Forecasting & Simulation

Ulf Brefeld

Machine Learning Group
Leuphana University of Lüneburg

June 5, 2024

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

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 $P(X = x) \in \mathbb{R}$

random variables (in terms of sets)

let X be a random variable with domain $\Omega = dom(X)$
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:*

- $\sum_{x \in \Omega} P(X = x) = 1$

probability 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
  $$\forall_{x \in dom(X)} : \; P(X = x) = \frac{1}{6}$$
  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

- *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., $X = (X_1, .., X_n)$, $Y = (Y_1, .., Y_m)$

joint distributions – implications:

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

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

Bayes Rule

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

- importance lies in its interpretation and use:

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

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

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
Shows *importance of the prior*

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 $Y_1, .., Y_K$

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

- 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}

cheat sheet – 1

- a random variable X assigns probabilties $P(Xx) \in \mathbb{R}$ to values $x \in dom(x)$
- 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)

cheat sheet – 2

- implications:

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

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

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

- definition: *inference* is the problem to compute

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

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 θ

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 **x** and **y** are not iid!

- structure is determined by length of input sequence

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

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

  - capture dependencies between neighboring words

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 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:

Hidden Markov Models

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

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

maximum likelihood

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

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

  - $\delta_{t+1}(\sigma_i)$ 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: $y_T^* = \text{argmax}_i\, \delta_T(\sigma_i)$
- loop $t = T-1, \ldots, 1$

$$y_t^* = \psi_{t+1}(y_{t+1}^*)$$

Trellis

$\rightarrow$ *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 $P(\mathbf{x}, \mathbf{y})$
  - Thus also $P(\mathbf{x}) \rightarrow$ difficult in high dimensions!
  - we are interested in discriminative models $P(\mathbf{y}|\mathbf{x})$
  - HMMs optimize the wrong criterion!

- Remedy:
  - Learn the quantity of interest $P(\mathbf{y}|\mathbf{x})$ direclty!
  - Use Markov random field instead of Bayesian network
  - Condition joint probability on the observations
  - Conditional random fields

Summary: HMMs

- 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 $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$

- 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
  - $P(\text{unobserved token}|Y_i) = 0$ (remedy: smoothing techniques)
  - generative models optmize the wrong criterion

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

Markov Random Fields

HMM:

MRF:

- every BN can be translated into equivalent MRF (moralization)
  - but: not always necessary (i.g., web pages)
  - dependencies now bidirectional

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

Hammersley Clifford theorem

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

- 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:

$w^T \phi^{trans}$ inner product

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

potentials

basis functions

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

  - 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

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

indicator $[\![z]\!] = \begin{cases} 1: z \text{ true} \\ 0: \text{otherwise} \end{cases}$

- simple case: indicator functions

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

- similar to HMM
- later more...

Basis Functions: label-observation

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

$|\Sigma|$

Dict. size

- simple case: indicator functions

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

- similar to HMM
- later more...

Putting Everything Together...

$$P(\mathbf{x},\mathbf{y}) = \frac{1}{Z}\prod_{i=1}^{T}\overbrace{\exp\{\sum_{j=1}^{d_o} w_j^o \phi_j^o(x_i,y_i)\}}^{\psi^o(x_i,y_i)}\prod_{i=2}^{T}\overbrace{\exp\{\sum_{j=1}^{d_t} w_j^t \phi_j^t(y_{i-1},y_i)\}}^{\psi^t(y_{i-1},y_i)}$$

inner product

$$= \frac{1}{Z}\prod_{i=1}^{T}\exp\{\langle \mathbf{w}^o, \phi^o(x_i,y_i)\rangle\}\prod_{i=2}^{T}\exp\{\langle \mathbf{w}^t, \phi^t(y_{i-1},y_i)\rangle\}$$

draw product into exp. fct.

$$= \frac{1}{Z}\exp\{\sum_{i=1}^{T}\langle \mathbf{w}^o, \phi^o(x_i,y_i)\rangle\}\exp\{\sum_{i=2}^{T}\langle \mathbf{w}^t, \phi^t(y_{i-1},y_i)\rangle\}$$

draw w out of sum

$$= \frac{1}{Z}\exp\{\langle \mathbf{w}^o, \sum_{i=1}^{T}\phi^o(x_i,y_i)\rangle\}\exp\{\langle \mathbf{w}^t, \sum_{i=2}^{T}\phi^t(y_{i-1},y_i)\rangle\}$$

concatenate

$$= \frac{1}{Z}\exp\{\langle \underbrace{\begin{pmatrix}\mathbf{w}^o\\ \mathbf{w}^t\end{pmatrix}}_{=:\mathbf{w}}, \underbrace{\begin{pmatrix}\sum_{i=1}^{T}\phi^o(x_i,y_i)\\ \sum_{i=2}^{T}\phi^t(y_{i-1},y_i)\end{pmatrix}}_{=:\Phi(\mathbf{x},\mathbf{y})}\rangle\} = \frac{1}{Z}\exp\{\langle \mathbf{w}, \Phi(\mathbf{x},\mathbf{y})\rangle$$

