

# NLP: Maximum Entropy Models

AMIT DHOMNE

## 1 Features

Why do we like feature?

- Give us additional, useful information.
- Parts of speech: prefixes, suffixes, capitalization, word shape, is a number, ...

Why do we like sequence models?

- Nouns and Adjectives more likely to follow Determiners
- “I enjoy walks”. Typically, “walks” is a verb, but not here since “enjoy” is definitely a verb.

## 2 Linear Regression

- Given a set of data points, find a line approximating the function that produced the points.
- Equation for a 2D line:  $y = b + mx$
- Each feature  $f_i$  has an associated weight  $w_i$
- So:  $y = w_0 + w_1 f_1$ , where  $y \in (-\infty, \infty)$  is a predicted value
  - $f_1$  is the observation
- With multiple features:  $y = w_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 + \dots = \sum_{i=0}^N w_i \times f_i = \vec{w} \cdot \vec{f}$  (dot product)
  - $\vec{f}$  is the observation (aka  $x$ )
- For a particular instance  $j$ ,  $y_{pred}^{(j)} = \sum_{i=0}^N w_i \times f_i^{(j)}$
- Learning: choose weights  $W$  that minimize the sum-squared error:

$$\hat{w} = \operatorname{argmin}_x \sum_{j=0}^M (y_{pred}^{(j)} - y_{obs}^{(j)})^2$$

- can be solved in closed form

### 3 Logistic Regression

- For many NLP applications, we don't want a real value output, we want a *classification*.
- Moreover, we want to assign a *probability* to each class
- Want to be able to use weighted features
- But, can't simply apply linear regression: it gives us a real value, not a probability

#### Binary Classification

- Need  $p(y = \text{true} \mid x)$ 
  - For observation  $x$  (which is  $\vec{\mathbf{f}}$ ), we want to make use of  $\sum_{i=0}^N w_i \times f_i$ 
    - \* but can't directly because it's a real value
  - We can use a ratio of probabilities,  $\frac{p(y=\text{true}|x)}{p(y=\text{false}|x)} = \frac{p(y=\text{true}|x)}{1-p(y=\text{true}|x)}$ , but this yields a value between 0 (definitely false) and  $\infty$  (definitely true)
  - Logarithm gets us a value between  $-\infty$  and  $\infty$ :  $\ln(\frac{p(y=\text{true}|x)}{1-p(y=\text{true}|x)})$ 
    - \* so we can say this equals  $\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}$ , since it is also in  $(-\infty, \infty)$
  - Exponentiating both sides gives us:  $\frac{p(y=\text{true}|x)}{1-p(y=\text{true}|x)} = e^{\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}}$
  - So,

$$\frac{p(y = \text{true} \mid x)}{1 - p(y = \text{true} \mid x)} = e^{\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}}$$

$$p(y = \text{true} \mid x) = \frac{e^{\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}}}{1 + e^{\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}}} = \frac{1}{1 + e^{-\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}}} \quad (\text{logistic function})$$

- To classify, just use  $\sum_{i=0}^N w_i f_i > 0$

$$p(y = \text{true} \mid x) > p(y = \text{false} \mid x)$$

$$\frac{p(y = \text{true} \mid x)}{p(y = \text{false} \mid x)} > 1$$

$$\frac{p(y = \text{true} \mid x)}{1 - p(y = \text{true} \mid x)} > 1$$

$$e^{\vec{\mathbf{w}} \cdot \vec{\mathbf{f}}} > 1 \quad \text{from above}$$

$$\vec{\mathbf{w}} \cdot \vec{\mathbf{f}} > 0$$

#### Learning

$$\hat{w} = \operatorname{argmax}_w \prod_i p(y^{(i)} \mid x^{(i)})$$

$$= \operatorname{argmax}_w \prod_i \begin{cases} p(y^{(i)} = 1 \mid x^{(i)}) & \text{for } y^{(i)} = 1 \\ p(y^{(i)} = 0 \mid x^{(i)}) & \text{for } y^{(i)} = 0 \end{cases}$$

