Computational Linguistics 1

CMSC/LING 723, LBSC 744

Kristy Hollingshead Seitz
Institute for Advanced Computer Studies
University of Maryland

Lecture 13: 13 October 2011

Agenda

- Homework
  - HW2 – graded end-of-day
  - HW3 – due today! (please hand in writeups at front)
- Questions, comments, concerns?
- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs
  - Features
- Midterm Review

Discriminative Training

- Statistical model training involves maximizing some *objective* function
- For an HMM, we use maximum likelihood training
  - Maximize the probability of the training set
- Reduction in errors is the true objective of learning
- Another option is to try to directly optimize error rate or some other closely related objective
- Consider not just truth, but also other candidates

Perceptron

- One approach that has been around since late 60s is the perceptron
- Basic idea:
  - Find the best scoring analysis (e.g. POS tag sequence)
  - Make its score lower, by penalizing its *features*
  - Make the score of the truth better, by rewarding its features
  - Go onto the next example

Formal Definition of Perceptron Algorithm

Formally, perceptron approach assumes:

- Training examples (x_i, y_i) for $i = 1...N$ where x_i is the input and y_i is the true output.
  - e.g. $(w_1 ... w_k, \tau_1 ... \tau_k)$, where $\tau_1 ... \tau_k$ is the true tag sequence
- A function GEN which enumerates a set of candidates GEN(x) for an input x.
  - e.g., run the tagger over input word sequence x, to output tag-sequence candidates
- A **representation** Φ mapping each $(x, y) \in X \times Y$ to a d-dimensional *feature* vector $\Phi(x, y) \in R^d$.
- A **parameter vector** $\alpha \in R^d$.
  - e.g., a vector of weights, one for each feature in Φ

Perceptron Algorithm

- **Inputs:** Training examples (x_i, y_i)
- **Initialization:** Set $\alpha = 0$
- **Algorithm:**
  - For $t = 1 \ldots T$, $i = 1 \ldots N$
    - Calculate $z_i = \text{argmax}_{z \in GEN(xi)} \Phi(x_i, z) \cdot \alpha$
    - If $(z_i \neq y_i)$ then $\alpha = \alpha + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
- **Output:** Parameters α

Perceptron: Notes

- Because this technique is optimizing (sequence) error rate, it does not involve a normalization factor
- Thus, it will overtrain
  - i.e. it will do very well on the training set, but not so well on new data, like unsmoothed maximum likelihood
  - Techniques exist for controlling overtraining, such as regularization, voting, and averaging
- Perceptron models outperform maximum likelihood–optimized models on a range of tasks
  - POS-tagging, NP-chunking

Agenda

- Homework
- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs
  - Features
- Midterm Review

Conditional Random Fields (CRFs)

- The perceptron algorithm only pays attention to best-scoring (argmax) path
- What if there were two top analyses, very close in score?
  - Should penalize features on both
  - How do we allocate the penalty?
- CRFs are a way to do this, by optimizing the conditional log-likelihood of the truth

Formal Definition of CRFs

- Define a conditional distribution over the members of GEN(*x*) for a given input *x*:

$$p_{\bar{\alpha}}(y|x) = \frac{1}{Z(x, \bar{\alpha})} \exp\left(\Phi(x, y) \cdot \bar{\alpha}\right)$$

- where

$$Z(x, \bar{\alpha}) = \sum_{y \in \mathrm{GEN}(x)} \exp\left(\Phi(x, y) \cdot \bar{\alpha}\right)$$

- (Can be calculated with forward-backward algorithm!)

CRF objective function

- Choose α to maximize the conditional log-likelihood of the training data:

$$LL(\bar{\alpha}) = \sum_{i=1}^{N} \log p_{\bar{\alpha}}(y_i|x_i) = \sum_{i=1}^{N} \left[\Phi(x_i, y_i) \cdot \bar{\alpha} - \log Z(x_i, \bar{\alpha})\right]$$

- Use a zero-mean Gaussian prior on the parameters resulting in the regularized objective function:

$$LL_R(\bar{\alpha}) = \sum_{i=1}^{N} \left[\Phi(x_i, y_i) \cdot \bar{\alpha} - \log Z(x_i, \bar{\alpha})\right] - \frac{||\bar{\alpha}||^2}{2\sigma^2}$$

- where the value σ is typically estimated on heldout data.