relation to softmax:

$$\frac{1}{Z}\exp\{\langle w, \Phi(x,y)\rangle\}$$

$$Z = \sum_{x}\sum_{y} \exp\{w, \Phi(x,y)\}$$

$$\Rightarrow P(x,y) = \frac{\exp\{\langle w, \Phi(x,y)\rangle\}}{\sum_{x'}\sum_{y'} \exp\{\langle w, \Phi(x',y')\rangle\}}$$

Joint Feature Representation

- joint representation of input and output variables:
  $\Phi(\mathbf{x}, \mathbf{y}) = (\sum_{i=1}^{T} \phi^o(x_i, y_i), \sum_{i=2}^{T} \phi^t(y_{i-1}, y_i))'$
- Example for HMM-alike basis functions:
  - $\Phi(\mathbf{x}, \mathbf{y})$ 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, $dim(\Phi) = 721296$

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
  - $\phi^{trans}(y_{t-1}, y_t) \rightarrow \phi^{trans}(y_{t-1}, y_t; x_t)$
  - or even: $\phi^{trans}(y_{t-1}, y_t) \rightarrow \phi^{trans}(y_{t-1}, y_t; \mathbf{x})$
  - similarly: $\phi^{obs}(x_t, y_t) \rightarrow \phi^{obs}(\mathbf{x}, y_t)$
  - (alternative graph structure)

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

The Exponential Family

$$P(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle\}$$

- $P(\mathbf{x}, \mathbf{y})$ is a member in the exp. family, rewrite in canonical form

$$P(\mathbf{x}, \mathbf{y}) = \exp\{\langle \mathbf{w}, \boxed{\Phi(\mathbf{x}, \mathbf{y})} \rangle - \log Z\}$$

- Identify the terms:
  - $\Phi(\mathbf{x}, \mathbf{y})$ is the sufficient statistics
  - $\mathbf{w}$ is the natural parameter
  - $\log Z < \infty$ is the moment generating function

Conditional Markov Random Fields

- joint probability

$$P(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle\}$$

  - partition function: $Z = \sum_{\mathbf{x}} \sum_{\mathbf{y}} \exp\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle\}$

- condition on the observation
  - apply the rule: $P(\mathbf{y}|\mathbf{x}) = P(\mathbf{x}, \mathbf{y}) / P(\mathbf{x})$
  - obtain new partition function:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle\}$$

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

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle\}$$

$$P(x,y) = \frac{1}{Z} \exp\{\langle w, \Phi(x,y)\rangle\}$$

$$P(y|x) = ?$$

$$Z = \cancel{\sum_x} \sum_y \exp\{\langle w, \Phi(x,y)\rangle\}$$

$$\Rightarrow P(y|x) = \frac{\exp\{\langle w, \Phi(x,y)\rangle\}}{\sum_y \exp\{\langle w, \Phi(x,y)\rangle\}}$$

Training CRFs with Maximum Likelihood

- given n input output examples $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$

$\mathcal{L} = \prod_{n=1}^{N} P(Y_n \mid X_n)$

- the log-likelihood is given

$$\log \mathcal{L} = \sum_{i=1}^{n} \langle \cancel{\mathbf{w}}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \log Z(\mathbf{w} | \mathbf{x}_i)$$

- differentiating wrt **w** gives

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

emp. distr. − model distr. $\overset{!}{=} 0$

  - 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)
  - ...

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 $\mathbf{w}^*$ are found these are used as plug-in estimates $P(\mathbf{y}|\mathbf{x};\mathbf{w}^*)$
  - 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 $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}\, P(\mathbf{y}|\mathbf{x})$
  - computing MAP estimates is much cheaper than full posterior!

$$\hat{y} = \underset{\tilde{y}}{\text{argmax}}\; P(\tilde{y}|x) = \cancel{\underset{\tilde{y}}{\text{argmax}}\; \frac{1}{Z(x)} \exp\{\langle w, \Phi(x,\tilde{y})\rangle\}}$$

$$= \underset{\tilde{y}}{\text{argmax}}\; \langle w, \Phi(x,\tilde{y})\rangle$$

Computing MAP Estimates

- For MAP estimates compute

$$\begin{aligned}\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \cancel{\frac{1}{Z(\mathbf{x})}} \cancel{\exp}\{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})\rangle\} \\ &= \operatorname*{argmax}_{\mathbf{y}} \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})\rangle\end{aligned}$$

