Classification

- y : random variable for prediction (output)
- x : random variable for observation (input)
- Training Data = Collection of (x, y) pairs
- Machine Learning = Given the training data, learn a mapping function $f(x) = y$ that can map input variables to output variables

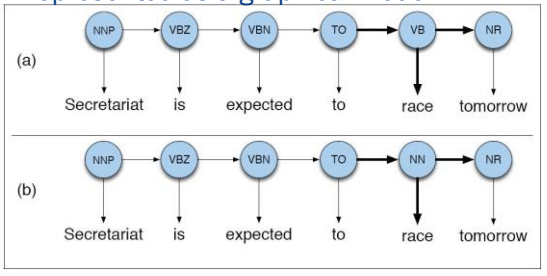
- Binary classification
- Multiclass classification

Sequence Tagging

- y : A sequence of random variables for prediction (output)
- x : A sequence of random variables for observation (input)
- Training Data = Collection of (x, y) pairs
- Machine Learning = Given the training data, learn a mapping function $f(x) = y$ that can map input variables to output variables

- Binary classification
- Multiclass classification

Hidden Markov Model (HMM) represented as a graphical model

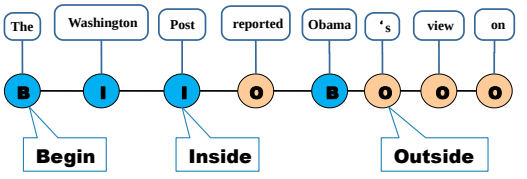


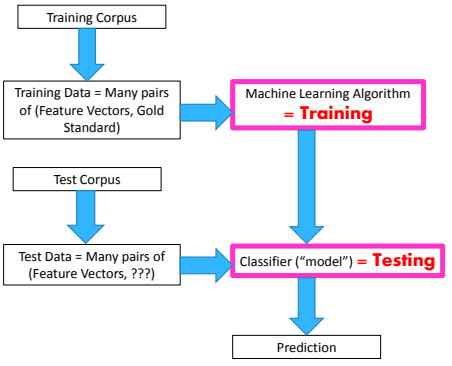
- Note that in this representation, the number of nodes (states) = the length of the word sequence.

Sequence Tagging

- Prediction using **BIO** tagging

<The Washington Post> reported <Obama>'s view on the oil crisis.





- Classifier
  - Maximum Entropy (MaxEnt)
  - Naïve Bayes
- Sequence Tagger
  - Hidden Markov Models (HMMs)
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)

Maximum Entropy (MaxEnt) Models

Maximum Entropy (MaxEnt) Models

- Also known as "**Log-linear**" Models (*linear if you take log*)

$$P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

- The feature vector representation may include redundant and overlapping features

(slide modified from Dan Klein's)

Training Maximum Entropy (MaxEnt)

- *Maximizing the likelihood* of the training data incidentally *maximizes the entropy* (hence "maximum entropy")

$$P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{y}'))}$$

Make positive / Normalize

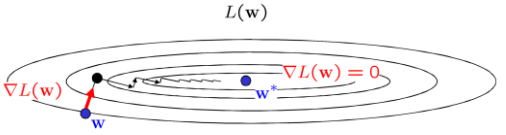
- Maximize the (log) conditional likelihood of training data

$$L(\mathbf{w}) = \log \prod_i P(\mathbf{y}^i|\mathbf{x}^i,\mathbf{w}) = \sum_i \log\left(\frac{\exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i))}{\sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))}\right)$$

$$= \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))\right)$$

(slide modified from Dan Klein's)

Convex Optimization for Training



- The likelihood function is convex. (can get global optimum)
- Many optimization algorithms/software available.
  - Gradient ascent (descent), Conjugate Gradient, L-BFGS, etc
- All we need are:
  (1) evaluate the function at current 'w'
  (2) evaluate its derivative at current 'w'

(slide modified from Dan Klein's)

Training Maximum Entropy

$$L(\mathbf{w}) = \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))\right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n\right)$$

Total count of feature n in correct candidates

Expected count of feature n in predicted candidates

(slide modified from Dan Klein's)

Training with Regularization

$$L(\mathbf{w}) = -k||\mathbf{w}||^2 + \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))\right)$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_n} = -2k\mathbf{w}_n + \sum_i \left(\mathbf{f}_i(\mathbf{y}^i)_n - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i)\mathbf{f}_i(\mathbf{y})_n\right)$$

Big weights are bad

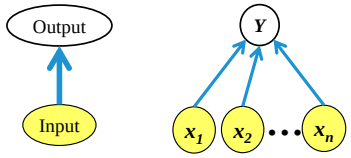
Total count of feature n in correct candidates

