[Day-1] Introduction to Statistical Natural Language Processing

(Part VI: Advanced Topics)

Keh-Yih Su and Jing-Shin Chang
kysu@bdc.com.tw, shin@nlp.csie.ncnu.edu.tw

(2002/8/17)

Behavior Design Corporation (BDC)
No. 5, 2F, Industrial East Road IV, Science-Based Industrial Park
Hsinchu, Taiwan 30077, R.O.C.

Department of Computer Science and Information Engineering
National Chi-Nan University
Puli, Nantou, Taiwan 545, R.O.C.

Day-1: Introduction to Statistical Natural Language Processing (mainly on Supervised Learning)

- Part I: Introduction (1)
  - Problems and Characteristics of Natural Language Processing
- Part II: Introduction (2)
  - What, When and Why Statistical Approach
- Part III: Basic Concepts and Background
  - Feature Space, Probability, Estimator, Stochastic Process, Data Set Classification, and Performance Measure
- Part IV: Typical Applications
  - Word Segmentation, Tagging, Selecting Parse Tree, Aligning Bilingual Corpus
- Part V: Techniques for Improving Performance
  - Smoothing, Class-Based Model, Adaptive Learning, Tips for Checking
- **Part VI: Advanced Topics: SVM, ME**
  - **Support Vector Machine, Maximum Entropy Models**
- Appendix: Related Techniques
  - Parameter Estimation, Fractional Factorial Experiment Design, Decision Tree

Part VI: Advanced Topics

- Support Vector Machine (SVM)
  - What is a Support Vector Machine (SVM)
  - How it operate under linear separable cases
  - How to deal with non-linear separable cases
  - Non-separable situations
  - Multi-class classification
  - A Part-of-Speech Tagging Example
  - General Comments to SVM

Part VI: Advanced Topics (Cont.)

- Maximum Entropy
  - A way to combine different knowledge sources under probabilistic framework
  - Training via Generalized Iterative Scaling
  - A Part-of-Speech Tagging Example
  - A Parse-Tree Selection Example
  - General Comments to Maximum Entropy
- How to embrace those new emerging Machine Learning Techniques

Support Vector Machines

What is a Support Vector Machine (SVM)

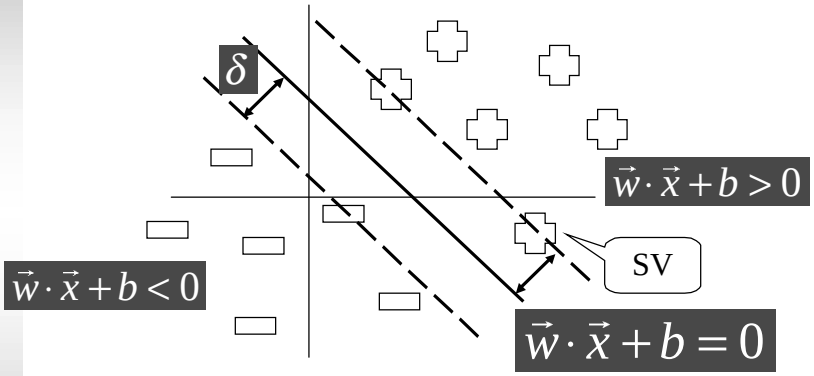
- SVM is a linear binary classifier:
  - Use a “hyperplane”, with the largest “margin”, to separate the data in the feature space into two classes. X: Feature Vector.

$$y_i(\vec{x}_i; w, b) = \begin{cases} -1 & \text{if } \vec{w} \cdot \vec{x}_i + b < 0 \\ +1 & \text{if } \vec{w} \cdot \vec{x}_i + b > 0 \end{cases}$$

  - Hyperplanes:
    - 2-D: a straight line
    - 3-D: a plane
    - N-D: a hyper-plane

$$\vec{w} \cdot \vec{x} + b = 0$$

Hyperplane (2D case)



