, non-negative, positive functions
  - capture relevant dependencies
  - defined across cliques
- problem:
  - size of largest clique depends on input (i.e., WWW)
  - remedy: represent only cliques of size 2 (=Markov network)

Representation

$$P(X_1, \ldots, Y_1, \ldots) = \frac{1}{Z} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i)$$

- sequences
  - all cliques are of size 2
  - only their number varies with T

- how to choose ψ^{trans}, ψ^{obs}?
  - (remember they have to capture relevant dependencies)

- common assumption (Hammersley & Clifford theorem):
  - ψ is log-linear combination of basis functions ϕ_j

Members in the Exponential Family

- basis functions:

$$\psi^{trans}(Y_i, Y_{i-1}) = \exp\{\sum_{j=1}^{d_{trans}} w_j^{trans} \phi_j^{trans}(Y_{i-1}, Y_i)\}$$

$$\psi^{obs}(X_i, Y_i) = \exp\{\sum_{j=1}^{d_{obs}} w_j^{obs} \phi_j^{obs}(X_i, Y_i)\}$$

  – math turns out to be nice!

  – write:

$$P(X_1, \ldots, Y_1, \ldots) = \frac{1}{Z} \prod_{i=1}^{T} \exp\{\sum_{j=1}^{d_{obs}} w_j \phi_j^{obs}(X_i, Y_i)\}$$
$$\prod_{i=2}^{T} \exp\{\sum_{j=1}^{d_{trans}} w_j \phi_j^{trans}(Y_{i-1}, Y_i)\}$$

Basis Functions: label-label

$$\psi^{trans}(Y_i, Y_{i-1}) = \exp\left\{\sum_{j=1}^{d_{trans}} w_j^{trans}\phi_j^{trans}(Y_{i-1}, Y_i)\right\}$$

- simple case: indicator functions

$$\begin{aligned}
\phi_1^{trans}(Y_{i-1}, Y_i) &= [[Y_{i-1} = \text{noun} \wedge Y_i = \text{noun}]] \\
\phi_2^{trans}(Y_{i-1}, Y_i) &= [[Y_{i-1} = \text{noun} \wedge Y_i = \text{verb}]] \\
\vdots \qquad & \qquad \vdots \\
\phi_{d_{trans}}^{trans}(Y_{i-1}, Y_i) &= [[Y_{i-1} = \text{adverb} \wedge Y_i = \text{adverb}]]
\end{aligned}$$

  – similar to HMM

  – later more...

Basis Functions: label-observation

$$\psi^{obs}(X_i, Y_i) = \exp\left\{\sum_{j=1}^{d_{obs}} w_j^{obs} \phi_j^{obs}(X_i, Y_i)\right\}$$

- simple case: indicator functions

$$\begin{aligned}
\phi_1^{obs}(X_i, Y_i) &= [[X_i = \text{Aachen} \wedge Y_i = \text{noun}]] \\
\phi_2^{obs}(X_i, Y_i) &= [[X_i = \text{Aar} \wedge Y_i = \text{noun}]] \\
\vdots \quad & \quad\quad \vdots \\
\phi_{d_{obs}}^{obs}(X_i, Y_i) &= [[X_i = \text{ZZ-top} \wedge Y_i = \text{adverb}]]
\end{aligned}$$

  – similar to HMM

  – later more...

Putting Everything Together...

$$
\begin{aligned}
P(\xi,) &= \frac{1}{Z} \prod_{i=1}^{T} \exp\{\sum_{j=1}^{d_o} w_j^o \phi_j^o(x_i, y_i)\} \prod_{i=2}^{T} \exp\{\sum_{j=1}^{d_t} w_j^t \phi_j^t(y_{i-1}, y_i)\} \\
&= \frac{1}{Z} \prod_{i=1}^{T} \exp\{\langle \wedge^o, \phi^o(x_i, y_i)\rangle\} \prod_{i=2}^{T} \exp\{\langle \wedge^t, \phi^t(y_{i-1}, y_i)\rangle\} \\
&= \frac{1}{Z} \exp\{\sum_{i=1}^{T} \langle \wedge^o, \phi^o(x_i, y_i)\rangle\} \exp\{\sum_{i=2}^{T} \langle \wedge^t, \phi^t(y_{i-1}, y_i)\rangle\} \\
&= \frac{1}{Z} \exp\{\langle \wedge^o, \sum_{i=1}^{T} \phi^o(x_i, y_i)\rangle\} \exp\{\langle \wedge^t, \sum_{i=2}^{T} \phi^t(y_{i-1}, y_i)\rangle\} \\
&= \frac{1}{Z} \exp\{\langle \underbrace{\begin{pmatrix} \wedge^o \\ \wedge^t \end{pmatrix}}_{=:\wedge}, \underbrace{\begin{pmatrix} \sum_{i=1}^{T} \phi^o(x_i, y_i) \\ \sum_{i=2}^{T} \phi^t(y_{i-1}, y_i) \end{pmatrix}}_{=:\Phi(\xi,)} \rangle\} \\
&= \frac{1}{Z} \exp\{\langle \wedge, \Phi(\xi,)\rangle\}
\end{aligned}
$$

Joint Feature Representation

- joint representation of input and output variables:
  $\Phi(\xi,) = (\sum_{i=1}^{T} \phi^{o}(x_i, y_i), \sum_{i=2}^{T} \phi^{t}(y_{i-1}, y_i))'$

- Example for HMM-alike basis functions:
  – $\Phi(\xi,)$ counts how many times ...
  – ... a *noun* is followed by *verb* (summing over transitions)
  – ... the token *Aachen* is observed as a noun (sum over obs-label)
  – dimensionality of Φ is $dom(x_i) \times dom(y_i) + dom(y_i)^2$

- POS-tagging:
  – dictionary size 20,000 tokens, 36 POS-tags, $dim(\Phi) = 721296$

Example

Features

- Features are engineered to capture important relations/dependencies

- all time favorites for natural language text:
  – n-grams (English: *-ing*, German: *-ung, -heit, -keit*)
  – surface clues (capitalization, all-caps, ...)
  – foreign symbols (α, ω, ...)
  – numbers (42, 1984, ...)

- CRFs allow for rich feature spaces
  – CRFs may contain any number of basis functions
  – basis functions can be defined on the entire input sequence
  – basis functions do need not have a probabilistic interpretation.

More Features / Relation to HMM

- observation-label/transitions can depend on input
  - $\phi^{trans}(y_{t-1}, y_t) \to \phi^{trans}(y_{t-1}, y_t; x_t)$
  - or even: $\phi^{trans}(y_{t-1}, y_t) \to \phi^{trans}(y_{t-1}, y_t; \xi)$
  - similarly: $\phi^{obs}(x_t, y_t) \to \phi^{obs}(\xi, y_t)$
  - (alternative graph structure)

- Implications for HMMs
  - Multi-bernoulli/nomial distribution
  - Generally infeasible

The Exponential Family

$$P(\xi,) = \frac{1}{Z} \exp\{\langle \wedge, \Phi(\xi,)\rangle\}$$

- $P(\xi,)$ is a member in the exp. family, rewrite in canonical form

$$P(\xi,) = \exp\{\langle \wedge, \Phi(\xi,)\rangle - \log Z\}$$

- Identify the terms:
  - $\Phi(\xi,)$ is the sufficient statistics
  - $\wedge$ is the natural parameter
  - $\log Z < \infty$ is the moment generating function

Conditional Markov Random Fields

- joint probability

$$P(\xi,) = \frac{1}{Z} \exp\{\langle \wedge, \Phi(\xi,)\rangle\}$$

  – partition function: $Z = \sum_{\xi} \sum_{\exp\{\langle \wedge, \Phi(\xi,)\rangle\}}$

- condition on the observation

  – apply the rule: $P(|\xi) = P(\xi,)/P(\xi)$

  – obtain new partition function:

$$Z(\xi) = \sum_{\exp\{\langle \wedge, \Phi(\xi,)\rangle\}}$$

- obtain a so-called conditional random field (CRF)

$$P(|\xi) = \frac{1}{Z(\xi)} \exp\{\langle \wedge, \Phi(\xi,)\rangle\}$$

Training CRFs with Maximum Likelihood

- given n input output examples $(\xi_{1},_{1}\ ), \ldots, (\xi_{n},_{n}\ )$

- the log-likelihood is given

$$\log \mathcal{L} = \sum_{i=1}^{n} \langle \wedge, \Phi(\xi_{i},_{i}\ ) \rangle - \log Z(\wedge | \xi_i)$$

- differentiating wrt $\wedge$ gives

$$\frac{\partial}{\partial \wedge} \log \mathcal{L} = \mathbf{E}_{\hat{p}(X,Y)}[\Phi(X,Y)] - \sum_{i=1}^{n} \mathbf{E}_{p(Y|\xi_i;\wedge)}[\Phi(Y,\xi_i)]$$

  – empirical distribution of data $\hat{p}$

  – model distribution p

Optimization

- direct optimization is expensive and often infeasible
  – E.g., calculating the partition function is time consuming if at all possible

- Many differerent optimization strategies have been proposed:
  – linear programming (Roth & Li, 2005)
  – iterative scaling (Lafferty et al., 2001)
  – conjugate gradients (Sha & Pereira, 2003)
  – Gauss-Newton subspace optimization (Altun et al., 2004)
  – gradient tree boosting (Dietterich et al., 2004)
  – stochastic meta descent (Vishwanathan et al., 2006)
  – perceptron algorithm (Altun et al., 2003)
  – ...