Expected count of feature n in predicted candidates

(slide modified from Dan Klein's)

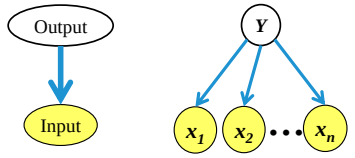
Graphical Representation of MaxEnt

$$P(y|x,w) = \frac{\exp(w^\top f(y))}{\sum_{y'} \exp(w^\top f(y'))}$$

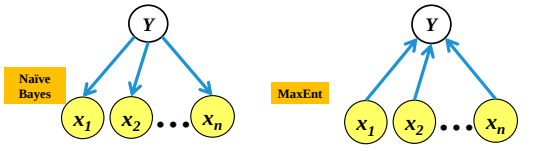


Graphical Representation of Naïve Bayes

$$P(X \mid Y) = \prod_{j=1} P(x_j \mid Y)$$



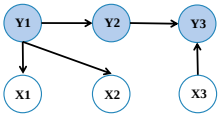
- Consult AI text book for more details



Naïve Bayes Classifier	Maximum Entropy Classifier
"Generative" models ➔ **p(input | output)** ➔ For instance, for text categorization, P(words | category) ➔ Waste energy on generating input (which we don't need to generate during test)	"Discriminative" models ➔ **p(output | input)** ➔ For instance, for text categorization, P(category | words) ➔ Focusing only on predicting the output
➔ **Independent assumption among input variables:** Given the category, each word is generated independently from other words **(too strong assumption in reality!)** ➔ Cannot incorporate arbitrary/redundant/overlapping features	➔ By conditioning on the entire input, we don't need to worry about the independent assumption among input variables ➔ **Can incorporate arbitrary features** ➔ **Can handle redundant and overlapping features**

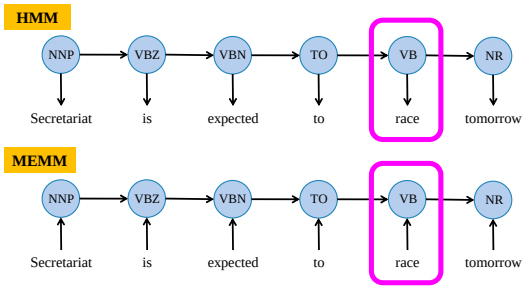
Sequence Tagging with HMM / MEMM / CRF

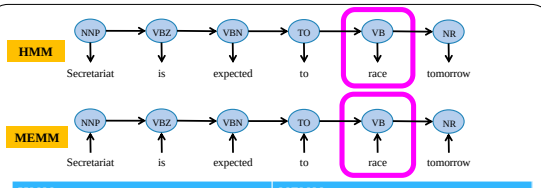
Graphical Model Basics



- Conditional probability for each node
  - e.g. p(Y3 | Y2, X3) for Y3
  - e.g. p(X3) for X3
- Conditional independence
  - e.g. p(Y3 | Y2, X3) = p(Y3 | Y1,Y2, X1, X2, X3)
- Joint probability of the entire graph
  = product of conditional probability of each node

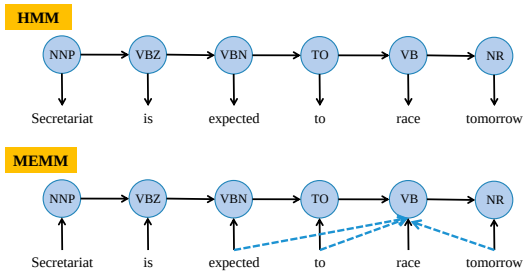
HMM v.s. MEMM (Maximum Entropy Markov Models)



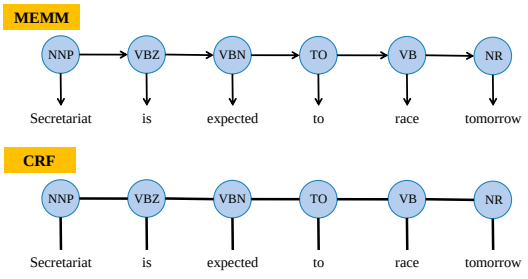


HMM	MEMM
"Generative" models ➔ joint probability **p(words, tags)** ➔"generate" input (in addition to tags) ➔ but we need to predict tags, not words!	"Discriminative" or "Conditional" models ➔ conditional probability **p(tags | words)** ➔"condition" on input ➔ Focusing only on predicting tags
Probability of each slice = emission * transition = p(word_i | tag_i) * p(tag_i | tag_i-1) = ➔ Cannot incorporate long distance features	Probability of each slice = p(tag_i | tag_i-1, word_i) or p(tag_i | tag_i-1, all words) ➔ Can incorporate long distance features