Linear Separable Cases

- For linearly separable training set, finding the maximum margin classifier from the training set
- Criteria:
  - Discrimination: Minimize the number of training set errors
    - Many possible solutions in linearly separable cases
  - Generalization/robustness: Maximum margin
    - Choose the one that maximizes the margin

NLP Examples

- Text Classification [Joachims, 1999]
  - Feature vector X is a list of word-stems (e.g., baseball, space, car, etc.) frequencies extracted from the given document
  - Dimensionality about 30,000 has been reported
- Part-of-Speech Tagging [Murata et al. NLPNN 2001]
  - X might include: (1) POS and order information, (2) Word information
  - POS and order information: current word, three previous words, and three sub-sequent words. 782 binary elements reported for Thai
  - Word information: current word, three previous words, and three sub-sequent words. 15,763 binary elements reported for Thai
- Chunk Identification [Kudoh and Matsumoto CoNLL-2000], Parse Tree Selection, etc.
  - All with high dimensionality

High Dimensional Feature Vector is Adopted

- In NLP applications, feature vector of SVM usually is a large array of binary value (even hundred thousands, sometimes)
  - In statistical model, the dimensionality of feature vector is small, and each feature element can have a large number of various discrete values (e.g., tagging tri-gram model: $P(W_i \mid C_i)$)
- Allow to incorporate more word context
  - Feature selection is not critical
  - Don't need to worry about dependency relationship between features
- Increasing dimensionality increases discriminating information
  - Might convert a non-separable problem into a separable problem

High Dimensional Feature Vector is Adopted (Cont.)

- Capable to convert a non-linear separable problem into a linear separable problem, if it is possible
  - Replace a complicated classifier (non-linear) in a lower dimensional feature space by a simple classifier (linear) in a higher dimensional feature space
- The only Problems:
  - Possible over-fitting (could be avoided by enlarging margin)
  - High computation cost

Enlarge Margin

- Margin: the minimum distance between hyperplane and training data

$$\delta = \min_i dist(\vec{x}_i, h: \vec{w}\cdot\vec{x}+b=0) = \min_i \left|\frac{\vec{w}\cdot\vec{x}_i+b}{\sqrt{\vec{w}\cdot\vec{w}}}\right|$$

$$= \min_i y_i\left(\frac{\vec{w}\cdot\vec{x}_i+b}{\sqrt{\vec{w}\cdot\vec{w}}}\right) \quad \left(|y_i|=1 \,\&\&\, \mathrm{sgn}(y_i)=\mathrm{sgn}(|\bullet|)\right)$$

- Which can be shown to be

$$\delta = \frac{1}{\sqrt{\vec{w}\cdot\vec{w}}}$$

Maximum margin classifier (1)

- To minimize training error: select (**w**, b) such that

$$y_i = \begin{cases} -1 & \Rightarrow \vec{w}\cdot\vec{x}_i + b < 0 \\ +1 & \Rightarrow \vec{w}\cdot\vec{x}_i + b > 0 \end{cases} \Rightarrow y_i\left(\vec{w}\cdot\vec{x}_i + b\right) > 0, \forall i$$

- To maximize margin: select (**w**, b) such that δ is maximum, where

$$\delta = \min_i \left| \frac{\vec{w}\cdot\vec{x}_i + b}{\sqrt{\vec{w}\cdot\vec{w}}} \right| = \frac{1}{\sqrt{\vec{w}\cdot\vec{w}}}$$

Maximum margin classifier (2)

- Primal optimization problem

$$\text{minimize} \quad J\left(\vec{w}, b\right) \equiv \frac{1}{2}\vec{w}\cdot\vec{w}, \quad \text{with } y_i\left(\vec{w}\cdot\vec{x}_i + b\right) \geq 1, \forall i$$

- Dual optimization problem: by introducing a Lagrange multiplier α_i for each constraint, leading to the following dual optimization problem:

$$\text{maximize} \quad L\left(\vec{\alpha}\right) \equiv \left(\sum_{i=1,n} \alpha_i\right) - \frac{1}{2}\sum_{i=1,n}\sum_{j=1,n} \alpha_i\alpha_j y_i y_j \left(\vec{x}_i \cdot \vec{x}_j\right)$$

$$\text{subject to} \quad \sum_{i=1,n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i$$

