Machine Learning for natural language processing

Classification: Maximum Entropy Models

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2016

Introduction

- Classification = supervised method for classifying an input, given a finite set of possible classes.
- Today: Discriminative classifier based on features of the input.

Jurafsky & Martin (2015), chapter 7, Berger et al. (1996); Ratnaparkhi (1997, 1998)

Table of contents

Motivation

Just like naive Bayes, logistic regression (= maximum entropy modeling, **MaxEnt**)

- extracts a set of weighted features from the input,
- takes logs,
- and combines them linearly.

But: naive Bayes is a **generative** classifier while MaxEnt is a **discriminative** classifier.

Motivation

- **Generative classifier**: estimates the best class c for a given x indirectly from $P(x|c)$ and $P(c)$

$$\arg\max_c P(c|x) = \arg\max_c \frac{P(x|c)P(c)}{P(x)} = \arg\max_c P(x|c)P(c)$$

  I.e., the classifier models how to generate the data x from a class c.

- **Discriminative classifier**: directly computes $P(c|x)$ by discriminating among the differnt possibles classes c.

$$\arg\max_c P(c|x)$$

MaxEnt classification

Logistic regression (MaxEnt) is a **linear classifier**: it estimates $P(c|x)$ by

- extracting some set of features from the input,
- combining these features linearly (= multiplying each by a weight and adding them up)
- and then applying a function to this linear combination.

MaxEnt classification

A linear combination of k weighted features would be

$$\sum_{i=1}^{k} w_i f_i = \vec{w} \cdot \vec{f}$$

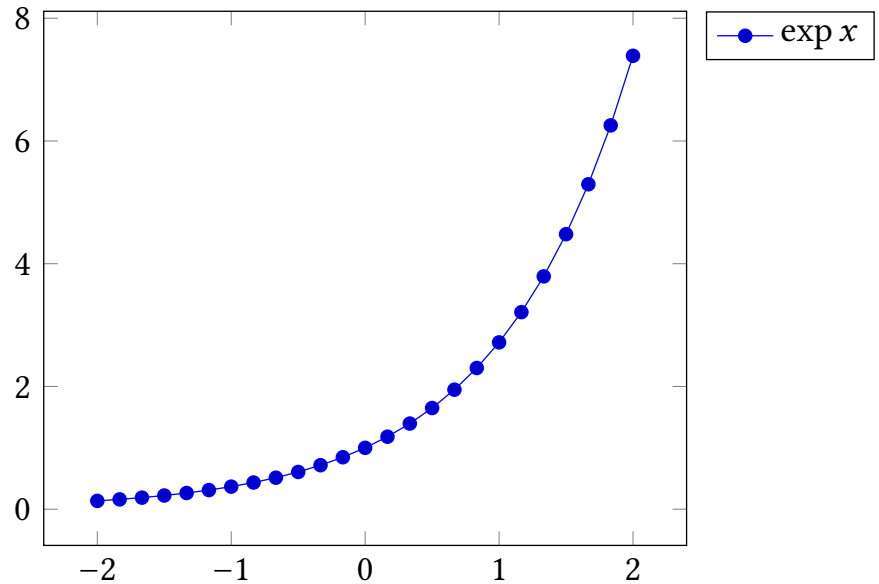
where $\vec{w}$ is the vector of the weights, $\vec{f}$ the feature vector.

But: this linear combination can be any real number, it does not give a probability. Therefore we

1. first turn $\sum_{i=1}^{k} w_i f_i$ into something positive by replacing it with $\exp \sum_{i=1}^{k} w_i f_i = e^{\sum_{i=1}^{k} w_i f_i}$, and
2. then normalize the result in order to obtain values between 0 and 1.

MaxEnt classification

Reminder: $e^0 = 1, e^1 = 2.7182818\ldots$



MaxEnt classification

The values of our features depend not only on the observation (for instance the document one wants to classify) but also on the class.

Intuition behind this: a feature f can be discriminative for one class but maybe not for another class.

For an observation x that we want to classify and a class c from the set of possible classes C, we therefore consider features $f_i(c, x)$ and our positive value becomes

$$e^{\sum_{i=1}^{k} w_i f_i(c,x)}$$

.