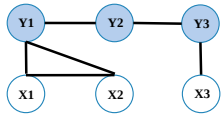
HMM v.s. MEMM



MEMM v.s. CRF (Conditional Random Fields)

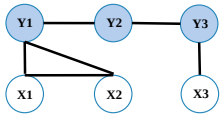


Undirected Graphical Model Basics



- Conditional independence
  - e.g. p(Y3 | all other nodes) = p(Y3 | Y3' neighbor)
- No conditional probability for each node
- Instead, "***potential function***" for each ***clique***
  - e.g. ϕ (X1, X2,Y1) or ϕ (Y1,Y2)
- Typically, log-linear potential functions
  - ➔ ϕ (Y1,Y2) = exp Σ_i w_i f_i(Y1,Y2)

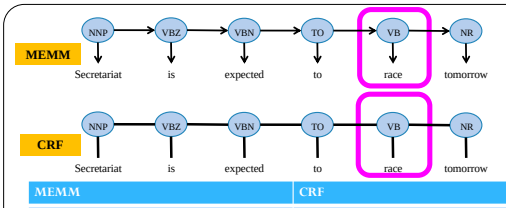
Undirected Graphical Model Basics



- Joint probability of the entire graph

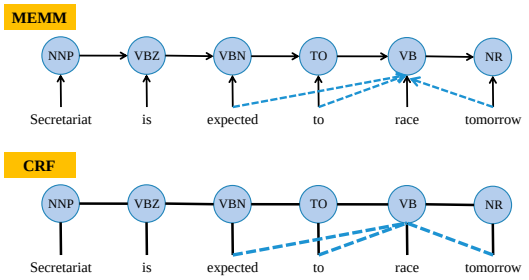
$$P(\vec{Y}) = \frac{1}{Z} \prod_{\text{clique } C} \varphi(\vec{Y}_C)$$

$$Z = \sum_{\vec{Y}} \prod_{\text{clique } C} \varphi(\vec{Y}_C)$$

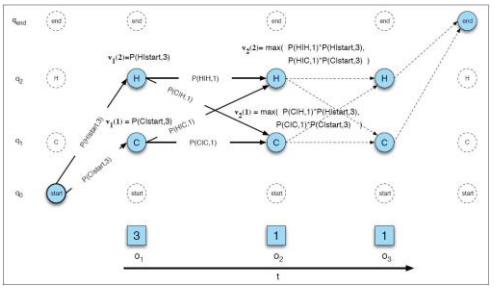


MEMM	CRF
Directed graphical model	Undirected graphical model
"Discriminative" or "Conditional" models ➔ conditional probability **p(tags | words)**	
Probability is defined for each slice = P (tag_i | tag_i-1, word_i) or p (tag_i | tag_i-1, all words)	Instead of probability, ***potential (energy function)*** is defined for each slide = ϕ (tag_i, tag_i-1) * ϕ (tag_i, word_i) or ϕ(tag_i, tag_i-1, all words) * ϕ (tag_i, all words)
➔ Can incorporate long distance features	

MEMM v.s. CRF



Inference (Viterbi)



Objective function for training

Given the training data $D = \{x^{(i)}, y^{(i)}\}_{j=1}^{N}$

and $p(y \mid x) = \frac{1}{Z(x)} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(y, x)$

Objective function :

conditional likelihood $L(\boldsymbol{\lambda}) = L(\boldsymbol{\lambda} \mid D) = P(D \mid \boldsymbol{\lambda}) = \prod_j p(y^{(j)} \mid x^{(j)})$

equiv. to optimize $\boldsymbol{l}(\boldsymbol{\lambda}) = \log L(\boldsymbol{\lambda}) = \sum_j \log p(y^{(j)} \mid x^{(j)})$

$$\boldsymbol{l}(\boldsymbol{\lambda}) = \sum_j \log p(y^{(j)} \mid x^{(j)}) = \sum_j \log \frac{1}{Z(x)} \exp \boldsymbol{\lambda} \bullet \mathbf{F}(y^{(j)}, x^{(j)})$$

$$= \sum_j \boldsymbol{\lambda} \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log Z(x^{(j)})$$

$$= \sum_j \left(\boldsymbol{\lambda} \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log \sum_y \exp \boldsymbol{\lambda} \bullet \mathbf{F}(y^{(j)}, x^{(j)}) \right)$$

nasty!

CRFs Software:

- Mallet (http://mallet.cs.umass.edu/),
- CRF++ (http://crfpp.sourceforge.net/),
- CRF (http://crf.sourceforge.net/)