Maximum margin classifier (3)

- Solution:

$$\vec{w}^* = \sum_{i=1,n} \alpha_i^* y_i \vec{x}_i$$

$$b^* = -\frac{1}{2}\left(\min_{i:y_i=+1} (\vec{w}^* \cdot \vec{x}_i) + \max_{i:y_i=-1} (\vec{w}^* \cdot \vec{x}_i) \right)$$

- Classifier:

$$\hat{y} = f(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x} + b) = \text{sgn}\left(\sum_{i=1,n} \alpha_i^* y_i (\vec{x}_i \cdot \vec{x}) + b \right)$$

Characteristics

- Optimal <**w**, b> has a unique solution
- Optimal **w** is a linear combination of training examples, each example being weighted by one α_i
  - $\alpha_i = 0$: do not affect optimal hyperplane (inactive constraint)
  - α_i =/= 0: contributes to optimal hyperplane, and
  - (xi , yi) whose associated α_i =/= 0 : exactly on the margin
    - Called a “Support Vector” (SV)

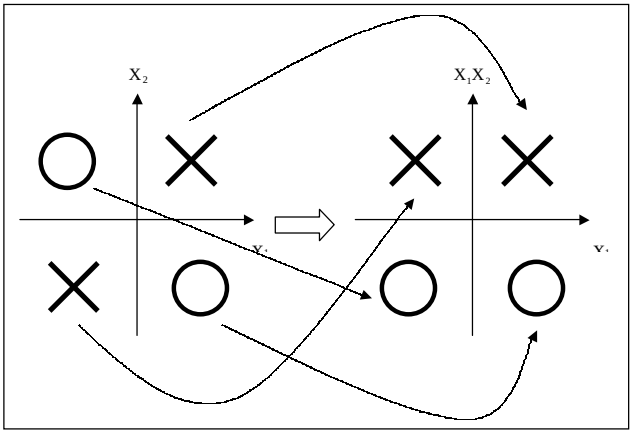
$$\sum_{i=1,n} \alpha_i^* y_i \vec{x}_i = \sum_{i:\vec{x}_i \in SV} \alpha_i^* y_i \vec{x}_i$$

Non-linear Separable Cases (1)

- Non-linear, but separable
  - No hyperplane can separate the training set
- Solution:
  - Mapping to some higher dimensional feature spaces, hoping that they are linearly separable in the higher dimensional feature spaces
  - If one can find a mapping $\phi(x)$ such that the training set becomes linearly separable, then the maximum margin classifier can be solved in the same manner
  - Mapping functions frequently used in NLP: polynomial function (usually with polynomial order less than 3, as it already has a huge dimensionality)

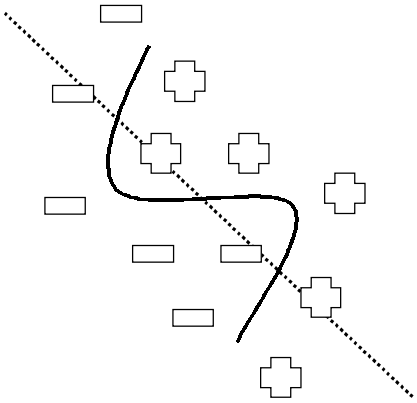
Non-linear Separable Cases (2)