- because exp is a monotone function and $\frac{1}{Z(\mathbf{x})}$ is constant

- We arrive at:

$$\boxed{P(\cancel{\mathbf{x}}|\cancel{\mathbf{y}}) \propto \underbrace{\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})\rangle}_{=:f(\mathbf{x},\mathbf{y})}}$$

Outlook

- adapt $f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$ to data

  number of

- 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

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

- potential functions:

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

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

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

$$P(Y|X) = \frac{1}{Z(X)} \exp\left\{ \langle w, \Phi(x, y) \rangle \right\}$$

N — V — D — N

Bob — drags — the — dog

sum over all time steps!

$$\Phi(x, y) = \begin{pmatrix} 1+0+\dots \\ 0+0 \\ 0+0 \\ 0+1 \\ \vdots+0 \\ \vdots+0 \\ 0+0 \\ 0+0 \\ \vdots+0 \\ 0+1 \\ 1+1 \\ \vdots \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 3 \\ \vdots \end{pmatrix}$$

obs

trans

$$\sum \left(w_1 \Phi_j^{obs}(\dots) \right)$$

$$= \left(w_1 \sum \left(\quad \right) \right)$$

stop!

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:

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

- Rewrite conditional probability:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle \right\}$$

- Observation:

$$P(\mathbf{y}|\mathbf{x}) \propto \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$$

- MAP estimate:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\bar{\mathbf{y}}} P(\bar{\mathbf{y}}|\mathbf{x}) = \operatorname{argmax} \langle \mathbf{w}, \Phi(\mathbf{x}, \bar{\mathbf{y}}) \rangle$$

Example

- $\mathbf{x}$ = Bob jagt den Hund
- We want

$$[N, V, \overset{\text{Det}}{\cancel{A}}, N] = \underset{\bar{\mathbf{y}}}{\text{argmax}} \langle \mathbf{w}, \Phi(\mathbf{x}, \bar{\mathbf{y}}) \rangle$$

→ consider spheres of length 1: $+1/-1$

$\Phi(x, y)$

⇒ logistic regression

Example

- $\mathbf{x} =$ Bob jagt den Hund
- We want

$$[N, V, \cancel{A}\,D, N] = \underset{\bar{\mathbf{y}}}{\text{argmax}} \langle \mathbf{w}, \Phi(\mathbf{x}, \bar{\mathbf{y}}) \rangle$$

- Equivalent representation:

$$\begin{aligned}
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, \cancel{A}\,D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [A, A, A, A]) \rangle \\
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, \cancel{A}\,D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [A, A, A, N]) \rangle \\
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, \cancel{A}\,D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [A, A, N, A]) \rangle \\
\vdots \quad &> \quad \vdots \\
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, \cancel{A}\,D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [V, V, V, V]) \rangle
\end{aligned}$$

Example Contd.

- Another equivalent representation:

$$\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, A, N]) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}, [A, A, A, A]) \rangle > 0$$
$$\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, A, N]) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}, [A, A, A, N]) \rangle > 0$$
$$\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, A, N]) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}, [A, A, N, A]) \rangle > 0$$
$$\vdots \qquad \vdots$$
$$\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, A, N]) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}, [V, V, V, V]) \rangle > 0$$

- The other way round:
  - Update weight vector **w** in case of an error:

$$\boxed{\langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \max_{\bar{\mathbf{y}}} \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle < 0}$$

good — bad one

Primal Perceptron

- Simplify things:
  - Error: $\mathbf{y}_i \neq \hat{\mathbf{y}} = \text{argmax}_{\bar{\mathbf{y}}} \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle$

- Recall gradient of CRF:

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

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

$$\mathbf{w} \leftarrow \mathbf{w} + \underbrace{\Phi(\mathbf{x}_i, \mathbf{y}_i)}_{\text{true pair}} - \underbrace{\Phi(\mathbf{x}_i, \hat{\mathbf{y}})}_{\text{erroneous prediction}}$$

Primal Perceptron Algorithm

loop $r = 1, \ldots, r_{max}$

  loop $i = 1, \ldots, n$

    Compute $\hat{\mathbf{y}} = \text{argmax}_{\bar{\mathbf{y}}} \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle$

    If $\mathbf{y}_i \neq \hat{\mathbf{y}}$

      Update $\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}})$

    End (if)

  End loop (i)

End loop (r)

Compare to MC, ranking & AUC!