MaxEnt classification

With a further normalization step, in order to turn these values into probabilities, we obtain

$$P(c|x) = \frac{1}{Z} e^{\sum_{i=1}^{k} w_i f_i(c,x)}$$

where $\frac{1}{Z}$ is the normalization factor

$$\frac{1}{\sum_{c' \in C} e^{\sum_{i=1}^{k} w_i f_i(c',x)}}$$

Consequently,

$$P(c|x) = \frac{e^{\sum_{i=1}^{k} w_i f_i(c,x)}}{\sum_{c' \in C} e^{\sum_{i=1}^{k} w_i f_i(c',x)}}$$

MaxEnt classification

If we are not interested in the probability itself but only in the best class $\hat{c}$ for some observation x, we can compute

$$
\begin{aligned}
\hat{c} &= \arg\max_{c \in C} P(c|x) \\
&= \arg\max_{c \in C} \frac{e^{\sum_{i=1}^{k} w_i f_i(c,x)}}{\sum_{c' \in C} e^{\sum_{i=1}^{k} w_i f_i(c',x)}} \\
&= \arg\max_{c \in C} e^{\sum_{i=1}^{k} w_i f_i(c,x)} \\
&= \arg\max_{c \in C} \sum_{i=1}^{k} w_i f_i(c,x)
\end{aligned}
$$

In other words, for some observation x, we assign the class with the highest weighted sum of feature values.

When the classifier is embedded into some larger system it can, however, be useful to calculate the probability.

MaxEnt classification

Features that take only values 0 or 1 are called **indicator functions**.

Example adapted from Jurafsky & Martin (2015)

Goal: sentiment analysis $C = \{+, -\}$. Some possible features and weights:

$$f_1(c, x) = \begin{cases} 1 \text{ if “great” } \in x \text{ and } c = + \\ 0 \text{ otherwise} \end{cases} \qquad w_1 = 1.9$$

$$f_2(c, x) = \begin{cases} 1 \text{ if “second-rate” } \in x \text{ and } c = - \\ 0 \text{ otherwise} \end{cases} \qquad w_2 = 0.9$$

$$f_3(c, x) = \begin{cases} 1 \text{ if “no” } \in x \text{ and } c = - \\ 0 \text{ otherwise} \end{cases} \qquad w_3 = 0.7$$

$$f_4(c, x) = \begin{cases} 1 \text{ if “enjoy” } \in x \text{ and } c = - \\ 0 \text{ otherwise} \end{cases} \qquad w_4 = -0.8$$

$$f_5(c, x) = \begin{cases} 1 \text{ if “great” } \in x \text{ and } c = - \\ 0 \text{ otherwise} \end{cases} \qquad w_5 = -0.6$$

MaxEnt classification

Example continued

Observation x: "... there are virtually no surprises, and the writing is second-rate. So why did I enjoy it so much? For one thing, the cast is great ..."

Weighted feature sums:
class +: $1.9 + 0 + 0 + 0 + 0 = 1.9$
class −: $0 + 0.9 + 0.7 - 0.8 - 0.6 = 0.2$

$$P(+|x) = \frac{e^{1.9}}{e^{1.9}+e^{0.9}e^{0.7}e^{-0.8}e^{-0.6}} = \frac{e^{1.9}}{e^{1.9}+e^{0.2}} = 0.85$$

$$P(-|x) = \frac{e^{0.2}}{e^{1.9}+e^{0.2}} = 0.15$$

- We often have a large number of features.
- Features can be created automatically via feature templates.

MaxEnt classification

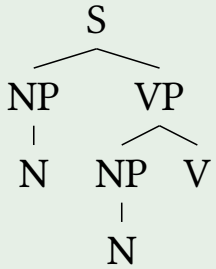
Example adapted from Malouf (2002a)

Now consider a constituency parsing task. The ME classifier ranks the different parse trees that are possible for some input string w.

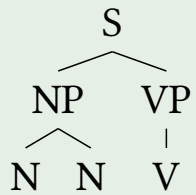
Features of tree t	weights
f_1: # of usage of $S \rightarrow NP\ VP$ in tree t	1
f_2: # of usages of $VP \rightarrow NP\ V$ in tree t	0.6
f_3: # of usages of $VP \rightarrow V$ in tree t	0.4
f_4: # of usages of $NP \rightarrow N\ N$ in tree t	0.1
f_5: # of usages of $NP \rightarrow N$ in tree t	0.9

Input: w =NNV. Question: Which is the best parse tree?

t_1:



t_2:



t_1: $1 + 0.6 + 0 + 0 + 1.8 = 3.4$, t_2: $1 + 0 + 0.4 + 0.1 = 1.5$

t_1 is the better parse tree according to this ME model.

Parameter estimation

Question: Given the features, how do we learn the parameters of the model, i.e., the weights w_i?

Logistic regression is trained with **conditional maximum likelihood estimation**.

We choose the weights $\vec{w} = \langle w_1, \ldots, w_k \rangle$ that maximize the (log) probability of the class labels of the training data.