The Perceptron Algorithm for CRFs

CRF vs. HMM

- characteristics:
  - CRF: undirected graph, conditional models
  - HMM: directed BN,. generative model

- CRFs generalize HMMs
  - CRFs allow for rich feature spaces
  - HMMs restricted to implicit bag-of-words representation

- Optimization
  - CRF: difficult, complex optimization problem
  - HMM: simple, easy to implement

- Similarities:
  - inference algorithms (Viterbi, sum-product)

Posterior vs. MAP

- Once optimal parameters $\wedge^*$ are found these are used as plug-in estimates $P(|\xi;\wedge^*)$
  - posterior distribution allows for computing confidence intervals

- However, the full posterior is not always needed
  - often, the maximum a posteriori (MAP) estimate suffices
  - e.g., prediction model $\hat{} = P(|\xi)$
  - computing MAP estimates is much cheaper than full posterior!

Computing MAP Estimates

- For MAP estimates compute

$$
\begin{aligned}
\hat{} &= P(|\xi) \\
&= \frac{1}{Z(\xi)} \exp\{\langle \wedge, \Phi(\xi,)\rangle\} \\
&= \langle \wedge, \Phi(\xi,)\rangle
\end{aligned}
$$

– because $\exp$ is a monotone function and $\frac{1}{Z(\xi)}$ is constant

- We arrive at:

$$P(\xi,) \propto \underbrace{\langle \wedge, \Phi(\xi,)\rangle}_{=: f(\xi,)}$$

Outlook

- adapt $f(\xi,) = \langle \wedge, \Phi(\xi,) \rangle$ to data
- perceptron algorithm
  – primal: efficient, nof parameters = $dim(\Phi)$
  – dual: nof parameters = nof possible output sequences
- dual perceptron
  – explicit representation is infeasible
  – solve implicitly by column generation
- examples

Recall: Sequential CRFs

$$P(|\xi) = \frac{1}{Z(\xi)} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i)$$

- potential functions:

$$\psi^{trans}(Y_i, Y_{i-1}) = \exp\left\{ \sum_{j=1}^{d_{trans}} w_j^{trans} \phi_j^{trans}(Y_{i-1}, Y_i) \right\}$$

$$\psi^{obs}(X_i, Y_i) = \exp\left\{ \sum_{j=1}^{d_{obs}} w_j^{obs} \phi_j^{obs}(X_i, Y_i) \right\}$$

- $Z(\xi) = \sum_{y_1,\ldots,y_T} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \; \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i)$

Exemplary Basis Functions

- label-label indicator functions:

$$\phi_1^{trans}(Y_{i-1}, Y_i) = [[Y_{i-1} = \text{noun} \wedge Y_i = \text{noun}]]$$
$$\phi_2^{trans}(Y_{i-1}, Y_i) = [[Y_{i-1} = \text{noun} \wedge Y_i = \text{verb}]]$$
$$\vdots \qquad\qquad \vdots$$
$$\phi_{d_{trans}}^{trans}(Y_{i-1}, Y_i) = [[Y_{i-1} = \text{adverb} \wedge Y_i = \text{adverb}]]$$

- label-observation indicators:

$$\phi_1^{obs}(X_i, Y_i) = [[X_i = \text{Aachen} \wedge Y_i = \text{noun}]]$$
$$\phi_2^{obs}(X_i, Y_i) = [[X_i = \text{Aar} \wedge Y_i = \text{noun}]]$$
$$\vdots \qquad\qquad \vdots$$
$$\phi_{d_{obs}}^{obs}(X_i, Y_i) = [[X_i = \text{ZZ-top} \wedge Y_i = \text{adverb}]]$$

Joint Feature Representation

- Joint representation of input and output variables:

$$\Phi(\xi,) = \left(\sum_{i=1}^{T} \phi^{o}(x_i, y_i)', \sum_{i=2}^{T} \phi^{t}(y_{i-1}, y_i)' \right)'$$

- Rewrite conditional probability:

$$P(|\xi) = \frac{1}{Z(\xi)} \exp \left\{ \langle \wedge, \Phi(\xi,) \rangle \right\}$$

- Observation:

$$P(|\xi) \propto \langle \wedge, \Phi(\xi,) \rangle$$

- MAP estimate:

$$\hat{} = P(\dagger|\xi) = \langle \wedge, \Phi(\xi, \bar{}) \rangle$$

Example

- $\xi =$ Bob jagt den Hund
- We want

$$[N, V, A, N] =\!- \langle \wedge, \Phi(\xi, \neg) \rangle$$

Example

- $\xi =$ Bob jagt den Hund
- We want

$$[N, V, A, N] = \langle \wedge, \Phi(\xi, \cdot) \rangle$$

- Equivalent representation:

$$\begin{aligned}
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, A, A]) \rangle \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, A, N]) \rangle \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, N, A]) \rangle \\
\vdots \quad &> \quad \vdots \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [V, V, V, V]) \rangle
\end{aligned}$$

Example Contd.

- Another equivalent representation:

$$
\begin{aligned}
&\langle \wedge, \Phi(\xi, [N, V, A, N])\rangle - \langle \wedge, \Phi(\xi, [A, A, A, A])\rangle > 0\\
&\langle \wedge, \Phi(\xi, [N, V, A, N])\rangle - \langle \wedge, \Phi(\xi, [A, A, A, N])\rangle > 0\\
&\langle \wedge, \Phi(\xi, [N, V, A, N])\rangle - \langle \wedge, \Phi(\xi, [A, A, N, A])\rangle > 0\\
&\qquad\qquad\qquad\vdots \qquad\qquad\qquad\qquad \vdots\\
&\langle \wedge, \Phi(\xi, [N, V, A, N])\rangle - \langle \wedge, \Phi(\xi, [V, V, V, V])\rangle > 0
\end{aligned}
$$

- The other way round:
  - Update weight vector $\wedge$ in case of an error:

$$\langle \wedge, \Phi(\xi_{i}, {}_{i}\ ))\rangle - \max_{\_}\langle \wedge, \Phi(\xi_{i}, \bar{})\rangle < 0$$

Primal Perceptron

- Simplify things:
  - Error: $_i \neq \hat{} = \langle \wedge, \Phi(\xi_i, \bar{}) \rangle$

- Recall gradient of CRF:

$$\frac{\partial \log \mathcal{L}}{\partial \wedge} = \underbrace{\mathbf{E}_{\hat{p}(X,Y)}[\Phi(X,Y)]}_{\text{truth/emp. distr.}} - \underbrace{\sum_{i=1}^{n} \mathbf{E}_{p(Y|\xi_i;\wedge)}[\Phi(Y,\xi_i)]}_{\text{prediction of model/model distr.}}$$

- Perceptron: perform gradient steps if i-th example is incorrect:

$$\wedge \leftarrow \wedge + \underbrace{\Phi(\xi_i,{}_i)}_{\text{true pair}} - \underbrace{\Phi(\xi_i,\hat{})}_{\text{erroneous prediction}}$$

Primal Perceptron Algorithm

```
1 loop $r = 1, \ldots, r_{max}$
2     loop $i = 1, \ldots, n$
3         Compute $\hat{} = \langle \wedge, \Phi(\xi_i, \bar{}) \rangle$
4         If $_i \neq \hat{}$
5             Update $\wedge \leftarrow \wedge + \Phi(\xi_{i,i}) - \Phi(\xi_i, \bar{})$
6         End (if)
7     End loop ($i$)
8 End loop ($r$)
```