- Polynomial Mapping Example: $\phi(x) = [X1, X1*X2]$



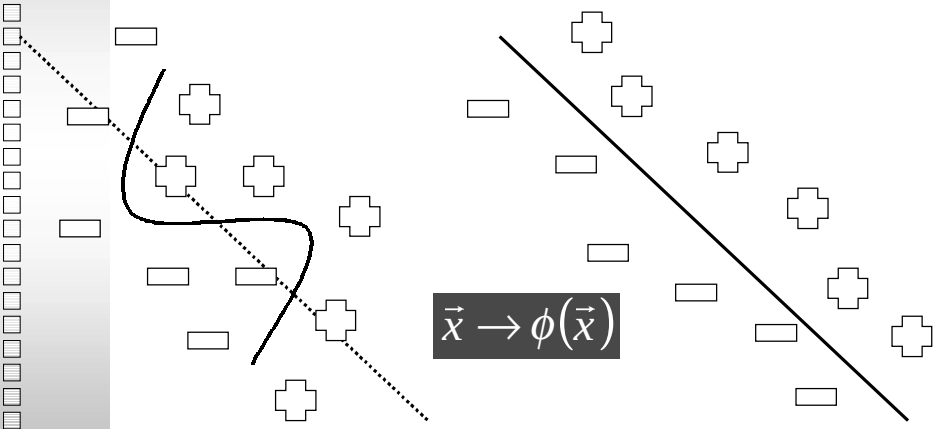
Non-linear Separable Cases (3)

- Another Example:



Non-linear Separable Cases (4)

- Kernel Mapping



Extension to non-linear cases

- Solution using implicit mapping:

$$\Psi : \vec{x} \to \phi(\vec{x}),\ k(\vec{x}_1, \vec{x}_2) \equiv \phi(\vec{x}_1) \cdot \phi(\vec{x}_2)$$

$$\text{maximize} \quad L(\vec{\alpha}) = \left(\sum_{i=1,n} \alpha_i \right) - \frac{1}{2} \sum_{i=1,n} \sum_{j=1,n} \alpha_i \alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j)$$

$$\text{subject to} \quad \sum_{i=1,n} \alpha_i y_i = 0 \text{ and } 0 \le \alpha_i$$

- Classifier:

$$\hat{y} = \text{sgn}\left(\sum_{\vec{x}_i \in SV} \alpha_i^* y_i k(\vec{x}_i, \vec{x}) + b \right)$$

Non-Separable Cases

- If non-separable: Make a tradeoff between training error and margin

$$\text{minimize} \quad J(\vec{w}, b, \vec{\xi}) \equiv \frac{1}{2} \vec{w} \cdot \vec{w} + C \sum_{i=1,n} \xi_i$$

$$\text{subject to} \quad y_i (\vec{w} \cdot \vec{x}_i + b) \ge 1 - \xi_i, \xi_i \ge 0, \forall i$$

C : controlling parameter for the tradeoff between error and margin

$\sum_i \xi_i$: upper bound on the number of training errors

$$\text{Dual}: \text{maximize } L(\vec{\alpha}), \text{subject to } \sum_{i=1,n} \alpha_i y_i = 0 \text{ and } 0 \le \boxed{\alpha_i < C}$$

- Need to define inner product (for distance) in the mapped feature space
  - Equal weighting for each symbol feature value (e.g., word existence indicator) in evaluating distance during searching hyperplane with maximum margin

Multi-Class Classification

- SVM is basically a two-class classifier
- Almost all NLP applications have several classes (e.g., part-of-speech)
- Extend binary classifiers into multi-class classifier
  - Re-derive a unified multi-class optimization equation
    - Obtained boundary usually is not optimal [Weston 98]
  - Combine pairwise binary classifiers [Kudo 01]
    - Design a binary SVM for every combination of class-pair
    - Adopt majority vote to decide the final candidate
    - Usually has better performance with high computation cost
  - Combine one-versus-rest binary classifiers [Nakagawa 02]
    - Design a binary SVM for every class versus all other classes
    - Choose the one with the highest positive distance
    - Usually has similar performance with less computation cost

A Part-of-Speech Tagging Example

- Revision Learning for tagging part-of-speech [Nakagawa 02]
  - Learn One-versus-Rest SVM for each labeled token in the training set
    - First use a stochastic trigram tagger to generate ranked POS candidates for each token
    - If the associated tag is correct, create a positive example for that corresponding class
    - If the associated tag is incorrect, create a negative example for the corresponding class of that incorrect tag; also create a positive example for the corresponding class of the correct tag
  - In testing, first use a stochastic trigram tagger to generate ranked POS candidates for each token
  - Then use SVM to verify each tag generated by the stochastic trigram tagger

A Part-of-Speech Tagging Example (Cont.)