Convergence

Theorem (Extension of Novikoff)

Given n labeled examples $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, with $\mathbf{y}_i \in \mathcal{Y}(\mathbf{x}_i)$. Let r be the radius of the smallest hypersphere enclosing all difference vectors $\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}})$, for all i and $\bar{\mathbf{y}} \neq \mathbf{y}_i$,

$$r = \max_{1 \leq i \leq n} \max_{\substack{\bar{\mathbf{y}} \in \mathcal{Y}(\mathbf{x}_i) \\ \bar{\mathbf{y}} \neq \mathbf{y}_i}} \|\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}})\|.$$

If there exists a vector $\mathbf{w}^*$ such that

$$\forall_{i=1}^{n} \forall_{\bar{\mathbf{y}} \in \mathcal{Y}(\mathbf{x}_i)} \boxed{\langle \mathbf{w}^*, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}^*, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \geq \bar{\gamma}} \tag{1}$$

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

Proof of Theorem

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

$$\mathbf{y}_i \neq \hat{\mathbf{y}}_i = \underset{\bar{\mathbf{y}} \in \mathcal{Y}(\mathbf{x}_i)}{\text{argmax}} \langle \mathbf{w}^{(t-1)}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle.$$

The corresponding update step is given by

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i) \tag{3}$$

Multiplying Equation 3 with the optimal weight vector $\mathbf{w}^*$ yields

$$\begin{aligned}\langle \mathbf{w}^*, \mathbf{w}^{(t)} \rangle &= \langle \mathbf{w}^*, \mathbf{w}^{(t-1)} \rangle + \boxed{\langle \mathbf{w}^*, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}^*, \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i) \rangle} \\ &\geq \langle \mathbf{w}^*, \mathbf{w}^{(t-1)} \rangle + \bar{\gamma}\end{aligned}$$

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

Proof of Theorem (Contd.)

Now we bound $\|\mathbf{w}^{(t)}\|^2$ from above by

$$
\begin{aligned}
\|\mathbf{w}^{(t)}\|^2 =& \langle \mathbf{w}^{(t-1)} + \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i)\,,\, \mathbf{w}^{(t-1)} + \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i)\rangle \\
=& \|\mathbf{w}^{(t-1)}\|^2 + \underbrace{2\langle \mathbf{w}^{(t-1)}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i)\rangle}_{\leq 0} + \|\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x} \\
\leq& \|\mathbf{w}^{(t-1)}\|^2 + \|\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}}_i)\|^2 \\
\leq& \|\mathbf{w}^{(t-1)}\|^2 + r^2.
\end{aligned}
$$

Thus, by induction we have $\|\mathbf{w}^{(t)}\|^2 \leq \boxed{tr^2}$. Putting everything together gives us

$$
\begin{aligned}
\boxed{t\bar{\gamma}} \leq& \langle \mathbf{w}^*, \mathbf{w}^{(t)}\rangle \\
\leq& \|\mathbf{w}^*\|\, \|\mathbf{w}^{(t)}\| \\
\leq& \|\mathbf{w}^*\|\, \sqrt{t}r.
\end{aligned}
$$

Cauchy Schwartz inequality!

Solving for t implies the upper bound

$$
t \leq \left(\frac{r}{\bar{\gamma}}\right)^2 \|\mathbf{w}^{(*)}\|^2.
$$

$w^0 \leftarrow \vec{0}$

$\vdots$

$w^t \leftarrow w^{t-1} + \Phi(x_n, y_n) - \Phi(x_n, \bar{y})$

$\Rightarrow$ introduce counters: $\alpha_n(\bar{y}) \leftarrow \alpha_n(\bar{y}) + 1$

init: $\alpha_n(\bar{y}) \leftarrow 0 \quad \forall n, \bar{y}$

$$w = \sum_{n=1}^{N} \sum_{\bar{y}} \alpha_n(\bar{y}) \left(\Phi(x_n, y_n) - \Phi(x_n, \bar{y}) \right)$$

negative examples computed during training

$$f(x, y) = w^T \Phi(x, y) = \sum_{n=1}^{N} \sum_{\bar{y}} \alpha_n(\bar{y}) \left(\Phi(x_n, y_n) - \Phi(x_n, \bar{y}) \right)^T \Phi(x, y)$$

structured prediction

$\Phi(x, y)$

$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$

$\downarrow$ encodes

$$\Phi(x, y) = \begin{pmatrix} \Sigma \phi_1(x, y) \\ \Sigma \phi_2(x, y) \end{pmatrix}$$

Φ_2

Φ_3

Φ_4

$\vdots$

Φ_z

$$P(A,B \mid B,D) = \frac{1}{Z}\,\psi_{c_1}(A,D)\,\psi_{c_2}(A,B,C)$$

$$= \frac{1}{Z}\exp\left\{\sum_{d=1}^{D_1} w_1^T \phi_d^1(A,D)\right\}\exp\left\{\sum_{d=1}^{D_2} w_2 \phi_d^2(A,B,C)\right\}$$

$$= \frac{1}{Z}\exp\left\{w_1^T \Phi_1(A,D)\right\}\exp\left\{w_2^T \Phi_2(A,B,C)\right\}$$