Convergence

Theorem (Extension of Novikoff)

Given n labeled examples $(\xi_{1,1}), \ldots, (\xi_{n,n})$, with $_i \in \mathcal{Y}(\xi_i)$. Let r be the radius of the smallest hypersphere enclosing all difference vectors $\Phi(\xi_{i,i}) - \Phi(\xi_i, \bar{})$, for all i and $\bar{} \neq_i$,

$$r = \max_{1 \leq i \leq n} \max_{\substack{\bar{} \in \mathcal{Y}(\xi_i) \\ \bar{} \neq_i}} |\, \Phi(\xi_{i,i}) - \Phi(\xi_i, \bar{}) \,| .$$

If there exists a vector $\wedge^*$ such that

$$\forall_{i=1}^{n} \forall_{\in \mathcal{Y}(\xi_i)} \langle \wedge^*, \Phi(\xi_{i,i}) \rangle - \langle \wedge^*, \Phi(\xi_i, \bar{}) \rangle \geq \bar{\gamma} \tag{2}$$

holds for some $\bar{\gamma} > 0$ then the number of update steps of the generalized perceptron algorithm is upper bounded by

$$\left(\frac{r}{\bar{\gamma}} \right)^2 |\, \wedge \,|^2. \tag{3}$$

Proof of Theorem

Proof. The weight vector is initialized with $\wedge^{(0)} = \mathbf{0}$. Let $t > 0$ indicate the t-th error of the generalized perceptron, that is for some $1 \leq i \leq n$

$$_i \neq \hat{}_i = _{\bar{} \in \mathcal{Y}(\xi_i)} \langle \wedge^{(t-1)}, \Phi(\xi_i, \bar{}) \rangle.$$

The corresponding update step is given by

$$\wedge^{(t)} = \wedge^{(t-1)} + \Phi(\xi_{i,i}) - \Phi(\xi_i, \hat{}_i) \tag{4}$$

Multiplying Equation 4 with the optimal weight vector $\wedge^*$ yields

$$\begin{aligned} \langle \wedge^*, \wedge^{(t)} \rangle =& \langle \wedge^*, \wedge^{(t-1)} \rangle + \langle \wedge^*, \Phi(\xi_{i,i}) \rangle - \langle \wedge^*, \Phi(\xi_i, \hat{}_i) \rangle \\ \geq& \langle \wedge^*, \wedge^{(t-1)} \rangle + \bar{\gamma} \end{aligned}$$

Applying the principle of induction gives us $\langle \wedge^*, \wedge^{(t)} \rangle \geq t\bar{\gamma}$.

Proof of Theorem (Contd.)

Now we bound $\mid \wedge^{(t)} \mid^2$ from above by

$$\begin{aligned}
\mid \wedge^{(t)} \mid^2 =& \langle \wedge^{(t-1)} + \Phi(\xi_{i,i}) - \Phi(\xi_{i,\hat{i}}) \, , \, \wedge^{(t-1)} + \Phi(\xi_{i,i}) - \Phi(\xi_{i,\hat{i}}) \rangle \\
=& \mid \wedge^{(t-1)} \mid^2 + 2\langle \wedge^{(t-1)}, \Phi(\xi_{i,i}) - \Phi(\xi_{i,\hat{i}}) \rangle + \mid \Phi(\xi_{i,i}) - \Phi(\xi_{i,\hat{i}}) \mid^2 \\
\leq& \mid \wedge^{(t-1)} \mid^2 + \mid \Phi(\xi_{i,i}) - \Phi(\xi_{i,\hat{i}}) \mid^2 \\
\leq& \mid \wedge^{(t-1)} \mid^2 + r^2.
\end{aligned}$$

Thus, by induction we have $\mid \wedge^{(t)} \mid^2 \leq tr^2$. Putting everything together gives us

$$\begin{aligned}
t\bar{\gamma} \leq& \langle \wedge^*, \wedge^{(t)} \rangle \\
\leq& \mid \wedge^* \mid \, \mid \wedge^{(t)} \mid \\
\leq& \mid \wedge^* \mid \sqrt{t} r.
\end{aligned}$$

Solving for t implies the upper bound

$$t \leq \left(\frac{r}{\bar{\gamma}} \right)^2 \mid \wedge^{(*)} \mid^2.$$

Towards Dual Perceptrons

- Observation: $\wedge^{(0)} \leftarrow \mathbf{0}$
- i-th example violates constraint:
  – Update: $\wedge^{(t+1)} = \wedge^{(i)} + \langle \wedge, \Phi(\xi_{i,i}\ )\rangle - \max\langle \wedge, \Phi(\xi_i, \bar{})\rangle$
- Idea: rembember how many times the pair $(\xi_i, \bar{})$ is used for an upate!
  – Variable $\alpha_i(\bar{})$ acts as a counter
  – Initialize: $\alpha_i(\bar{}) \leftarrow 0$
  – Update: $\alpha_i(\bar{}) \leftarrow \alpha_i(\bar{}) + 1$
- The α are bound to violated constraints!

Dual Representation

- Dual parameters
  - $\alpha_i(\bar{})$ is proportional to the importance of $\Phi(\xi_{i,i}\,))\rangle - \Phi(\xi_i,\bar{})$
- Recall: α counted the number of updates for $\wedge$
  - we can thus write:

$$\wedge = \sum_{i=1}^{n} \sum_{^{-}\neq_i} \alpha_i(\bar{})\left(\Phi(\xi_{i,i}\,))\rangle - \Phi(\xi_i,\bar{})\right)$$

- Sparse representation
  - Generally, there are exponentially many $^{-}\neq$
  - However, only a few of them will have an $\alpha_i(\bar{}) > 0$
  - Feature vector can efficiently be encoded and stored (compare dimensionality of primal and dual!)

Dual Decision Function

$$\wedge = \sum_{i=1}^{n} \sum_{\bar{}\neq_i} \alpha_i(\bar{}) \left(\Phi(\xi_i,{}_i\,) \rangle - \Phi(\xi_i,\bar{}) \right)$$

- Plug dual representation of $\wedge$ into decision function:

$$\begin{aligned}
f(\xi',{}'\,) &= \langle \wedge, \Phi(\xi',{}'\,) \rangle \\
&= \langle \sum_{i=1}^{n} \sum_{\bar{}\neq_i} \alpha_i(\bar{}) \left(\Phi(\xi_i,{}_i\,) \rangle - \Phi(\xi_i,\bar{}) \right), \Phi(\xi',{}'\,) \rangle \\
&= \sum_{i=1}^{n} \sum_{\bar{}\neq_i} \alpha_i(\bar{}) \Big(\langle \Phi(\xi_i,{}_i\,) - \Phi(\xi_i,\bar{}), \Phi(\xi',{}'\,) \rangle \Big) \\
&= \sum_{i=1}^{n} \sum_{\bar{}\neq_i} \alpha_i(\bar{}) \Big(\langle \Phi(\xi_i,{}_i\,), \Phi(\xi',{}'\,) \rangle - \langle \Phi(\xi_i,\bar{}), \Phi(\xi',{}'\,) \rangle \Big)
\end{aligned}$$

Kernels and the Dual Perceptron

- Define $K(\xi, , \xi', ') = \langle \Phi(\xi,), \Phi(\xi', ') \rangle$
  - K is called kernel
  - computes inner product in space spanned by Φ
  - rewrite $f(\xi,)$ in terms of kernel functions:

$$f_D(\xi', ') = \sum_{i=1}^{n} \sum_{\bar{} \neq_i} \alpha_i(\bar{}) \left(K(\xi_{i}, _{i}, \xi', ') - K(\xi_i, \bar{}, \xi', ') \right)$$

- Example (sequences, indicator functions)

$$\begin{aligned} K(\xi, , \bar{\xi}, \bar{}) &= \langle \Phi(\xi,), \Phi(\bar{\xi}, \bar{}) \rangle \\ &= \sum_{s,t} [[y^{s-1} = \bar{y}^{t-1} \wedge y^s = \bar{y}^t]] \\ &+ \sum_{s,t} [[y^s = \bar{y}^t]] K_x(x^s, \bar{x}^t) \end{aligned}$$

Kernels on Tokens

- Kernel K_x computes similarity of two tokens
  - Simplest case: $K_x(x, x') = [[x == x']]$
  - No generalization!

