MaxEnt (II): Modeling and Decoding

LING 572

Advanced Statistical Methods for NLP

January 31, 2019

Based on Xia '17

Outline

- Overview
- The Maximum Entropy Principle

- Modeling**
- Decoding

- Training**

- Case study

Modeling

The Setting

- From the training data, collect (x, y) pairs:
  - x in X: observed data
  - y in Y: thing to be predicted (e.g., a class in a classification problem)
  - Ex: In a text classification task
    - x: a document
    - y: the category of the document

- Goal: estimate $P(y \mid x)$

Basic Idea

- Goal: estimate p(y | x)

- Choose p(x, y) with maximum entropy (or "uncertainty") subject to the constraints (or "evidence").

$$H(p) = -\sum_{(x,y)\in X \times Y} p(x,y)\log p(x,y)$$

Outline for Modeling

- Feature function: $f_j(x, y)$

- Calculating the expectation of a feature function

- Forms of P(x, y) and P(y | x)

Feature function

Definition

- A feature function is a binary-valued function on events:

$$f_j : X \times Y \rightarrow \{0,1\}$$

The j in f_j corresponds to a (feature, class) pair (t, c)

$f_j(x, y) = 1$ iff t is present in x and $y = c$.

- Ex:

$$f_j(x, y) = \begin{cases} 1 & if \;\; y = Politics \; \& \; x \; contains \;\; "the" \\ 0 & o.w. \end{cases}$$

Weights in NB

	f_1	f_2	…	f_k
c_1				
c_2				
…				
c_i				

Weights in NB

	f_1	f_2	**…**	f_j
c_1	$P(f_1 \mid c_1)$	$P(f_2 \mid c_1)$	…	$P(f_j \mid c_1)$
c_2	$P(f_1 \mid c_2)$	…	…	…
…	…			
c_i	$P(f_1 \mid c_i)$	…	…	$P(f_j \mid c_i)$

Each cell is a weight for a particular (class, feat) pair.

Matrix in MaxEnt

	t_1	t_2	…	t_k
c_1	f_1	f_2	…	f_k
c_2	f_{k+1}	f_{k+2}	…	f_{2k}
…	…			
c_i	$f_{k*(i-1)+1}$			f_{k*i}

Each feature function f_j corresponds to a (feat, class) pair.

Weights in MaxEnt

	t_1	t_2	…	t_k
c_1	λ_1	λ_2	…	λ_k
c_2	…	…	…	…
…	…			
c_i	…			λ_{ki}

Each feature function f_j has a weight λ_j.

Feature function summary

- A feature function in MaxEnt corresponds to a (feat, class) pair.
- The number of feature functions in MaxEnt is approximately |C| * |V|.
- A MaxEnt trainer learns the weights for the feature functions.

The outline for modeling

- Feature function: $f_j(x, y)$
- Calculating the expectation of a feature function
- The forms of P(x,y) and P(y | x)

Expected Return

- Ex1:
  - Flip a coin
    - if it's heads, you win 100 dollars
    - if it's tails, you lose 50 dollars
  - What is the expected return?

    P(X=H) * 100 + P(X=T) * (-50)

- Ex2:
  - If it is x_i, you will receive v_i dollars?
  - What is the expected return?

$$\sum_i P(X = x_i)\ v_i$$

expectation of a function

Let $P(X = x)$ be a distribution of a random variable X.
Let $f(x)$ be a function of x.
Let $E_p(f)$ be the expectation of $f(x)$ based on $P(x)$.

$$E_P(f) = \sum_i P(X = x_i) \boxed{f(x_i)}$$

$$\sum_i P(X = x_i) \boxed{v_i}$$

Empirical expectation

- Denoted as : $\tilde{p}(x)$

- Ex1: Toss a coin four times and get H, T, H, and H.
- The average return: (100-50+100+100)/4 = 62.5

- Empirical distribution: $\tilde{p}(X = H) = 3/4$

  $\tilde{p}(X = T) = 1/4$
- Empirical expectation:

  ¾ * 100 + ¼ * (-50) = 62.5

Model Expectation

- Ex1: Toss a coin four times and get H, T, H, and H.

- A model: $p(x)$
  - Assume a fair coin
  - P(X=H) = P(X=T) = 1/2

- Model expectation:

  1/2 * 100 + 1/2 * (-50) = 25

Some Notation

Training data: S

Empirical distribution: $\tilde{p}(x, y)$

A model: $p(x, y)$

The j[th] feature function: $f_j(x, y)$

Empirical expectation of f_j $$E_{\tilde{p}} f_j = \sum_{(x,y)\in X \times Y} \tilde{p}(x, y) f_j(x, y)$$

Model expectation of f_j $$E_p f_j = \sum_{(x,y)\in X \times Y} p(x, y) f_j(x, y)$$