CRF Optimization

- The objective function is convex and there is a globally optimal solution
- Can use general numerical optimization techniques to find the global optimum
  - e.g. for a language modeling project we used a general *limited memory variable metric* method to optimize LL_R from a publically available software library
- The optimizer needs the function value and the derivative (or gradient)

Derivative of LL_R: Refresher

Remember the chain rule:

$$\frac{df(g(x))}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

Also remember derivative of (natural) log:

$$\frac{d\log(x)}{dx} = \frac{1}{x}$$

And don't forget the derivative of exp:

$$\frac{d\exp(ax)}{dx} = a\exp(ax)$$

Derivative of LL_R

$$\begin{aligned} LL_R(\bar\alpha) &= \sum_{i=1}^{N}[\Phi(x_i, y_i)\cdot\bar\alpha - \log Z(x_i,\bar\alpha)] - \frac{||\bar\alpha||^2}{2\sigma^2} \\ &= \sum_{i=1}^{N}\left[\sum_{s=1}^{d}\Phi_s(x_i,y_i)\alpha_s - \log\sum_{y\in \mathrm{GEN}(x_i)}\exp(\sum_{j=1}^{d}\Phi_j(x_i,y)\bar\alpha_j)\right] - \sum_{m=1}^{n}\frac{\bar\alpha_m^2}{2\sigma^2} \end{aligned}$$

$$\begin{aligned} \frac{\partial LL_R}{\partial\alpha_s} &= \sum_{i=1}^{N}\left[\Phi_s(x_i,y_i) - \frac{\sum_{y\in \mathrm{GEN}(x_i)}\exp(\sum_{j=1}^{d}\Phi_j(x_i,y)\bar\alpha_j)\Phi_s(x_i,y)}{\sum_{y\in \mathrm{GEN}(x_i)}\exp(\sum_{j=1}^{d}\Phi_j(x_i,y)\bar\alpha_j)}\right] - \frac{2\alpha_s}{2\sigma^2} \\ &= \sum_{i=1}^{N}\left[\Phi_s(x_i,y_i) - \sum_{y\in \mathrm{GEN}(x_i)}\frac{\exp(\sum_{j=1}^{d}\Phi_j(x_i,y)\bar\alpha_j)}{Z(x_i,\bar\alpha)}\Phi_s(x_i,y)\right] - \frac{\alpha_s}{\sigma^2} \\ &= \sum_{i=1}^{N}\left[\Phi_s(x_i,y_i) - \sum_{y\in \mathrm{GEN}(x_i)}p(y|x_i)\Phi_s(x_i,y)\right] - \frac{\alpha_s}{\sigma^2} \end{aligned}$$

Perceptron vs CRFs

- Training time
  - More expensive (calculating derivative) for CRFs...
  - ...but can be parallelized
- Performance
  - In Sha & Pereira, perceptron performance not statistically significantly different from CRF with same feature set

Agenda

- Homework
- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs
  - Features
- Midterm Review

Features (Φ)

- Good feature sets matter a lot
- These discriminative methods allow for easy use of many features
  - Unlike HMM based methods
- Examples of feature sets

Features for Shallow Parsing

Sha & Pereira, 2003

c_i is the class of w_i

t_i is the POS-tag of w_i

$y_i = c_{i-1}c_i$

e.g. BI or IO,

but never OI

$c(y_i) = c_i$

$q(y_{i-1}, y_i)$	$p(x, i)$
$y_i = y$ $y_i = y,\ y_{i-1} = y'$ $c(y_i) = c$	true
$y_i = y$ or $c(y_i) = c$	$w_i = w$ $w_{i-1} = w$ $w_{i+1} = w$ $w_{i-2} = w$ $w_{i+2} = w$ $w_{i-1} = w',\ w_i = w$ $w_{i+1} = w',\ w_i = w$ $t_i = t$ $t_{i-1} = t$ $t_{i+1} = t$ $t_{i-2} = t$ $t_{i+2} = t$ $t_{i-1} = t',\ t_i = t$ $t_{i-2} = t',\ t_{i-1} = t$ $t_i = t',\ t_{i+1} = t$ $t_{i+1} = t',\ t_{i+2} = t$ $t_{i-2} = t'',\ t_{i-1} = t',\ t_i = t$ $t_{i-1} = t'',\ t_i = t',\ t_{i+1} = t$ $t_i = t'',\ t_{i+1} = t',\ t_{i+2} = t$