- A better choice:
  - K_x computes similarity of feature vectors of observations
  - e.g., n-grams, surface clues
  - Let $\psi(x)$ be the feature vector of token x, then

$$K_x(x, x') = \langle \psi(x), \psi(x') \rangle$$

- K_x can be precomputed for the training process

Dual Perceptron Algorithm

```
1 loop $r = 1, \ldots, r_{max}$
2     loop $i = 1, \ldots, n$
3         Compute $\hat{} = f_D(\xi_i, \vec{})$
4         If $_i \neq \hat{}$
5             Increment $\alpha_i(\hat{}) \leftarrow \alpha_i(\hat{}) + 1$
6         End (if)
7     End loop ($i$)
8 End loop ($r$)
```

- Convegence
  - see Collins (2002) and Altun et al. (2003)

What about the Argmax?

- For dual perceptron it's easy!
- Decompose $f(\xi,) = f_1(\xi,) + f_2(\xi,)$ with

$$f_1(\xi,) = \sum_{\sigma,\tau} a(\sigma,\tau) \sum_s [[y^{s-1} = \sigma \ \wedge \ y^s = \tau]]$$

$$a(\sigma,\tau) = \sum_{i,\bar{\ }\neq_i} \alpha_i(\bar{\ }) \sum_t [[\bar{y}^{t-1} = \sigma \ \wedge \ \bar{y}^t = \tau]]$$

- and

$$f_2(\xi,) = \sum_{s,\sigma} [[y^s = \sigma]] \sum_{i,t} b(i,t,\sigma) K_x(x^s, x_i^t),$$

$$b(i,t,\sigma) = \sum_{\neq_i} [[y^t = \sigma]] \alpha_i()$$

- (homework: show that $f = f_1 + f_2$!)

Correspondence to Viterbi Algorithm

- $a(\sigma, \tau)$ corresponds to transition probabilities $P(y_t = \tau | y_{t-1} = \sigma)$

- for observation scores compute:
  - $B_i^{s\sigma} = \sum_j \sum_t b(j, t, \sigma) k(x_i^s, x_j^t)$
  - $B_i^{s\sigma}$ corresponds to $P(x_{i,s} | y_s = \sigma)$

- Note that a and b (or B) are scores and can be interpreted as log-probs.

- a and B can be directly plugged into log-Viterbi algorithm

- Equivalence between log-Viterbi ($\log(P(|\xi))$ and $f(\xi,)$

Named Entity Recognition

Example: Como (O) contrapartida (O) Deutsche (C-B) Telekom (C-I) vender (O) al (O) consorcio (O) francs (O) su (O) participacion (O) del (O) por (O) ciento (O) en (O) el (O) empresa (O) mixta (O) britnica (O) MetroHoldings (C-B).

(see Altun et al. (2003))

Natural Language Parsing

(see Collins&Duffy, 2002)

BioCreative

- Detection of gene and protein names in biomedical abstracts

  (Brefeld et al., 2005)

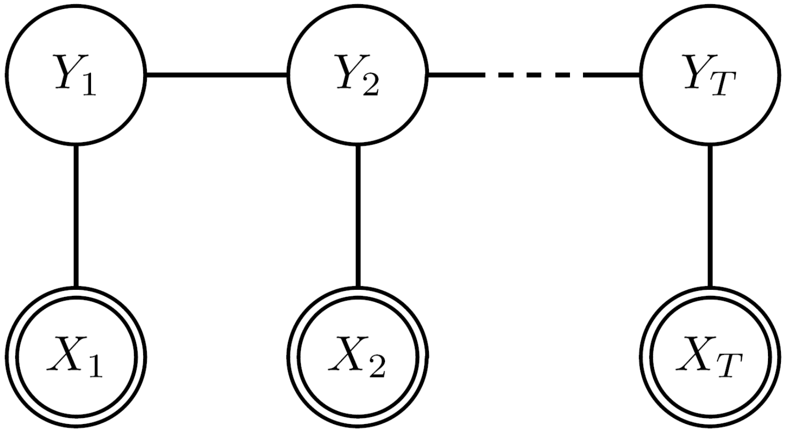
Summary

- Perceptrons for CRFs
  - aka generalized/structured perceptron
- pos:
  - easy to implement
  - efficient training process
- neg:
  - depends on ordering
  - no confidences
  - only 0/1 loss

Outlook

- Remedy: structural SVMs!

Recall: Conditional MRF



$$
\begin{aligned}
P(|\xi) &= \frac{1}{Z(\xi)} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i) \\
&\propto \langle \wedge, \Phi(\xi,) \rangle
\end{aligned}
$$

Recall: Generalized Linear Models

- $\xi =$ Bob jagt den Hund
- We want

$$[N, V, A, N] = \langle \wedge, \Phi(\xi, \cdot) \rangle$$

- Equivalent representation:

$$\begin{aligned}
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, A, A]) \rangle \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, A, N]) \rangle \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, N, A]) \rangle \\
\vdots \qquad &> \qquad \vdots \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [V, V, V, V]) \rangle
\end{aligned}$$

Recall: Primal/Dual Perceptron

- Primal perceptron:
  - Decision function: $f(\xi,) = \langle \wedge, \Phi(\xi,) \rangle$
  - Update rule: $\wedge \leftarrow \wedge + \Phi(\xi_i,_i) - \Phi(\xi_i, \hat{})$
- Dual perceptron:
  - Use relation: $\wedge = \sum_{i=1}^{n} \sum_{\bar{} \neq_i} \alpha_i(\bar{}) \left(\Phi(\xi_i,_i)\rangle - \Phi(\xi_i, \bar{})\right)$
  - Decision function:

$$f(\xi',') = \sum_{i=1}^{n} \sum_{\bar{} \neq_i} \alpha_i(\bar{}) \left(\langle \Phi(\xi_i,_i), \Phi(\xi',')\rangle - \langle \Phi(\xi_i, \bar{}), \Phi(\xi',')\rangle \right)$$

  - Update rule: $\alpha_i(\hat{}) \leftarrow \alpha_i(\hat{}) + 1$

Primal/Dual Algorithm

```
loop $r = 1, \dots, r_{max}$
    loop $i = 1, \dots, n$
        Compute prediction $\hat{}$
        If $_i \neq \hat{}$
            Update $\wedge$ (primal) or $\alpha_i(\hat{})$ (dual)
        End (if)
    End loop ($i$)
End loop ($r$)
```

How to Compute the Prediction in Step 3?

- What is the relation between...
  - Viterbi algorithm
  - max-product algorithm
  - max-sum algorithm
  - scoring function $f(\xi,)$
  - ???

- How can we compute $-f(\xi, \bar{})$?
- Answer: use log-Viterbi = max-sum algorithm!

Recall: Homework

- Dual perceptron:
- Decompose $f(\xi,) = f_1(\xi,) + f_2(\xi,)$ with

$$f_1(\xi,) = \sum_{\sigma,\tau} a(\sigma,\tau) \sum_s [[y^{s-1} = \sigma \,\wedge\, y^s = \tau]]$$

$$a(\sigma,\tau) = \sum_{i,\bar{}\neq_i} \alpha_i(\bar{}) \sum_t [[\bar{y}^{t-1} = \sigma \,\wedge\, \bar{y}^t = \tau]]$$

- and

$$f_2(\xi,) = \sum_{s,\sigma} [[y^s = \sigma]] \sum_{i,t} b(i,t,\sigma) K_x(x^s, x_i^t),$$

$$b(i,t,\sigma) = \sum_{\neq_i} [[y^t = \sigma]] \alpha_i()$$

Recall: Viterbi Algorithm

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

- Viterbi = max-product algorithm
  - define $\delta_{t+1}(\sigma) = \max_{y_1,\ldots,y_t} \ P(y_1, \ldots, y_{t+1} = \sigma, x_1, \ldots, x_{t+1})$
  - best score along a single path that ends in state σ at time $t + 1$

- log-Viterbi = max-sum algorithm
  - define $\delta_{t+1}(\sigma) = \max_{y_1,\ldots,y_t} \ \log P(y_1, \ldots, y_{t+1} = \sigma, x_1, \ldots, x_{t+1})$
  - apply $\delta_{t+1}(\sigma_i)$ recursively

Log-Viterbi Algorithm

- initialize $\delta_1(\sigma) = \log P(y_1 = \sigma) + \log P(x_1|y_1 = \sigma)$
- initialize $\psi_1(\sigma) = 0$
- loop $\sigma \in \Sigma$ and $t = 2, \ldots, T$:
  - $\delta_t(\sigma) = \big[\max_{\tau} \delta_{t-1}(\tau) + \log P(y_t = \sigma | y_{t-1} = \tau)\big] + \log P(x_t|y_t = \sigma)$
  - $\psi_t(\sigma) = \big[_{\tau} \delta_{t-1}(\tau) + \log P(y_t = \sigma | y_{t-1} = \tau)\big] + \log P(x_t|y_t = \sigma)$
- termination: $y_T^* =_{\sigma} \delta_T(\sigma)$
- loop $t = T, \ldots, 2$
  - $y_{t-1}^* = \psi_t(y_t^*)$

Scoring Function $f(\xi,)$

- Capture $\log P(y_1 = \sigma)$ implictely by adding constant label y_0.