$$= \frac{1}{Z}\exp\left\{\underbrace{\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}}_{w}^T \underbrace{\begin{pmatrix} \Phi_1(A,D) \\ \Phi_2(A,B,C) \end{pmatrix}}_{\Phi(x,y)}\right\}$$

$$P(A,B \mid B,D) = \frac{1}{Z}\prod_{k=1}^{2}\psi_{c_1}(A,D_k)\,\psi_{c_2}(A,B,C)$$

$$= \frac{1}{Z}\prod_{k=1}^{2}\exp\left\{\sum_{d=1}^{D_1} w_1^T \phi_d^1(A,D_k)\right\}\exp\left\{\sum_{d=1}^{D_2} w_2 \phi_d^2(A,B,C)\right\}$$

$$= \frac{1}{Z}\exp\left\{\sum_{k=1}^{2} w_1^T \Phi_1(A,D_k)\right\}\exp\left\{w_2^T \Phi_2(A,B,C)\right\}$$

$$= \frac{1}{Z}\exp\left\{\underbrace{\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}}_{w}^T \underbrace{\begin{pmatrix} \sum_{k=1}^{2}\Phi_1(A,D_k) \\ \Phi_2(A,B,C) \end{pmatrix}}_{\Phi(x,y)}\right\}$$

len = length(word)

[len-3 : len] == "ing"

A, B, C, D ....

ϕ (...)

Towards Dual Perceptrons

implementation:

$$\sum_{n=0}^{N} \sum_{\bar{y}} \alpha_n(\bar{y})$$

- Observation: $\mathbf{w}^{(0)} \leftarrow \mathbf{0}$

- i-th example violates constraint:
  - Update: $\mathbf{w}^{(t+1)} = \mathbf{w}^{(i)} + \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \max_{\bar{\mathbf{y}}} \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle$

- Idea: rembember how many times the pair $(\mathbf{x}_i, \bar{\mathbf{y}})$ is used for an upate!
  - Variable $\alpha_i(\bar{\mathbf{y}})$ acts as a counter
  - Initialize: $\alpha_i(\bar{\mathbf{y}}) \leftarrow 0$
  - Update: $\alpha_i(\bar{\mathbf{y}}) \leftarrow \alpha_i(\bar{\mathbf{y}}) + 1$

- The α are bound to violated constraints!

Dual Representation

- Dual parameters
  - $\alpha_i(\bar{\mathbf{y}})$ is proportional to the importance of $\Phi(\mathbf{x}_i, \mathbf{y}_i)\rangle - \Phi(\mathbf{x}_i, \bar{\mathbf{y}})$

- Recall: α counted the number of updates for $\mathbf{w}$
  - we can thus write:

$$\mathbf{w} = \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\Phi(\mathbf{x}_i, \mathbf{y}_i)\rangle - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \right)$$

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

Dual Decision Function

$$\mathbf{w} = \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \right)$$

- Plug dual representation of **w** into decision function:

$$\begin{aligned}
f(\mathbf{x}', \mathbf{y}') &= \langle \mathbf{w}, \Phi(\mathbf{x}', \mathbf{y}') \rangle \\
&= \langle \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \right), \Phi(\mathbf{x}', \mathbf{y}') \rangle \\
&= \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\langle \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}), \Phi(\mathbf{x}', \mathbf{y}') \rangle \right) \\
&= \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\langle \Phi(\mathbf{x}_i, \mathbf{y}_i), \Phi(\mathbf{x}', \mathbf{y}') \rangle - \langle \Phi(\mathbf{x}_i, \bar{\mathbf{y}}), \Phi(\mathbf{x}', \mathbf{y}') \rangle \right.
\end{aligned}$$

Kernels and the Dual Perceptron

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

$$f_D(\mathbf{x}', \mathbf{y}') = \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(K(\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}', \mathbf{y}') - K(\mathbf{x}_i, \bar{\mathbf{y}}, \mathbf{x}', \mathbf{y}') \right)$$

- Example (sequences, indicator functions)

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

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

```
loop r = 1, ..., r_max
    loop i = 1, ..., n
        Compute ŷ = argmax_ȳ f_D(x_i, ȳ)
        If y_i ≠ ŷ
            Increment α_i(ŷ) ← α_i(ŷ) + 1
        End (if)
    End loop (i)
End loop (r)
```

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

What about the Argmax?

- For dual perceptron it's easy!
- Decompose $f(\mathbf{x}, \mathbf{y}) = f_1(\mathbf{x}, \mathbf{y}) + f_2(\mathbf{x}, \mathbf{y})$ with

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

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

- and

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

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

- (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$(P(\mathbf{y}|\mathbf{x}))$ and $f(\mathbf{x}, \mathbf{y})$

Named Entity Recognition

Figure 1. Test error of NER task over a window of size 3 using 5-fold cross validation.