Parameter estimation

For a single training observation x and its class c we have

$$\hat{\vec{w}} = \arg\max_{\vec{w}} \log P(c|x)$$

and for all training observations $x_1, \ldots, x_N$ with their classes $c_1, \ldots, c_N$, we obtain

$$\hat{\vec{w}} = \arg\max_{\vec{w}} \sum_{j=1}^{N} \log P(c_j|x_j)$$

(Reminder: $\log \prod_{j=1}^{N} P(c_j|x_j) = \sum_{j=1}^{N} \log P(c_j|x_j)$.)

The objective function we are maximizing is thus

$$L(\vec{w}) = \sum_{j=1}^{N} \log P(c_j|x_j)$$

Parameter estimation

Using the previously given definition of $P(c_j|x_j)$, this means

$$
\begin{aligned}
L(\vec{w}) &= \sum_{j=1}^{N} \log P(c_j|x_j) \\
&= \sum_{j=1}^{N} \log \frac{e^{\sum_{i=1}^{k} w_i f_i(c_j,x_j)}}{\sum_{c' \in C} e^{\sum_{i=1}^{k} w_i f_i(c',x_j)}} \\
&= \sum_{j=1}^{N} (\log e^{\sum_{i=1}^{k} w_i f_i(c_j,x_j)} - \log(\sum_{c' \in C} e^{\sum_{i=1}^{k} w_i f_i(c',x_j)})) \\
&= \sum_{j=1}^{N} \log e^{\sum_{i=1}^{k} w_i f_i(c_j,x_j)} - \sum_{j=1}^{N} \log(\sum_{c' \in C} e^{\sum_{i=1}^{k} w_i f_i(c',x_j)})
\end{aligned}
$$

Parameter estimation

One way to estimate our parameters is by **generalized iterative scaling** (see Ratnaparkhi (1997)). Assume that we have only indicator functions. The algorithm works if the following constraints are satisfied for all training observations x_j and their associated classes c_j, $1 \leq j \leq N$:

1. $\sum_{i=1}^{k} f_i(c_j, x_j) = m$ for some constant m.
2. There is at least one i, $1 \leq i \leq k$ such that $f_i(x_j) = 1$.

If the first condition is not satisfied, we can add a $(k+1)$st feature f_{k+1} with values as follows:

For all x_j, $1 \leq j \leq N$, $f_{k+1}(x_j) = m - \sum_{i=1}^{k} f_i(x_j)$. (Note that this feature is not an indicator function.)

Parameter estimation

The learning algorithm iteratively refines the weight vector. Assume that we have training observations $x_1, \ldots, x_N$ with classes $c_1, \ldots, c_N$. We start with

$$w_i^{(0)} = 1 \quad \text{for all } i, 1 \leq i \leq k$$

and the iteration step is

$$w_i^{(n+1)} = w_i^{(n)} \left(\frac{\frac{1}{N} \sum_{j=1}^{N} f_i(c_j, x_j)}{\sum_{j=1}^{N} \tilde{p}(x_j) \sum_{c \in C} P^{(n)}(c|x_j) f_i(c, x_j)} \right)^{\frac{1}{m}} \quad \text{for all } i, 1 \leq i \leq k$$

where $\tilde{p}(x_j)$ the observed probability of x_j in the training data and $P^{(n)}(c|x_j)$ is the probability we obtain for c given x_j with the weights of the nth iteration step.

Parameter estimation

$$\frac{1}{N}\sum_{j=1}^{N} f_i(c_j, x_j)$$

is the average value of f_i on our training data while

$$P^{(n)}(f_i) = \sum_{j=1}^{N} \tilde{p}(x_j) \sum_{c \in C} P^{(n)}(c|x_j) f_i(c, x_j)$$

is the expected value of f_i with respect to the model $P^{(n)}$

The closer the expected value is to the actual value on the training set, the smaller becomes the weight difference $|w_i^{(n+1)} - w_i^{(n)}|$.

Parameter estimation

Example

Classes A and B, observations are $x \in \{a, b\}^*$.

Features:

$$f_1(c, x) = \begin{cases} 1 \text{ if } |x|_a > 0, c = A \\ 0 \text{ otherwise} \end{cases}$$

$$f_2(c, x) = \begin{cases} 1 \text{ if } |x|_b > 0, c = B \\ 0 \text{ otherwise} \end{cases}$$

Training data:

$x_1 = aa$	$c_1 = A$	$\tilde{p}(x_1) = \frac{1}{4}$
$x_2 = ab$	$c_2 = A$	$\tilde{p}(x_2) = \frac{1}{4}$
$x_3 = ba$	$c_3 = A$	$\tilde{p}(x_3) = \frac{1}{4}$
$x_4 = bb$	$c_4 = B$	$\tilde{p}(x_4) = \frac{1}{4}$

$$m = 1,\ \frac{1}{4}\sum_{j=1}^{4} f_1(c_j, x_j) = \frac{3}{4} \text{ and } \frac{1}{4}\sum_{j=1}^{4} f_2(c_j, x_j) = \frac{1}{4}$$

$$w_1^{(0)} = w_2^{(0)} = 1$$

$$w_1^{(1)} = 1 \cdot \frac{\frac{3}{4}}{0.25(P^{(0)}(A|x_1)+P^{(0)}(A|x_2)+P^{(0)}(A|x_3))} = \frac{3}{\frac{e}{e+1}+\frac{e}{e+e}+\frac{e}{e+e}} = 1.73$$

$$w_2^{(1)} = 1 \cdot \frac{0.25}{0.25(P^{(0)}(B|x_2)+P^{(0)}(B|x_3)+P^{(0)}(B|x_4))} = \frac{1}{\frac{e}{e+e}+\frac{e}{e+e}+\frac{e}{e+1}} = 0.58$$

Parameter estimation

Example continued

$$w_1^{(0)} = w_2^{(0)} = 1$$

$$w_1^{(1)} = 1 \cdot \frac{\frac{3}{4}}{0.25(P^{(0)}(A|x_1)+P^{(0)}(A|x_2)+P^{(0)}(A|x_3))} = \frac{3}{\frac{e}{e+1}+\frac{e}{e+e}+\frac{e}{e+e}} = \frac{3e+3}{2e+1} = 1.73$$

$$w_2^{(1)} = 1 \cdot \frac{0.25}{0.25(P^{(0)}(B|x_2)+P^{(0)}(B|x_3)+P^{(0)}(B|x_4))} = \frac{1}{\frac{e}{e+e}+\frac{e}{e+e}+\frac{e}{e+1}} = \frac{e+1}{2e+1} = 0.58$$

$$w_1^{(2)} = \frac{3}{\frac{e^{1.73}}{e^{1.73}+1}+\frac{e^{1.73}}{e^{1.73}+e^{0.58}}+\frac{e^{1.73}}{e^{1.73}+e^{0.58}}} = \frac{3}{\frac{5.64}{6.64}+\frac{5.64}{7.43}+\frac{5.64}{7.43}} = 1.27$$

$$w_2^{(2)} = \frac{1}{\frac{e^{0.58}}{e^{0.58}+e^{1.73}}+\frac{e^{0.58}}{e^{0.58}+e^{1.73}}+\frac{e^{0.58}}{e^{0.58}+1}} = \frac{1}{\frac{1.79}{7.43}+\frac{1.79}{7.43}+\frac{1.79}{2.79}} = 0.89$$

$$w_1^{(3)} = \frac{3}{\frac{e^{1.27}}{e^{1.27}+1}+\frac{e^{1.27}}{e^{1.27}+e^{0.89}}+\frac{e^{1.27}}{e^{1.27}+e^{0.89}}} = \frac{3}{\frac{3.56}{4.56}+\frac{3.56}{6.00}+\frac{3.56}{6.00}} = 1.52$$

$$w_2^{(3)} = \frac{1}{\frac{e^{0.89}}{e^{0.89}+e^{1.27}}+\frac{e^{0.89}}{e^{0.89}+e^{1.27}}+\frac{e^{0.89}}{e^{0.89}+1}} = \frac{1}{\frac{2.44}{6.00}+\frac{2.44}{6.00}+\frac{2.44}{3.44}} = 0.66$$

Applications

MaxEnt is a very flexible classifier that can be applied to a lot of different problems. Applications are (among others)

- Named Entity Recognition (Malouf, 2002b)
- POS tagging (Ratnaparkhi, 1998)
- Dependency parsing (Hall, 2007)

References

Berger, Adam L., Vincent J. Della Pietra & Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.* 22(1). 39–71. http://dl.acm.org/citation.cfm?id=234285.234289.

Hall, Keith. 2007. K-best spanning tree parsing. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, 392–399. Prague, Czech Republic: Association for Computational Linguistics. http://www.aclweb.org/anthology/P07-1050.

Jurafsky, Daniel & James H. Martin. 2015. Speech and language processing. an introduction to natural language processing, computational linguistics, and speech recognition. Draft of the 3rd edition.

Malouf, Robert. 2002a. A comparison of algorithms for maximum entropy parameter estimation. In *The 6th conference on natural language learning 2002 (conll-2002)*, .

Malouf, Robert. 2002b. Markov models for language-independent named entity recognition. In *The 6th conference on natural language learning 2002 (conll-2002)*, .

Ratnaparkhi, Adwait. 1997. A simple introduction to maximum entropy models for natural language processing. Tech. Rep. 97–08 Institue for Research in Cognitive Science, University of Pennsylvania.

Ratnaparkhi, Adwait. 1998. *Maximum entropy models for natural language ambiguity resolution*: University of Pennsylvania dissertation.