- Observation probabilities:

$$\log P(x_t|y_t = \sigma) \propto \underbrace{\sum_{j} \sum_{s=1}^{T_j} \sum_{\bar{}\neq_i} [[y^t = \sigma]] \alpha_i(\bar{}) k(x_t, x_{j,s})}_{b(\sigma,x_t)}$$

- Transition probabilities:

$$\log P(y_t = \tau|y_{t-1} = \sigma) \propto \underbrace{\sum_{i,\bar{}\neq_i} \alpha_i(\bar{}) \sum_{t} [[\bar{y}^{t-1} = \sigma \ \wedge \ \bar{y}^t = \tau]]}_{a(\sigma,\tau)}$$

It holds...

Theorem

Given n input-output pairs of sequences of length T_i for $1 \leq i \leq n$, let Σ denote the output alphabet with $|\Sigma| < \infty$. Let f be defined as

$$f(\xi,) = \qquad \sum_{i=1}^{n} \sum_{i} \alpha_i(\bar{\ }) \left(\langle \Phi(\xi_i, {}_i), \Phi(\xi,) \rangle - \langle \Phi(\xi_i, \bar{\ }), \Phi(\xi,) \rangle \right),$$

where $\Phi(\xi,)$ denotes the joint feature map. Then for all $\alpha_i(\bar{\ }) \geq 0$ and any observation sequence ξ of length T,

$$\hat{\ } =_{\bar{\ } \in \Sigma^T} f(\xi, \bar{\ })$$

can be computed with a Viterbi algorithm in time $\mathcal{O}(T|\Sigma|^2)$.

Proof:

- The model f has the form

$$f(\xi,) = \sum_{i=1}^{n} \sum_{i} \alpha_i(\bar{}) \left(\langle \Phi(\xi_i, _i), \Phi(\xi,) \rangle - \langle \Phi(\xi_i, \bar{}), \Phi(\xi,) \rangle \right)$$

$$= \sum_{i=1}^{n} \sum_{i} \alpha_i(\bar{}) \left(\sum_{s,t} \left([[y_{i,s} = y_t]] - [[\bar{y}_s = y_t]] \right) k(x_{i,s}, x_t) \right.$$

$$\left. + \sum_{s,t} [[y_{i,s-1} = y_{t-1} \wedge y_{i,s} = y_t]] - [[\bar{y}_{s-1} = y_{t-1} \wedge \bar{y}_s = y_t]] \right).$$

- Make the dependency on labels $\sigma, \tau \in \Sigma$ explicit by summing over all transitions and observation states

$$f(\xi,) = \sum_{\sigma,\tau \in \Sigma} \sum_{i, \bar{} \neq _i} \alpha_i(\bar{}) \Big(\sum_{s,t} \left([[y_{i,s} = \sigma]] - [[\bar{y}_s = \sigma]] \right) [[y_t = \tau]] k(x_{i,s}, x_t)$$

$$+ \sum_{s,t} \Big([[y_{i,s-1} = \sigma \wedge y_{i,s} = \tau]] - [[\bar{y}_{s-1} = \sigma \wedge \bar{y}_s = \tau]] \Big)$$

$$\times [[y_{t-1} = \sigma \wedge y_t = \tau]] \Big).$$

Proof Contd.

The transition scores from label σ to label τ are now given by

$$a(\sigma,\tau)=\sum_{i=1}^{n}\sum_{i}\alpha_i(\bar{})\Big(\sum_{t=1}^{T_i}[[y_{i,t-1}=\sigma\wedge y_{i,t}=\tau]]-[[\bar{y}_{t-1}=\sigma\wedge\bar{y}_t=\tau]]\Big)$$

and observation scores for label $y_s=\sigma$ and observation x_s by

$$b(\sigma,x)=\sum_{i=1}^{n}\sum_{t=1}^{T_i}\sum_{i}\alpha_i(\bar{})\,([[y_{i,t}=\sigma]]-[[\bar{y}_t=\sigma]])\,k(x_{i,t},x).$$

...

The hypothesis $f(\xi,)$ can be rewritten in terms of transition scores $a(\sigma,\tau)$ and observation scores $b(\sigma,x)$

$$f(\xi,) = \underbrace{\sum_{\sigma,\tau\in\Sigma} a(\sigma,\tau)\sum_{s=1}^{T}[[y_{s-1}=\sigma \wedge y_s=\tau]]}_{=:f_a(\xi,)} + \underbrace{\sum_{s=1}^{T}\sum_{\sigma\in\Sigma}[[y_s=\sigma]]b(\sigma,x_s)}_{=:f_b(\xi,)}.$$

where f_a weights the occurences of neighboring labels in by corresponding scores of the model and f_b determines how well observations x_s fit to their labels y_s given the model. To decode the top scoring sequence we define

$$\delta_t(\sigma) = \max_{y_1,\ldots,y_{t-1}} f(\xi, y_1,\ldots,y_{t-1},y_t=\sigma), \tag{5}$$

that is, $\delta_t(\sigma)$ denotes the top scoring partial sequence up to position $t-1$ where $y_t=\sigma$.

Mathematical Induction: The Base Case

We first show by induction that

$$\delta_{t+1}(\sigma) = \max_{\tau \in \Sigma} \left[\delta_t(\tau) + a(\tau, \sigma)\right] + b(\sigma, x_{t+1}) \tag{6}$$

holds. The initialization is simply given by

$$\begin{aligned} \delta_0(\sigma) &= 0, \quad \forall \sigma \in \Sigma \\ \delta_1(\sigma) &= \max_{\tau \in \Sigma} \left[\delta_t(\tau) + a(\tau, \sigma)\right] + b(\sigma, x_{t+1}) \\ &= a(\epsilon, \sigma) + b(\sigma, x_1). \end{aligned}$$

The Inductive Step

The recursion step is given for $2 \leq t \leq T$ by

$$\begin{aligned}
\delta_t(\sigma) &= \max_{y_1,\ldots,y_{t-1}} f(\xi, y_1, \ldots, y_{t-1}, y_t = \sigma) \\
&= \max_{y_1,\ldots,y_{t-1}} \sum_{\tau,\bar{\tau}\in\mathcal{Y}} a(\tau,\bar{\tau}) \sum_{s=2}^{t-1} [[y_{s-1} = \tau \wedge y_s = \bar{\tau}]] \\
&\qquad + \sum_{\tau\in\Sigma} a(\tau,\sigma)[[y_{t-1} = \tau \wedge y_t = \sigma]] \\
&\qquad + \sum_{s=1}^{t-1} \sum_{\tau\in\Sigma} [[y_s = \tau]] b(\tau, x_s) + [[y_t = \sigma]] b(\sigma, x_t) \\
&= \max_{\sigma^\star} \max_{y_1,\ldots,y_{t-2}} \sum_{\tau,\bar{\tau}\in\mathcal{Y}} a(\tau,\bar{\tau}) \sum_{s=2}^{t-2} [[y_{s-1} = \tau \wedge y_s = \bar{\tau}]] \\
&\qquad + \sum_{\tau\in\Sigma} a(\tau,\sigma^\star)[[y_{t-2} = \tau \wedge y_{t-1} = \sigma^\star]] \\
&\qquad + a(\sigma^\star,\sigma)[[y^{t-1} = \sigma^\star \wedge y^t = \sigma]] \\
&\qquad + \sum_{s=1}^{t-2} \sum_{\tau\in\Sigma} [[y_s = \tau]] b(\tau, x_s) + b(\sigma^\star, x_{t-1}) + b(\sigma, x_t)
\end{aligned}$$

The Inductive Step Contd.

$$= \max_{\sigma^\star} \left[\max_{y_1,\ldots,y_{t-2}} f(\xi, y_1, \ldots, y_{t-2}, y_{t-1} = \sigma^\star) + a(\sigma^\star, \sigma) \right] + b(\sigma, x_t)$$
$$= \max_{\sigma^\star} \left[\delta_{t-1}(\sigma^\star) + a(\sigma^\star, \sigma) \right] + b(\sigma, x_t).$$

Thus, the top scoring sequence has the score

$$\max f(\xi,) = \max_{\sigma \in \Sigma} \delta_T(\sigma).$$

We only sketch the extension to the argument of the maximum since it is analoguous to the regular Viterbi algorithm. We introduce path variables $\varphi_t(\sigma)$ that are initialized by $\varphi_1(\sigma) = \epsilon$ for all $\sigma \in \Sigma$.