Empirical expectation**

$$E_{\tilde{p}} f_j = \sum_{x \in X, y \in Y} \tilde{p}(x, y) f_j(x, y)$$

$$= \sum_{x \in X, y \in Y} \tilde{p}(x) \tilde{p}(y \mid x) f_j(x, y) = \sum_{x \in X} \tilde{p}(x) \sum_{y \in Y} \tilde{p}(y \mid x) f_j(x, y)$$

$$= \sum_{x \in S} \tilde{p}(x) \sum_{y \in Y} \tilde{p}(y \mid x) f_j(x, y) = \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in Y} \tilde{p}(y \mid x_i) f_j(x_i, y)$$

$$= \frac{1}{N} \sum_{i=1}^{N} f_j(x_i, y_i)$$

An example

- Training data:

  x1 c1 t1 t2 t3

  x2 c2 t1 t4

  x3 c1 t3 t4

  x4 c3 t1 t3

Raw counts $\sum_{i=1}^{N} f_j(x_i, y_i)$

	t1	t2	t3	t4
c1	1	1	2	1
c2	1	0	0	1
c3	1	0	1	0

$$E_{\tilde{p}} f_j = \frac{1}{N} \sum_{i=1}^{N} f_j(x_i, y_i)$$

An example

- Training data:

  x1 c1 t1 t2 t3

  x2 c2 t1 t4

  x3 c1 t3 t4

  x4 c3 t1 t3

Empirical expectation

	t1	t2	t3	t4
c1	1/4	1/4	2/4	1/4
c2	1/4	0/4	0/4	1/4
c3	1/4	0/4	1/4	0/4

$$E_{\tilde{p}} f_j = \frac{1}{N} \sum_{i=1}^{N} f_j(x_i, y_i)$$

Calculating empirical expectation

Let N be the number of training instances

for each instance x in the training data

let y be the true class label of x

for each feature t in x

empirical_expect [t] [y] += 1/N

Model expectation**

$$E_p f_j = \sum_{x \in X, y \in Y} p(x, y) f_j(x, y)$$

$$= \sum_{x \in X, y \in Y} p(x) p(y \mid x) f_j(x, y) \qquad \approx \sum_{x \in X, y \in Y} \tilde{p}(x) p(y \mid x) f_j(x, y)$$

$$= \sum_{x \in X} \tilde{p}(x) \sum_{y \in Y} p(y \mid x) f_j(x, y) \qquad = \sum_{x \in S} \tilde{p}(x) \sum_{y \in Y} p(y \mid x) f_j(x, y)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in Y} p(y \mid x_i) f_j(x_i, y)$$

An example

- Suppose $P(y \mid x_i) = 1/3$
- Training data:

x1	t1 t2 t3
x2	t1 t4
x3	t4
x4	t1 t3

"Raw" counts

	t1	t2	t3	t4
c1	3/3	1/3	2/3	2/3
c2	3/3	1/3	2/3	2/3
c3	3/3	1/3	2/3	2/3

$$E_p f_j = \frac{1}{N} \boxed{\sum_{i=1}^{N} \sum_{y \in Y} p(y \mid x_i) f_j(x_i, y)}$$

An example

- Suppose $P(y \mid x_i) = 1/3$
- Training data:

x1 t1 t2 t3

x2 t1 t4

x3 t4

x4 t1 t3

Model expectation

	t1	t2	t3	t4
c1	3/12	1/12	2/12	2/12
c2	3/12	1/12	2/12	2/12
c3	3/12	1/12	2/12	2/12

$$E_p f_j = \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in Y} p(y \mid x_i) f_j(x_i, y)$$

Calculating model expectation

$$E_p f_j = \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in Y} p(y \mid x_i) f_j(x_i, y)$$

Let N be the number of training instances

for each instance x in the training data

  calculate P(y | x) for every y in Y

  for each feature t in x

    for each y in Y

      model_expect [t] [y] += 1/N * P(y | x)

Empirical expectation vs. model expectation

$$E_{\tilde{p}} f_j = \frac{1}{N} \sum_{i=1}^{N} f_j(x_i, \boxed{y_i})$$

$$E_p f_j = \frac{1}{N} \sum_{i=1}^{N} \boxed{\sum_{y \in Y}} \boxed{p(y \mid x_i)} f_j(x_i, \boxed{y})$$

Outline for modeling

- Feature function: $f_j(x, y)$

- Calculating the expectation of a feature function

- The forms of P(x, y) and P(y | x)**

Constraints

- Model expectation = Empirical expectation

$$E_p f_j = E_{\tilde{p}} f_j = d_j$$

- Why impose such constraints?
  - MaxEnt principle: Model what is known
  - Maximize the conditional likelihood: see Slides #24-28 in (Klein and Manning, 2003)

The conditional likelihood (**)

- Given the data (X,Y), the conditional likelihood is a function of the parameters ¸

$$logP(Y|X,\lambda)$$

$$= log \prod_{(x,y)\in(X,Y)} P(y|x,\lambda)$$

$$= \sum_{(x,y)\in(X,Y)} logP(y|x,\lambda)$$

$$= \sum_{(x,y)\in(X,Y)} log \frac{e^{\sum_j \lambda_j f_j(x,y)}}{\sum_{y\in Y} e^{\sum_j \lambda_j f_j(x,y)}}$$

$$= \sum_{(x,y)\in(X,Y)} (log\ e^{\sum_j \lambda_j f_j(x,y)} - log \sum_{y\in Y}\ e^{\sum_j \lambda_j f_j(x,y)})$$

$$= \ldots$$

The effect of adding constraints

- Bring the distribution closer to the data
- Bring the distribution further away from uniform
- Lower the entropy
- Raise the likelihood of data