Example: *Como (O) contrapartida (O) Deutsche (C-B) Telekom (C-I) vender (O) al (O) consorcio (O) frances (O) su (O) participacion (O) del (O) por (O) ciento (O) en (O) el (O) empresa (O) mixta (O) britanica (O) MetroHoldings (C-B).*
(see Altun et al. (2003))

Natural Language Parsing

Depth	1	2	3	4	5	6
Score	73 ± 1	79 ± 1	80 ± 1	79 ± 1	79 ± 1	78 ± 0.01
Improvement	-1 ± 4	20 ± 6	23 ± 3	21 ± 4	19 ± 4	18 ± 3

Table 1: *Score* shows how the parse score varies with the maximum depth of sub-tree considered by the perceptron. *Improvement* is the relative reduction in error in comparison to the PCFG, which scored 74%. The numbers reported are the mean and standard deviation over the 10 development sets.

(see Collins&Duffy, 2002)

BioCreative

- Detection of gene and protein names in biomedical abstracts

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

Alternative HMM Structures

- Input-Output HMMs: Turn HMMs into a discriminative model by reversing the dependency between the states and the observations and introducing extra output variables

Alternative HMM Structures

- Autoregressive HMMs: Add adding explicit dependencies between the observations to improve long-range context modelling

Alternative HMM Structures

- Factorial HMMs: Add extra chains of hidden states, thereby moving from a single-valued to a distributed architecture and increasing the memory capacity of HMMs: $\mathcal{O}(logN) \rightarrow \mathcal{O}(N)$

Recall: Conditional MRF

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{T} \psi^{obs}(X_i, Y_i) \prod_{i=2}^{T} \psi^{trans}(Y_{i-1}, Y_i) \propto \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$$

Recall: Generalized Linear Models

- $\mathbf{x} =$ Bob chases the dog
- We want

$$[N, V, D, N] = \underset{\bar{\mathbf{y}}}{\text{argmax}} \langle \mathbf{w}, \Phi(\mathbf{x}, \bar{\mathbf{y}}) \rangle$$

- Equivalent representation:

$$\begin{aligned}
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [D, D, D, D]) \rangle \\
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [D, D, D, N]) \rangle \\
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [D, D, N, D]) \rangle \\
\vdots \quad &> \quad \vdots \\
\langle \mathbf{w}, \Phi(\mathbf{x}, [N, V, D, N]) \rangle &> \langle \mathbf{w}, \Phi(\mathbf{x}, [V, V, V, V]) \rangle
\end{aligned}$$

Recall: Primal/Dual Perceptron

- Primal perceptron:
  - Decision function: $f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$
  - Update rule: $\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \hat{\mathbf{y}})$

- Dual perceptron:
  - Use relation: $\mathbf{w} = \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \right)$
- Dual decision function:

$$f(\mathbf{x}', \mathbf{y}') = \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \left(\langle \Phi(\mathbf{x}_i, \mathbf{y}_i), \Phi(\mathbf{x}', \mathbf{y}') \rangle - \langle \Phi(\mathbf{x}_i, \bar{\mathbf{y}}), \Phi(\mathbf{x}', \mathbf{y}') \rangle \right.$$