Computing the Argmax

The sequence $\varphi_t(\sigma)$ is then defined recursively for $2 \leq t \leq T$ by

$$\varphi_t(\sigma) =_{\sigma^\star \in \Sigma} \left[\delta_{t-1}(\sigma^\star) + a(\sigma^\star, \sigma)\right].$$

Once the $\delta_t(\sigma)$ of Theorem 2 are fixed, the optimal label sequence can be found by backtracking

$$y_T^\star =_{\sigma \in \Sigma} \delta_T(\sigma)$$
$$y_t^\star = \varphi_{t+1}(y_{t+1}^\star) \quad \text{for} \quad t = T-1, \ldots, 1.$$

Conclusion

Given the transition matrix $[\mathbf{A}]_{\sigma,\tau} = a(\sigma, \tau)$ and the observation matrix $[\mathbf{B}_\xi]_{\sigma,t} = b(\sigma, x_t)$ for input ξ, the computation of δ and φ for a fixed t and $\sigma \in \Sigma$ involves visiting $|\Sigma|$ predecessors; thus, for a sequence of length T the time needed is in $\mathcal{O}(T|\Sigma|^2)$. This concludes the proof. $\square$

Visualization

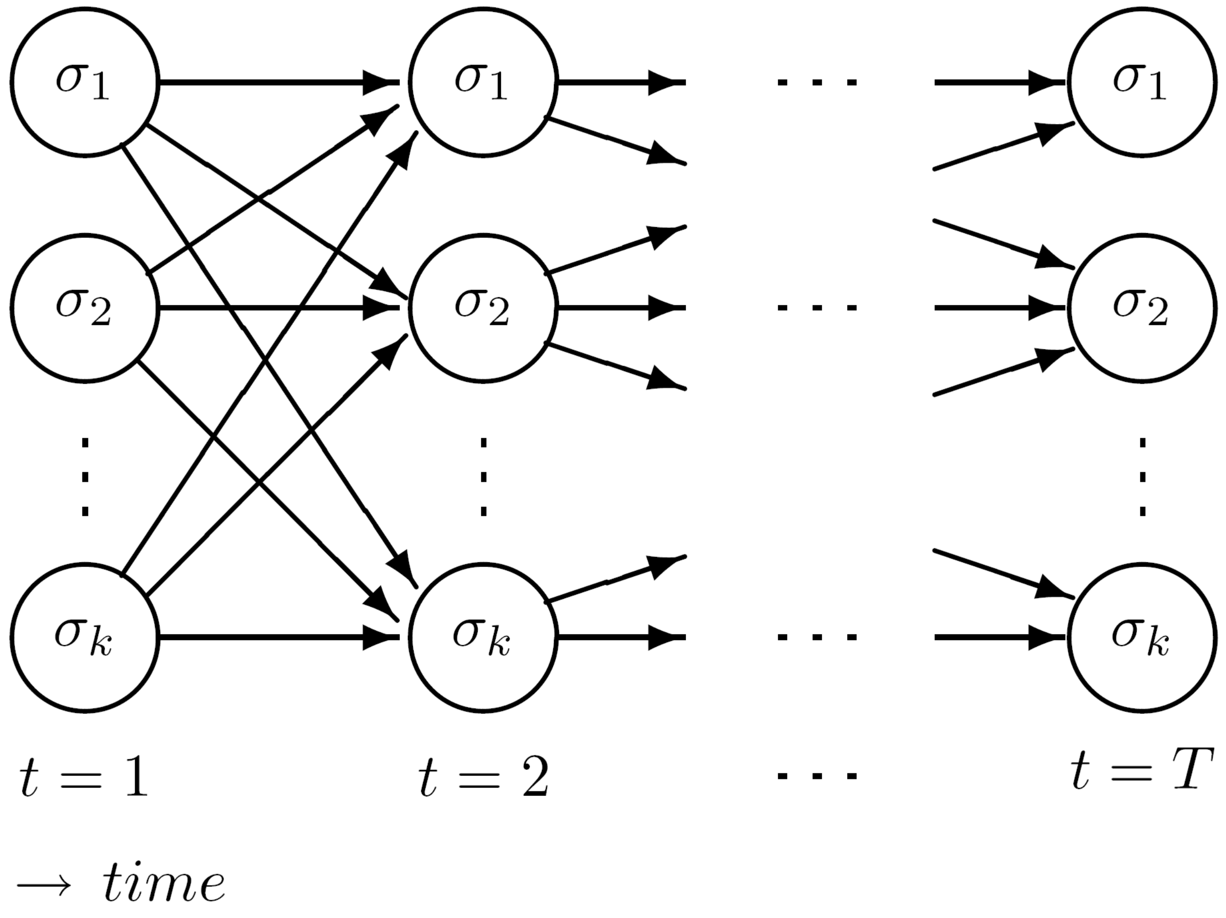


Figure: Visualization of a trellis over the alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$.

Summary

- Equivalence: Dual perceptron $f(\xi,)$ and Viterbi algorithm
  - similar proof for primal perceptron
- pos:
  - easy to implement
  - efficient training process
- neg:
  - depends on ordering
  - no confidences
  - only 0/1 loss

From Perceptrons to SVMs

- Add confidence to decision
- Incorporate arbitrary (structured) loss functions
- Impact of ordering resolved by quadratic programming

Confidence Term

- Perceptron:

$$\begin{aligned}
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, A, A]) \rangle \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, A, N]) \rangle \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle &> \langle \wedge, \Phi(\xi, [A, A, N, A]) \rangle \\
\vdots \quad &> \quad \vdots
\end{aligned}$$

- Now, add a confidence $\bar{\gamma}$:

$$\begin{aligned}
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle - \langle \wedge, \Phi(\xi, [A, A, A, A]) \rangle &\geq \bar{\gamma} \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle - \langle \wedge, \Phi(\xi, [A, A, A, N]) \rangle &\geq \bar{\gamma} \\
\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle - \langle \wedge, \Phi(\xi, [A, A, N, A]) \rangle &\geq \bar{\gamma} \\
\vdots \qquad\qquad\qquad &\vdots
\end{aligned}$$

Optimization Problem

$$\max_{\bar{\gamma},\wedge} \qquad \frac{\bar{\gamma}}{||\wedge||}$$

s.t. $\qquad \forall_{i=1}^{n}, \forall_{\neq_i} : \langle \wedge, \Phi(\xi_{i,i}) \rangle - \langle \wedge, \Phi(\xi_{i,}\bar{}) \rangle \geq \bar{\gamma}$

- We call
  - $\wedge$ the weight vector
  - $\bar{\gamma}$ the functional margin
  - $\gamma = \frac{\bar{\gamma}}{||\wedge||}$ the geometrical margin

- Problem: $\bar{\gamma}$ and $\wedge$ interdepend!
  - Remedy: fix one, solve for the other
  - Common approach: $\bar{\gamma} = 1$.