	Total Number of Examples for SVMs	Training Time (hour)	Testing Time (second)	Accuracy
T3 Original	–	0.004	89	96.59%
with RL (polynomial kernel, cutoff-1)	1027840	16	2089	96.98%
with RL (linear kernel, cutoff-1)	1027840	2	129	96.94%
TnT	–	0.002	4	96.62%
SVMs 1-v-r	999984 * 50	625	55239	97.11%

Table 2：Computational Cost of English POS Tagging [Nakagawa et al. ACL-02]

General Comments on SVM

- A universal learning algorithm plus a problem specific kernel
  - Which kernel function should be adopted usually is not clear
- Good generalization capability
  - Balance between training and testing errors
- Large computational cost
  - Slight improvement might not justify the required computation load
- Better than Neural Net (in performance and analytic form)
  - But, will it be like NN, just another big bubble?
- Publicly accessible resources:
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - http://svmlight.joachims.org/
  - http://www.ai.univie.ac.at/oefai/ml/ml-resources.html (General ML)

Maximum Entropy Models

A way to combine different knowledge sources under probabilistic framework

- What Maximum Entropy aims to:
  - Many different hints (i.e., features, or knowledge sources) are needed in NLP for making decision (as NLP is complicated)
  - Since the number of those features is large, the associated probability values of their joint events cannot be reliably estimated from the corpus; appropriate smoothing technique is thus required
  - Traditional smoothing techniques (Linear Interpolation, Good-Turing, Backoff, etc.) have some drawbacks (as explained below)
  - Maximum Entropy technique hopes to offer a new solution to the problems (actually, it also brings in its own problem)

Inconsistency of marginal distributions in Linear Interpolation and Backoff

- Linear Interpolation [Rosenfeld 1996]:
  - Weighting coefficients are usually fixed for different information sources. It is not easy to adopt many different values according to their strengths and weakness in particular contexts (i.e., weights are optimized globally, not locally)
  - Marginal distributions obtained from the joint distribution (after smoothing) are not consistent with the data (see illustration figure)
- Backoff Smoothing [Rosenfeld 1996]:
  - Exhibits a discontinuity around the point where the backoff decision is made (i.e., choosing one marginal distribution)
  - Marginal distributions obtained from the joint distribution (after smoothing) are not consistent with the data (see illustration figure)

Inconsistency of marginal distributions in Linear Interpolation and Backoff (Cont.)

- Outcome Space

Outcome Space (w,f1,f2) partitioned into various equivalent classes by f1 and f2				
	f1= x1	f1 = x2	…	…
f2 = y1	…	…	…	…
f2 = y2	…	…	…	…
…	…	…	...	…
…	…	…	…	…

How ME Model Handle that

- Assign probabilities which obey all (desired) constraints exhibited in the data, and does not make any other extra assumptions
  - Empirical expectation value of the i-th feature should be identical to its true/desired expectation value
  - For those events not observed in the data, assume we know nothing about that (make flat guessing, that is, maximize entropy)
- Therefore, ME is a single combined model that simultaneously matches all (desired) constraints observed in the data
  - Not construct several separate models and then combine them later
  - Integrate various information sources into one combined model, and choose the event distribution with highest entropy among all probability distribution that satisfy these constraints
  - It guarantees a unique solution under the proposed exponential form

Example for predicting word

- Various possible Heterogeneous knowledge sources for predicting a new word
  - Uni-gram: context free
  - N-gram: short-term history
  - Class-based N-gram: short-term class history
  - Long-distance N-gram: long-distance dependency
  - Trigger Pairs: arbitrarily related history
  - Various kinds of hetrogeneous knowledge sources
- Each Smoothing method can be regarded as a kind of combination of knowledge sources of different historical tracks