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

Primal/Dual Algorithm

```
loop r = 1, ..., r_max
    loop i = 1, ..., n
        Compute prediction ŷ
        If y_i ≠ ŷ
            Update w (primal) or α_i(ŷ) (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 $f(\mathbf{x}, \mathbf{y})$
  - ???

- How can we compute $\text{argmax}_{\bar{\mathbf{y}}}\, f(\mathbf{x}, \bar{\mathbf{y}})$?
- Answer: use log-Viterbi = max-sum algorithm!

Altun et al. 2003:

- Dual perceptron:
- Decompose $f(\mathbf{x}, \mathbf{y}) = f_1(\mathbf{x}, \mathbf{y}) + f_2(\mathbf{x}, \mathbf{y})$ with

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

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

- and

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

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

Recall: Viterbi Algorithm

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

- 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 $t = 2, \ldots, T$:
  - $\delta_t(\sigma) = \left[\max_\tau \delta_{t-1}(\tau) + \log P(y_t = \sigma | y_{t-1} = \tau)\right] + \log P(x_t | y_t = \sigma)$
  - $\psi_t(\sigma) = \left[\operatorname{argmax}_\tau \delta_{t-1}(\tau) + \log P(y_t = \sigma | y_{t-1} = \tau)\right] + \log P(x_t | y_t = \sigma)$
- termination: $y_T^* = \operatorname{argmax}_\sigma \delta_T(\sigma)$
- loop $t = T, \ldots, 2$
  - $y_{t-1}^* = \psi_t(y_t^*)$

Scoring Function $f(\mathbf{x}, \mathbf{y})$

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

- Observation probabilities:

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

- Transition probabilities:

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

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

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

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

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\bar{\mathbf{y}} \in \Sigma^T} f(\mathbf{x}, \bar{\mathbf{y}})$$

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

Proof:

- The model f has the form

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

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

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

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

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

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

Proof Contd.

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

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

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

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

...

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

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

where f_a weights the occurences of neighboring labels in $\mathbf{y}$ 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

$$\delta_t(\sigma) = \max_{y_1,\ldots,y_{t-1}} f(\mathbf{x}, y_1, \ldots, y_{t-1}, y_t = \sigma), \tag{4}$$

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

$$\delta_{t+1}(\sigma) = \max_{\tau \in \Sigma} \left[\delta_t(\tau) + a(\tau, \sigma)\right] + b(\sigma, x_{t+1}) \tag{5}$$

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

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

The Inductive Step Contd.

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

Thus, the top scoring sequence has the score

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

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

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

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

$$y_T^\star = \operatorname*{argmax}_{\sigma \in \Sigma} \delta_T(\sigma)$$

$$y_t^\star = \varphi_{t+1}(y_{t+1}^\star) \quad \text{for} \quad t = T-1, \ldots, 1.$$

Conclusion

Given the transition matrix $[\mathbf{A}]_{\sigma,\tau} = a(\sigma, \tau)$ and the observation matrix $[\mathbf{B_x}]_{\sigma,t} = b(\sigma, x_t)$ for input **x**, 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 $f(\mathbf{x}, \mathbf{y})$ 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:

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

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

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

Optimization Problem

$$\max_{\bar{\gamma},\mathbf{w}} \quad \frac{\bar{\gamma}}{||\mathbf{w}||}$$

$$\text{s.t.} \quad \forall_{i=1}^{n}, \forall_{\bar{\mathbf{y}}\neq\mathbf{y}_i} : \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i)\rangle - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}})\rangle \geq \bar{\gamma}$$

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

Structural Hard-margin SVM

$$\min_{\mathbf{w}} \quad \frac{1}{2}||\mathbf{w}||^2$$

$$\text{s.t.} \quad \forall_{i=1}^{n}, \forall_{\bar{\mathbf{y}} \neq \mathbf{y}_i} : \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \geq 1$$

- 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

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

$$\text{s.t.} \quad \forall_{i=1}^{n}, \forall_{\bar{\mathbf{y}} \neq \mathbf{y}_i} : \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \geq 1 - \xi_i$$

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

- Sum of slacks upper bounds 0/1 loss
- Now: maximize margin between true $\mathbf{y}_i$ and best runner-up $\bar{\mathbf{y}}$
- Alternative formulation:
  - slack $\xi_{i\bar{\mathbf{y}}}$ are bound to constraint $\langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle$
  - computationally demanding

Hinge-loss

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

Examplary Loss Functions

- 0/1 loss: $\Delta(\mathbf{y}, \bar{\mathbf{y}}) = [[\mathbf{y} == \bar{\mathbf{y}}]]$
- Hamming loss for sequences

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

  - Property: decomposes across the cliques!

Risk Minimization

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

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta\big(\mathbf{y}, \operatorname*{argmax}_{\bar{\mathbf{y}}} f(\mathbf{x}, \bar{\mathbf{y}})\big) dP(\mathbf{x}, \mathbf{y})$$

- In general, we don't know $P(\mathbf{x}, \mathbf{y})$
  - Remedy: Use training sample instead!

- Minimize the empirical risk

$$\hat{R}(f) = \sum_{i=1}^{n} \Delta\big(\mathbf{y}_i, \operatorname*{argmax}_{\bar{\mathbf{y}}} f(\mathbf{x}_i, \bar{\mathbf{y}})\big)$$

Idea

- SVMs minimize the (regularized) empirical risk:

$$\hat{R}(f) = \sum_{i=1}^{n} \Delta\big(\mathbf{y}_i, \operatorname*{argmax}_{\bar{\mathbf{y}}} f(\mathbf{x}_i, \bar{\mathbf{y}})\big)$$

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

Margin-rescaling

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

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

$$\text{s.t.} \quad \forall_{i=1}^{n}, \forall_{\bar{\mathbf{y}} \neq \mathbf{y}_i} : \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \geq \Delta(\mathbf{y}_i, \bar{\mathbf{y}}) - \xi_i$$

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

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

$$\operatorname*{argmax}_{\bar{\mathbf{y}}} \left(\underbrace{\Delta(\mathbf{y}_i, \bar{\mathbf{y}}) - \left(\langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \right)}_{\xi_i} \right)$$

Slack-rescaling

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

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

$$\text{s.t.} \quad \forall_{i=1}^{n}, \forall_{\bar{\mathbf{y}} \neq \mathbf{y}_i} : \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{y}_i, \bar{\mathbf{y}})}$$

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

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

$$\operatorname*{argmax}_{\bar{\mathbf{y}}} \left(\underbrace{1 - \Delta(\mathbf{y}_i, \bar{\mathbf{y}}) \times \left(\langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \rangle \right)}_{\xi_i} \right)$$

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 $\alpha_i(\bar{\mathbf{y}})$ and μ_i

- Build Lagrangian:

$$
\begin{aligned}
L =& \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\
& - \sum_{i=1}^{n}\sum_{\bar{\mathbf{y}}\neq\mathbf{y}_i}\alpha_i(\bar{\mathbf{y}})\langle\mathbf{w},\Phi(\mathbf{x}_i,\mathbf{y}_i)\rangle - \langle\mathbf{w},\Phi(\mathbf{x}_i,\bar{\mathbf{y}})\rangle - 1 + \xi_i \\
& - \sum_{i=1}^{n}\beta_i\xi_i
\end{aligned}
$$

- Minimum of Lagrangian is a saddle-point
  - max wrt α, μ, min wrt $\mathbf{w}, \xi$

Partial Derivatives: ξ_i

- Compute partial derivatives wrt ξ:

$$\frac{\partial L}{\partial \xi_i} = C - \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) - \beta_i \quad \stackrel{!}{=} \quad 0 \tag{6}$$

- Using the non-negativity of α and β yields

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

Partial Derivatives: $\mathbf{w}$

- Compute partial derivatives wrt $\mathbf{w}$:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \Big(\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \Big) \stackrel{!}{=} 0.$$

- We obtain:

$$\mathbf{w} = \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \Big(\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) \Big)$$

- Recall: dual perceptron!
  - Definition of $\mathbf{w}$ is equivalent
  - α's act like counters

Putting all together...

- Dual SVM for structured output variables:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) - \frac{1}{2} \sum_{i,j=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \sum_{\bar{\bar{\mathbf{y}}} \neq \mathbf{y}_j} \Big(\alpha_i(\bar{\mathbf{y}})\, \alpha_j(\bar{\bar{\mathbf{y}}}) \, \langle \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \bar{\mathbf{y}}), \Phi(\mathbf{x}_j, \mathbf{y}_j) - \Phi(\mathbf{x}_j, \bar{\bar{\mathbf{y}}}) \rangle \Big)$$

  - subject to the constraints

$$\forall_{i=1}^{n} \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_i(\bar{\mathbf{y}}) \leq C$$

$$\forall_{i=1}^{n} \, \forall i : \quad \alpha_i(\bar{\mathbf{y}}) \geq 0.$$

Optimization

- Dual SVM is a QP (quadratic program)
- Sketch of algorithm:

```
Loop
    Loop i = 1, ..., n
        Compute most strongly violated constraint ȳ
        If constraint causes more slack than before (ξ_i)
            Add constraint Φ(x_i, y_i) − Φ(x_i, ȳ) to working set,
            Initialize α_i(ȳ)
    Solve QP
Until convergence
```

(For details see Tsochantaridis et al. JMLR,2005)

Comments

- Working set contains active constraints
  - Only a few of the constraints will be active
  - remove constraints if $\alpha_i(\bar{\mathbf{y}}) = 0$ after optimization

- Convergence in polynomial time
  - if computation of argmax takes at most polynomial time
  - proof: see Tsochantaridis et al. (2005)

- Implementation on the web:
  - SVM^{struct} by Thorsten Joachims (extension of SVM^{light})
  - `http://svmlight.joachims.org/svm_struct.html`

Named Entity Recognition

Figure 1. Test error of NER task over a window of size 3 using 5-fold cross validation.

Example: *Como (O) contrapartida (O) Deutsche (C-B) Telekom (C-I) vender (O) al (O) consorcio (O) franc□ s (O) su (O) participacion (O) del (O) por (O) ciento (O) en (O) el (O) empresa (O) mixta (O) brit□ nica (O) MetroHoldings (C-B).*
(see Altun et al. (2003))

CRF: Special Case

- Sequential approach =
  - a chain of multi-class classification problems
  - linked via label-label transitions

Multi-class Classification

- Let $\psi(\mathbf{x})$ the feature vector of observation $\mathbf{x}$
- Let $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ be the set of classes
- Then, the joint feature map is given by

$$\Phi(\mathbf{x}, y) = ([[y = \sigma_1]]\psi(\mathbf{x})^\mathsf{T}, \ldots, [[y = \sigma_k]]\psi(\mathbf{x})^\mathsf{T})^\mathsf{T}.$$

  - dimension of Φ is $dim(\Phi) = |\Sigma| dim(\psi)$
  - see also Crammer & Singer (2001), Weston & Watkins (1998)

Outlook

- What happens if only observations but no labels are observed?
  - i.e., training set consists of n unlabeled observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$