Structural Hard-margin SVM

$$\min_{\wedge} \quad \frac{1}{2}||\wedge||^2$$

s.t. $\forall_{i=1}^{n}, \forall_{\neq_i} : \langle \wedge, \Phi(\xi_{i,i})\rangle - \langle \wedge, \Phi(\xi_i, \bar{})\rangle \geq 1$

- Converges only when data is linear separable
- Remedy: allow for pointwise relaxations of the margin constraint
  – introduce slack variables ξ_i for input examples

Structural Soft-margin SVM

$$\min_{\wedge} \quad \frac{1}{2}||\wedge||^2 + \sum_{i=1}^{n} \xi_i$$

s.t. $\quad \forall_{i=1}^{n}, \forall_{\neq_i} : \langle \wedge, \Phi(\xi_i,_i)\rangle - \langle \wedge, \Phi(\xi_i, \bar{})\rangle \geq 1 - \xi_i$

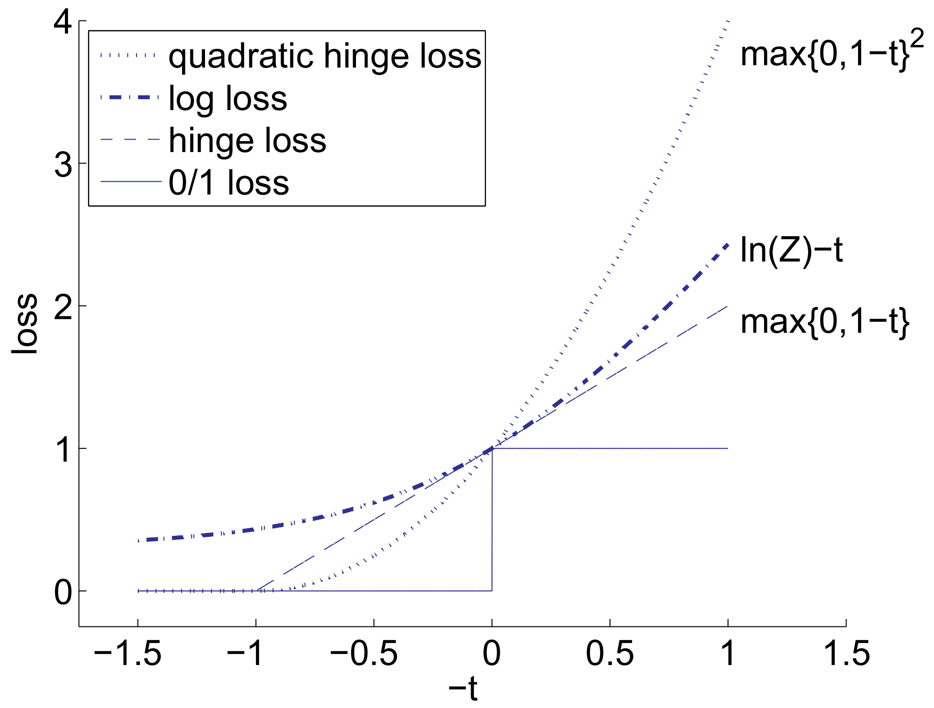
$\quad \forall_{i=1}^{n} : \xi_i \geq 0$

- Sum of slacks upper bounds 0/1 loss

- Now: maximize margin between true $_i$ and best runner-up^{-}

- Alternative formulation:
  – slack $\xi_{i\bar{}}$ are bound to constraint $\langle \wedge, \Phi(\xi_i,_i)\rangle - \langle \wedge, \Phi(\xi_i, \bar{})\rangle$
  – computationally demanding

Hinge-loss



- SVM implicitely implements a hinge loss (solve for slacks)
- Hinge loss can be rescaled to incorporate arbitrary loss functions
  - Let $\Delta({}_i, \hat{})$ denote a structural loss.
  - $\Delta : \mathcal{Y} \times \mathcal{Y} \to \Re_0^+$.
  - $\Delta({}_i, {}_i) = 0$

Examplary Loss Functions

- 0/1 loss: $\Delta(,\bar{}) = [[==\bar{}]]$

- Hamming loss for sequences

$$\Delta(,\bar{}) = T - \sum_{t=1}^{T} [[y_t == \bar{y}_t]]$$

  – Property: decomposes across the cliques!

Margin-rescaling

- Taskar et al. (2004)
- Rescale the (functional) margin by actual loss

$$\begin{aligned}
\min_{\wedge} \quad & \frac{1}{2}||\wedge||^2 + \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & \forall_{i=1}^{n}, \forall_{\neq i} : \langle \wedge, \Phi(\xi_i, _i)\rangle - \langle \wedge, \Phi(\xi_i, \bar{})\rangle \geq \Delta(_i, \bar{}) - \xi_i \\
& \forall_{i=1}^{n} : \xi_i \geq 0
\end{aligned}$$

– Implicit hinge loss upper bounds Δ

– Most strongly violated constraint:

$$\text{-}\Big(\Delta(_i, \bar{}) - \big(\langle \wedge, \Phi(\xi_i, _i) - \Phi(\xi_i, \bar{})\rangle\big)\Big)$$

Slack-rescaling

- Tsochantaridis et al. (2005)
- Rescale slack variables by actual loss

$$\min_{\wedge} \qquad \frac{1}{2}||\wedge||^2 + \sum_{i=1}^{n} \xi_i$$

s.t. $$\forall_{i=1}^{n}, \forall_{\bar{}\neq_i} : \langle \wedge, \Phi(\xi_i,_i)\rangle - \langle \wedge, \Phi(\xi_i,\bar{})\rangle \geq 1 - \frac{\xi_i}{\Delta(_i,\bar{})}$$

$$\forall_{i=1}^{n} : \xi_i \geq 0$$

– Implicit hinge loss upper bounds Δ

– Most strongly violated constraint:

$$\text{-}\Big(1 - \Delta(_i,\bar{}) \times \big(\langle \wedge, \Phi(\xi_i,_i) - \Phi(\xi_i,\bar{})\rangle\big)\Big)$$

Implications

- Loss Δ decomposes across the cliques of the graph
  – Margin-rescaling is easily integrated into inference
  – Slack-rescaling difficult

- Loss not decomposable
  – Both difficult!

- In practice, slack-rescaling often better than margin-rescaling
  – rarely applicable (needs good approximation or enumerable sets)

Recall:

- relation between
  - $P(|\xi)$
  - model $f(\xi,)$
  - log-Viterbi algorithm

- intuition
  - $f(\xi,)$ = how good does fits to ξ
  - log-Viterbi: find top-scoring

Towards Structured Support Vector Machines

- Add confidence to decision
- Incorporate arbitrary (structured) loss functions
- Impact of ordering resolved by quadratic programming

Confidence Term

- Perceptron:

$$\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle > \langle \wedge, \Phi(\xi, [A, A, A, A]) \rangle$$
$$\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle > \langle \wedge, \Phi(\xi, [A, A, A, N]) \rangle$$
$$\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle > \langle \wedge, \Phi(\xi, [A, A, N, A]) \rangle$$
$$\vdots \quad > \quad \vdots$$

- Now, add a confidence $\bar{\gamma}$:

$$\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle - \langle \wedge, \Phi(\xi, [A, A, A, A]) \rangle \geq \bar{\gamma}$$
$$\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle - \langle \wedge, \Phi(\xi, [A, A, A, N]) \rangle \geq \bar{\gamma}$$
$$\langle \wedge, \Phi(\xi, [N, V, A, N]) \rangle - \langle \wedge, \Phi(\xi, [A, A, N, A]) \rangle \geq \bar{\gamma}$$
$$\vdots \qquad \qquad \vdots$$

Optimization Problem

$$\max_{\bar{\gamma}, \wedge} \quad \frac{\bar{\gamma}}{|| \wedge ||}$$

s.t. $\quad \forall_{i=1}^{n}, \forall_{\neq_i} : \langle \wedge, \Phi(\xi_{i,i}) \rangle - \langle \wedge, \Phi(\xi_{i,\bar{}}) \rangle \geq \bar{\gamma}$

- We call
  - $\wedge$ the weight vector
  - $\bar{\gamma}$ the functional margin
  - $\gamma = \frac{\bar{\gamma}}{||\wedge||}$ the geometrical margin
- Problem: $\bar{\gamma}$ and $\wedge$ interdepend!
  - Remedy: fix one, solve for the other
  - Common approach: $\bar{\gamma} = 1$.