Table 1: Shallow parsing features

Features for Tagging, & OOVs

Ratnaparkhi, 1993

Condition	Features	
w_i is not rare	$w_i = X$	$\& \ t_i = T$
w_i is rare	X is prefix of w_i, $\lvert X \rvert \leq 4$	$\& \ t_i = T$
	X is suffix of w_i, $\lvert X \rvert \leq 4$	$\& \ t_i = T$
	w_i contains number	$\& \ t_i = T$
	w_i contains uppercase character	$\& \ t_i = T$
	w_i contains hyphen	$\& \ t_i = T$
$\forall \ w_i$	$t_{i-1} = X$	$\& \ t_i = T$
	$t_{i-2}t_{i-1} = XY$	$\& \ t_i = T$
	$w_{i-1} = X$	$\& \ t_i = T$
	$w_{i-2} = X$	$\& \ t_i = T$
	$w_{i+1} = X$	$\& \ t_i = T$
	$w_{i+2} = X$	$\& \ t_i = T$

Table 1: Features on the current history h_i

Instantiated Features

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

Table 2: Sample Data

$w_i =$ `about`	$\& \ t_i =$ `IN`
$w_{i-1} =$ `stories`	$\& \ t_i =$ `IN`
$w_{i-2} =$ `the`	$\& \ t_i =$ `IN`
$w_{i+1} =$ `well-heeled`	$\& \ t_i =$ `IN`
$w_{i+2} =$ `communities`	$\& \ t_i =$ `IN`
$t_{i-1} =$ `NNS`	$\& \ t_i =$ `IN`
$t_{i-2}t_{i-1} =$ `DT NNS`	$\& \ t_i =$ `IN`

Table 3: Features Generated From h_3 (for tagging **about**) from Table 2

Instantiated Features

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

$w_{i-1} =$ `about`	$\& \ t_i =$ `JJ`
$w_{i-2} =$ `stories`	$\& \ t_i =$ `JJ`
$w_{i+1} =$ `communities`	$\& \ t_i =$ `JJ`
$w_{i+2} =$ `and`	$\& \ t_i =$ `JJ`
$t_{i-1} =$ `IN`	$\& \ t_i =$ `JJ`
$t_{i-2}t_{i-1} =$ `NNS IN`	$\& \ t_i =$ `JJ`
prefix(w_i)=`w`	$\& \ t_i =$ `JJ`
prefix(w_i)=`we`	$\& \ t_i =$ `JJ`
prefix(w_i)=`wel`	$\& \ t_i =$ `JJ`
prefix(w_i)=`well`	$\& \ t_i =$ `JJ`
suffix(w_i)=`d`	$\& \ t_i =$ `JJ`
suffix(w_i)=`ed`	$\& \ t_i =$ `JJ`
suffix(w_i)=`led`	$\& \ t_i =$ `JJ`
suffix(w_i)=`eled`	$\& \ t_i =$ `JJ`
w_i contains hyphen	$\& \ t_i =$ `JJ`

Table 4: Features Generated From h_4 (for tagging **well-heeled**) from Table 2

Agenda

- Homework
- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs
  - Features
- Midterm Review

Midterm Topics

- Sequences and n-grams
- FSAs, FSTs
  - Construction
  - Composition
- Smoothing
  - Algorithms
  - Interpolation, Backoff
- HMMs
  - Tagging
  - Viterbi
  - Forward-Backward

Midterm Format

- Some short answer questions
- Some basic numerical computation
- Questions from the homeworks
- *No programming*

- Ground rules:
  - Work completely independently – no communication of any kind
    - No communication with the TA or instructor
  - Open book, open note. *Not* open internet, except for web pages explicitly linked from the class webpage.
  - Turn in a hard copy on Tuesday October 25 (or earlier, to Kristy)

Agenda: Summary

- Supervised Learning – Discriminative Training
  - Perceptron
  - CRFs
  - Features
- Midterm Review