Restating the problem

The task: find p* s.t. $p^* = \arg\max_{p \in P} H(p)$

where $P = \{ p \mid E_p f_j = E_{\tilde{p}} f_j, j = \{1,\ldots,k\} \}$

Objective function: H(p)

Constraints: $\{ E_p f_j = E_{\tilde{p}} f_j = d_j, j = \{1,\ldots,k\} \}$

Using Lagrange multipliers (**)

Minimize A(p): $$A(p) = -H(p) - \sum_{j=1}^{k} \lambda_j (E_p f_j - d_j) - \lambda_0 (\sum_{x,y} p(x,y) - 1)$$

$$A'(p) = 0$$

$$\Rightarrow \frac{\delta(\sum_{x,y} p(x,y) \ln p(x,y) - \sum_{j=1}^{k} \lambda_j ((\sum_{x,y} p(x,y) f_j(x,y)) - d_j) - \lambda_0 (\sum_{x,y} p(x,y) - 1))}{\delta p(x,y)} = 0$$

$$\Rightarrow 1 + \ln p(x,y) - \sum_{j=1}^{k} \lambda_j f_j(x,y) - \lambda_0 = 0$$

$$\Rightarrow \ln p(x,y) = (\sum_{j=1}^{k} \lambda_j f_j(x,y)) + \lambda_0 - 1$$

$$\Rightarrow p(x,y) = e^{\sum_{j=1}^{k} \lambda_j f_j(x,y) + \lambda_0 - 1} = e^{\sum_{j=1}^{k} \lambda_j f_j(x,y) + \lambda_0 - 1}$$

$$\Rightarrow p(x,y) = \frac{e^{\sum_{j=1}^{k} \lambda_j f_j(x,y)}}{Z} \quad where \quad Z = e^{1-\lambda_0}$$

Questions

$$p^* = \arg\max_{p \in P} H(p)$$

where $$P = \{p \mid E_p f_j = E_{\tilde{p}} f_j, j = \{1, \ldots, k\}\}$$

- Is P empty?
- Does p* exist?
- Is p* unique?
- What is the form of p*?
- How can we find p*?

What is the form of p*? (Ratnaparkhi, 1997)

$$P = \{ p \mid E_p f_j = E_{\tilde{p}} f_j, j = \{1, \ldots, k\} \}$$

$$Q = \{ p \mid p(x, y) = \pi \prod_{j=1}^{k} \alpha_j^{f_j(x,y)}, \alpha_j > 0 \}$$

Theorem: if $p^* \in P \cap Q$ then $p^* = \arg\max_{p \in P} H(p)$

Furthermore, p* is unique.

Two equivalent forms

$$p(x,y) = \pi \prod_{j=1}^{k} \alpha_j^{f_j(x,y)}$$

$$p(x,y) = \frac{e^{\sum_{j=1}^{k} \lambda_j f_j(x,y)}}{Z}$$

$$\Rightarrow \quad \pi = \frac{1}{Z} \qquad \lambda_j = \ln \alpha_j$$

Modeling summary

Goal: find p* in P, which maximizes H(p).

$$P = \{ p \mid E_p f_j = E_{\tilde{p}} f_j , j = \{1,\ldots,k\} \}$$

It can be proved that, **when** p* exists

- it is unique
- it maximizes the conditional likelihood of the training data
- it is a model in Q, where

$$Q = \{ p \mid p(x) = \pi \prod_{j=1}^{k} \alpha_j^{f_j(x)}, \alpha_j > 0 \}$$

Outline

- Overview
- The Maximum Entropy Principle
- Modeling**
- Decoding
- Training**
- Case study: POS tagging

Decoding

Decoding

$$p(y \mid x) = \frac{e^{\sum_{j=1}^{k} \lambda_j f_j(x,y)}}{Z}$$

Z is the normalizer.

	t_1	t_2	…	t_k
c_1	λ_1	λ_2	…	λ_k
c_2	…	…	…	…
…	…			
c_i	…			λ_{ki}

Procedure for calculating P(y | x)

```
Z=0;
for each y in Y
   sum = 0;   // or sum = default_weight_for_class_y;
   for each feature t present in x
       sum +=  weight for (t, y);
   result[y] = exp(sum);
   Z += result[y];
 for each y in Y
    P(y | x) = result[y] / Z;
```

MaxEnt summary so far

- Idea: choose the p* that maximizes entropy while satisfying all the constraints.
- p* is also the model within a model family that maximizes the conditional likelihood of the training data.
- MaxEnt handles overlapping features well.
- In general, MaxEnt achieves good performance on many NLP tasks.
- Next: Training: many methods (e.g., GIS, IIS, L-BFGS).