Maximum Entropy Model (1) [Rosenfeld 96]

- View each information source as defining a subset of the event space (h, w) [h: history, w: the word to be predicted]
- For each subset, impose a constraint on the combined estimate to be derived: that
  - It must agree, *on average*, with a certain statistics of the training data, defined over that subspace.
  - e.g., the "statistics" refers to the marginal distribution of the training data in each one of the equivalent classes of history

Maximum Entropy Model (2)

- Features as Constraints
  - Indicator function: K_j (h,w) = 1 if some condition holds for h and w, K_j (h,w) =0 if otherwise
  - Example, [Jelinek 1997]: To construct a bigram language model with distribution P(x,y) satisfying the constraints

$$P(x,y) = f(x,y) \quad \text{if } C(x,y) \geq K$$
$$P(x) = f(x) \quad \text{if } C(x,y) \geq L$$
$$P(y) = f(y) \quad \text{if } C(x,y) \geq L$$
$$\sum_{x,y} P(x,y) = 1$$

P(.): probability
f(.): relative frequency
K(.): constraints

$$K_{\{x',y'\}}(x,y) \equiv \begin{cases} 1 & \text{if } x = x', y = y' \\ 0 & otherwise \end{cases} \quad \forall x', y', \text{ s.t. } C(x',y') \geq K$$

$$K_{\{x'\}}(x,y) \equiv \begin{cases} 1 & \text{if } x = x' \\ 0 & otherwise \end{cases} \quad \forall x', \text{ s.t. } C(x') \geq L$$

$$K_{\{y'\}}(x,y) \equiv \begin{cases} 1 & \text{if } y = y' \\ 0 & otherwise \end{cases} \quad \forall y', \text{ s.t. } C(y') \geq L$$

Maximum Entropy Model (3)

- Goal:
  - Given an event space {x}, a set of constraint functions $f_i(x)$ and their desired expectations K_i, find the probability assignment to Pr(X) which satisfies the constraints and has the highest entropy.
- Maximize:
  - $H(X) = -\Sigma_x \Pr(x)\log\Pr(x)$
  - Equivalent to minimizing D(P||Q), with Q ~ uniform distribution

Subject to the following constraints:

$$E[f_i(x)] = \sum_x \Pr(x) f_i(x) = K_i \quad \text{for all } i = 1, m$$

$$\sum_x \Pr(x) = 1$$

Maximum Entropy Model (4)

- Optimal Solution:
  - A constrained optimization problem
  - Solved by introducing a Lagrange multiplier for each constraint and make gradient (w.r.t. Pr) to be zero vector
- Solution Form:

$$\Pr(x) = \prod_i \mu_i^{f_i(x)}$$

$$\log(\Pr(x)) = \sum_i f_i(x) \cdot \log \mu_i$$

Generalized Iterative Scaling

- GIS for finding unknown constants μ_i [Darroch & Ratcliff 72], an EM variant
  - Start with any initial weights
  - Compute current expectation of the various feature functions
  - Scale up the weights by the ratio between the desired expectation and the current expectation
  - Update the probability distribution
  - Repeat until converge
- Convergence is guaranteed for consistent constraints

Generalized Iterative Scaling (Cont.)

$$\text{Initialization}: K_i, \mu_i^{(0)}, \Pr^{(0)}(x) = \prod_i \mu_i^{(0) f_i(x)}$$

$$f_0(x) \equiv M - \sum_i f_i(x), \forall x, \text{ where } M \equiv \max_x \sum_i f_i(x)$$