Structural Hard-margin SVM

$$\min_{\wedge} \quad \frac{1}{2}||\wedge||^2$$

s.t. $\forall_{i=1}^{n}, \forall_{\neq_i} : \langle \wedge, \Phi(\xi_i,_i\,)\rangle - \langle \wedge, \Phi(\xi_i, \bar{})\rangle \geq 1$

- Converges only when data is linear separable
- Remedy: allow for pointwise relaxations of the margin constraint
  - introduce slack variables ξ_i for input examples

Structural Soft-margin SVM

$$\min_{\wedge} \qquad \frac{1}{2}||\wedge||^2 + C\sum_{i=1}^{n}\xi_i$$

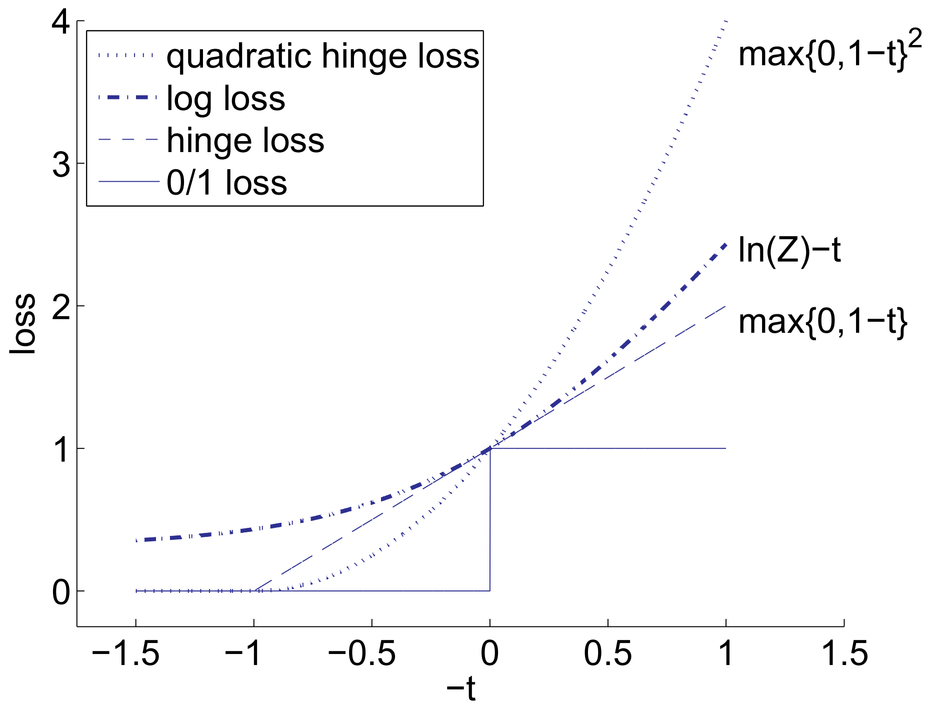
s.t. $\qquad \forall_{i=1}^{n}, \forall_{\bar{}\neq_i} : \langle \wedge, \Phi(\xi_i,_i)\rangle - \langle \wedge, \Phi(\xi_i,\bar{})\rangle \geq 1 - \xi_i$

$\qquad \forall_{i=1}^{n} : \xi_i \geq 0$

- Maximize margin between true $_i$ and best runner-up $^-$
  - Sum of slacks upper bounds 0/1 loss
  - Trade-off parameter $C > 0$

- Alternative formulation:
  - slack $\xi_{i\bar{}}$ are bound to constraint $\langle \wedge, \Phi(\xi_i,_i)\rangle - \langle \wedge, \Phi(\xi_i,\bar{})\rangle$
  - computationally demanding

Hinge-loss



- SVM implicitely implements a hinge loss (solve for slacks)
- Hinge loss can be rescaled to incorporate arbitrary loss functions
  - Let $\Delta(_i, \hat{})$ denote a structural loss.
  - $\Delta : \mathcal{Y} \times \mathcal{Y} \to \Re_0^+$.
  - $\Delta(_i,_i\,) = 0$

Examplary Loss Functions

- 0/1 loss: $\Delta(,\bar{}) = [[==\bar{}]]$

- Hamming loss for sequences

$$\Delta(,\bar{}) = T - \sum_{t=1}^{T} [[y_t == \bar{y}_t]]$$

  – Property: decomposes across the cliques!

Risk Minimization

- We want to minimize the theoretical risk (the generalization error)

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta\big(, f(\xi,)\big) dP(\xi,)$$

- In general, we don't know $P(\xi,)$
  – Remedy: Use training sample instead!

- Minimize the empirical risk

$$\hat{R}(f) = \sum_{i=1}^{n} \Delta\big(_i, f(\xi_i,)\big)$$

Idea

- SVMs minimize the (regularized) empirical risk:

$$\hat{R}(f) = \sum_{i=1}^{n} \Delta\big({}_i, f(\xi_i,)\big)$$

- Sum of slacks $\sum_i \xi_i$ upper bounds empirical risk
- Slack variable ξ_i denotes the error for input ξ_i
  – Now: Find maximal error wrt Δ

Margin-rescaling

- Taskar et al. (2004)
- Rescale the (functional) margin by actual loss

$$\begin{array}{ll} \min\limits_{\wedge} & \frac{1}{2}||\wedge||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} & \forall_{i=1}^{n}, \forall_{\bar{}\neq_i} : \langle \wedge, \Phi(\xi_i, _i)\rangle - \langle \wedge, \Phi(\xi_i, \bar{})\rangle \geq \Delta(_i, \bar{}) - \xi_i \\ & \forall_{i=1}^{n} : \xi_i \geq 0 \end{array}$$

– Implicit hinge loss upper bounds Δ

– Most strongly violated constraint:

$$_{\bar{}\neq_i}\left(\underbrace{\Delta(_i,\bar{}) - \left(\langle \wedge, \Phi(\xi_i, _i) - \Phi(\xi_i, \bar{})\rangle\right)}_{\xi_i}\right)$$

Slack-rescaling

- Tsochantaridis et al. (2005)
- Rescale slack variables by actual loss

$$\min_{\wedge} \qquad \frac{1}{2}||\wedge||^2 + C\sum_{i=1}^{n} \xi_i$$

s.t. $$\forall_{i=1}^{n}, \forall_{\bar{}\neq_i} : \langle \wedge, \Phi(\xi_i,_i)\rangle - \langle \wedge, \Phi(\xi_i,\bar{})\rangle \geq 1 - \frac{\xi_i}{\Delta(_i,\bar{})}$$

$$\forall_{i=1}^{n} : \xi_i \geq 0$$

- – Implicit hinge loss upper bounds Δ
- – Most strongly violated constraint:

$$\bar{}_{\neq_i}\left(\underbrace{1 - \Delta(_i,\bar{}) \times \left(\langle \wedge, \Phi(\xi_i,_i) - \Phi(\xi_i,\bar{})\rangle\right)}_{\xi_i}\right)$$

Implications

- Loss Δ decomposes across the cliques of the graph
  – Margin-rescaling is easily integrated into inference
  – Slack-rescaling difficult
- Loss not decomposable
  – Both difficult!
- In practice, slack-rescaling often better than margin-rescaling
  – rarely applicable (needs good approximation or enumerable sets)

Example

- Margin rescaling for sequences / Viterbi algorithm
- Remainder: 0/1 loss for simplicity

Towards Dual SVMs

- Intergrate constraints into objective
  - Apply Lagrange's Theorem
  - Lagrange multipliers $\alpha_i()$ and μ_i

- Build Lagrangian:

$$
\begin{aligned}
L =& \frac{1}{2}|| \wedge ||^2 + C \sum_{i=1}^{n} \xi_i \\
&- \sum_{i=1}^{n} \sum_{\neq_i} \alpha_i() \langle \wedge, \Phi(\xi_{i},_i\,) \rangle - \langle \wedge, \Phi(\xi_i,) \rangle - 1 + \xi_i \\
&- \sum_{i=1}^{n} \beta_i \xi_i
\end{aligned}
$$

- Minimum of Lagrangian is a saddle-point
  - max wrt α, μ, min wrt $\wedge, \xi$

Partial Derivatives: ξ_i

- Compute partial derivatives wrt ξ:

$$\frac{\partial L}{\partial \xi_i} = C - \sum_{\bar{\ } \neq_i} \alpha_i(\bar{\ }) - \beta_i \quad \overset{!}{=} \quad 0 \tag{7}$$

- Using the non-negativity of α and β yields

$$\forall_{i=1}^{n}: \quad 0 \leq \sum_{\bar{\ } \neq_i} \alpha_i(\bar{\ }) \leq C$$

Partial Derivatives: $\wedge$

- Compute partial derivatives wrt $\wedge$:

$$\frac{\partial L}{\partial \wedge} = \mathbf{w} - \sum_{i=1}^{n} \sum_{\bar{} \neq i} \alpha_i(\bar{})\Big(\Phi(\xi_{i,i}\,) - \Phi(\xi_i,\bar{})\Big) \overset{!}{=} 0.$$

- We obtain:

$$\wedge = \sum_{i=1}^{n} \sum_{\bar{} \neq i} \alpha_i(\bar{})\Big(\Phi(\xi_{i,i}\,) - \Phi(\xi_i,\bar{})\Big)$$

- Recall: dual perceptron!
  - Definition of $\wedge$ is equivalent
  - α's act like counters