Iteration t:

$$\text{Step-1}: E_{f_i}^{(t)} = \sum_x \Pr^{(t)}(x) f_i(x), \forall i$$

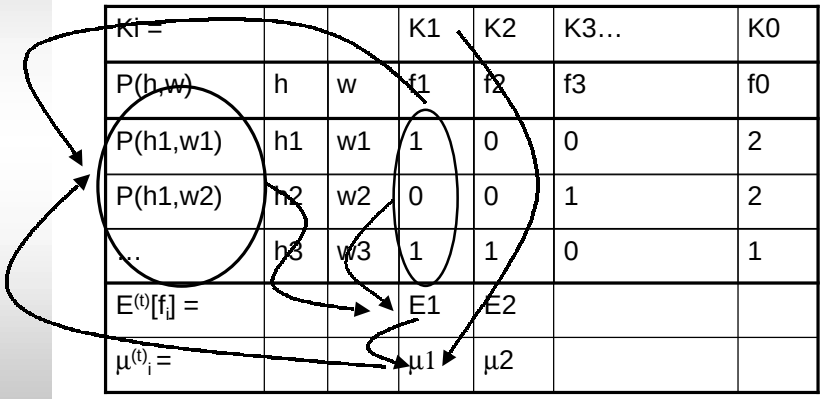
$$\text{Step-2}: \mu_i^{(t+1)} = \mu_i^{(t)} \cdot \left(\frac{K_i}{E_{f_i}^{(t)}} \right)^{\frac{1}{M}}$$

$$\text{Step-3}: \Pr^{(t+1)}(x) = \prod_i \mu_i^{(t+1) f_i(x)}$$

$f_0(x)$: augmented constraint that makes $\sum_i f_i(x)$ constant

GIS Training

- Computing M.E. Variables (a tabular view):



Ki =			K1	K2	K3…	K0
P(h,w)	h	w	f1	f2	f3	f0
P(h1,w1)	h1	w1	1	0	0	2
P(h1,w2)	h2	w2	0	0	1	2
…	h3	w3	1	1	0	1
$E^{(t)}[f_i]$ =			E1	E2		
$\mu^{(t)}_i$ =			μ1	μ2		

Evaluating Conditional Distributions: Pr(w|h)

- Only moderately successful to compute Pr(w|h) by estimating Pr(h,w) first.
- Modified ME:

$$\text{Desired expectation re-formulated}: x = (h, w)$$

$$K_i \equiv \sum_h P(h) \sum_w P(w \mid h) f_i(h, w), \forall i$$

$$\text{Modified constraints}:$$

$$K_i = \sum_h \tilde{P}(h) \sum_w P(w \mid h) f_i(h, w), \forall i$$

$$\tilde{P}(h, w), \tilde{P}(h): \text{empirical distribution}$$

Evaluating Conditional Distributions: Pr(w|h)

- Solution:

$$P(w|h) = \frac{1}{Z_\lambda(h)} \exp\left(\sum_i \lambda_i f_i(h,w)\right) = \frac{1}{Z_\lambda(h)} \prod_i \alpha_i^{f_i(h,w)}$$

- Z_λ : normalizing constant for sum of P(w|h) over all w's to be 1.0

Intuitive Interpretation

- Heuristic Interpretation

$$P(y|x) = \frac{1}{Z_\lambda(x)} \exp\left(\sum_i \lambda_i f_i(x,y)\right) = \frac{1}{Z_\lambda(x)} \prod_i \alpha_i^{f_i(x,y)}$$

  - Product of exponential power of indicator functions of weights
  - Up-weighted if condition satisfied, otherwise, down-weight it

Example: Tagging Part-of-Speech [Ratnaparkhi 96]

- Features & Performance for tagging part-of-speech

Condition	Features	
W(i) is not rare	W(i)=X	& Ti = T
W(i) is rare	|prefix(W(i))| <= 4	& Ti = T
C(W(i))<C0	|suffix(W(i))| <= 4	& Ti = T
	W(i) contains number	& Ti = T
	W(i) contains uppercase char	& Ti = T
	W(i) contains hyphen	& Ti = T
For all W(i)	T(i-1) = X	& Ti = T
	T(i-1)T(i-2) = XY	& Ti = T
	W(i-1) = X	& Ti = T
	W(i-2) = X	& Ti = T
	W(i+1) = X	& Ti = T
	W(i+2) = X	& Ti = T

Total word accuracy	Unknown word accuracy	Sentence accuracy
96.63	85.56	47.51

Applicability and Limitations (1)

- Advantages
  - Allows lots of different information to be combined
    - Can formulate complex-feature based bigrams, trigrams, long-distance N-gram, caches, triggers, class-based or not, with right indicator functions [Rosenfeld 96]
  - Combine knowledge sources, consistent to the training data, in a single combined model
    - not an interpolation of component models in which each component being trained in different event spaces
  - GIS lends itself to incremental adaptation.
    - New/old constraints can be added/removed at any time.
  - A unique ME solution is guaranteed to exist for consistent constraints. GIS is guaranteed to converge to the unique solution.

Applicability and Limitations (2)

- Disadvantages
  - Not trained directly toward the error rate criteria
  - GIS is computationally very expensive
    - Very slow to train: weeks or months
    - Requires computing probability distribution in all training contexts. For each training context:
      - Determine all indicators that might apply
      - Compute normalization constant
      - Note that number of indicators that can apply, and time to normalize are both bounded by a factor of vocabulary size.
    - Clustering usually help
  - Overfitting is possible, as no mechanism in the training process can help prevent it (unlike SVM)

Applicability and Limitations (3)

- It can also lead to absurd results [Jelinek 97]
  - For instance, impose constraints to keep sufficiently large values of L:

$$\left.\begin{array}{l} P(x) = f(x), \text{ if } C(x) \geq L \\ \& \sum_x P(x) = 1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \Pr(x) = g_0 g(x)^{k(x)} \text{ where} \\ k(x) = \begin{cases} 1 & \text{if } C(x) \geq L \\ 0 & \text{otherwise} \end{cases} \end{array}\right.$$

  - Unfortunately, the resulting probability has the following undesiredable property:
    - $P(x) = g_0$, [i.e., equally likely], for all x s.t. $C(x) < L$
    - However, it is expected that

$$P(x) > P(x') \text{ whenever } f(x) \geq f(x') + \varepsilon \text{ (if sufficiently larger)}$$

  - This is mainly due to the dichotomy that is the basis of ME model: either we know a constraint perfectly or not at all

How to embrace those new emerging Machine Learning Techniques (1)

- New Machine Learning Techniques emerge almost every certain period
  - Machine Learning Community keep inventing new techniques (e.g., Neural-Net, ME, SVM, AdaBoost, Co-Training, CoBoost, etc.)
  - It is easy to publish papers, if
    - You are the first few ones to adopt that technique to NLP, and
    - You can report better performance in some specific topics
  - However, those performance comparison should be examined carefully
    - They are compared with the baseline of other approaches sometimes
    - For example, a statistical tagger is compared without refining applied techniques (e.g., smoothing, start and end tags, etc.) [Brants 00]

How to embrace those new emerging Machine Learning Techniques (2)

- Characteristics they share:
  - Since those ML techniques are universal techniques, they usually have the capability to handle a large number of primitive features (frequently a binary indicator), and do not require a deep analysis of the problem
    - CBSO approach has only a few features, however, with a large number of possible values (searching versus computation, all needs many parameters)
  - New features can be easily added without worrying about their dependency relationship
  - They usually require heavy computation cost; however, only report a little improvement for some NLP problems (e.g., tagging, etc.)
- Most ML techniques are not significantly better than others, if all have been well-tuned
  - There is no ML technique that is consistently better than others
  - Usually, it is still the appropriate feature set that matters most

How to embrace those new emerging Machine Learning Techniques (3)

- Should we quickly grasp those new tools?
  - If you are under the pressure to publish papers, get into it as soon as possible (people would like to see how effectively that those buzz words are applied to our fields)
    - This field has struggled for a long time, every new thing is a hope
  - If you are building systems, just wait and see
    - Don't switch to a new technique, unless significant improvement has been consistently reported, if you have already adopted reasonable approaches.
    - Heavy computation requirement is frequently not emphasized in the paper. It might be too heavy for your current environment
    - The main problem might not be those issues attacked and reported in the paper (e.g., unknown words in tagging part-of-speech)