- convex optimization
- gradient ascent or L-BFGS

## 4 Maximum Entropy

- Multi-class logistic regression
- Generalizing the above: we want the probability of a class  $c$  for some observation  $x$ ,  $p(c | x)$

$$p(c | x) = \frac{\exp\left(\sum_{i=0}^N w_{ci} f_i\right)}{Z} = \frac{\exp\left(\sum_{i=0}^N w_{ci} f_i\right)}{\sum_{c' \in C} p(c' | x)} = \frac{\exp\left(\sum_{i=0}^N w_{ci} f_i\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^N w_{c'i} f_i\right)}$$

- $Z$  is the *normalization factor*
- So, we need to assign a per-class, per-feature weight  $w_{ci}$  for each feature  $f_i$  and class  $c$

### Binary Features

- In NLP, we typically use binary features: “ends in *-ed*”, “starts with capital”, etc
- Indicator function: feature that takes only values 0 or 1
- $f_i(c, x)$ : feature  $i$  for class  $c$  for observation  $x$ :

$$p(c | x) = \frac{\exp\left(\sum_{i=0}^N w_{ci} f_i(c, x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^N w_{c'i} f_i(c', x)\right)}$$

Example: POS prediction based on word (not sequence tagging!)

- He/PRP is/VBZ expected/VBN to/TO **race**/?? tomorrow/
- Indicator function: feature that takes only values 0 or 1
- $f_i(c, x)$ : feature  $i$  for class  $c$  for observation  $x$ :

$f_1(c, x) = 1$ iff $word_i = \text{“race”}$	$\& c = \text{NN}$	(0 otherwise)
$f_2(c, x) = 1$ iff $t_{i-1} = \text{TO}$	$\& c = \text{VB}$	
$f_3(c, x) = 1$ iff $\text{suffix}(word_i) = \text{“ing”}$	$\& c = \text{VBG}$	
$f_4(c, x) = 1$ iff $\text{isLowerCase}(word_i)$	$\& c = \text{VB}$	
$f_5(c, x) = 1$ iff $word_i = \text{“race”}$	$\& c = \text{VB}$	
$f_6(c, x) = 1$ iff $t_{i-1} = \text{TO}$	$\& c = \text{NN}$	

- Each  $f_i(c, x)$  gets a real-valued weight:

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$
VB		0.8		0.01	0.1	
NN	0.8					-1.3

- Now we can compute the relative probabilities (can leave out  $f_i(c, x) = 0$  entries):

$$p(VB | x) = \frac{e^{0.8} e^{0.01} e^{0.1}}{e^{0.8} e^{-1.3} + e^{0.8} e^{0.01} e^{0.1}} = 0.80$$

$$p(NN | x) = \frac{e^{0.8} e^{-1.3}}{e^{0.8} e^{-1.3} + e^{0.8} e^{0.01} e^{0.1}} = 0.20$$

- Features can be arbitrarily complex
  - For example, this feature can be used to recognize that words at the start of the sentence may be capitalized even though they are common nouns.

$$f_{125}(c, x) = 1 \text{ iff } word_{i-1} = \langle S \rangle \ \& \ \text{startsUpper}(word_i) \quad \& \ c = \text{NN}$$

Learning

- $\hat{w} = \operatorname{argmax}_w \sum_i \log p(y^{(i)} | x^{(i)})$
- Regularization is like smoothing:  $\hat{w} = \operatorname{argmax}_w \sum_i \log p(y^{(i)} | x^{(i)}) - \alpha \sum_{j=1}^n w_j^2$ 
  - Reduce the score if the weights are high
  - Penalizes overly-specific weights
  - $\alpha$  controls the amount of regularization (analogous to  $\lambda$  in smoothing)
  - Assume a Gaussian prior on the weights, saying they prefer to have a value 0

## 5 MaxEnt Markov Model

- Discriminative Model: discriminate between various possible tag sequences
- use  $o$  for “observation” (word) since  $w$  means “weight”
- Single probabilistic model to estimate  $p(t_i | o_i, t_{i-1})$  instead of two separate models like HMM

Recall for HMM:

- Likelihood of sequence:  $p(t_1 \dots t_n | o_1 \dots o_n) = \prod_{i=1}^n p(o_i | t_i) \cdot p(t_i | t_{i-1})$
- $\hat{t}_i^n = \operatorname{argmax}_{t_1^n} p(t_1 \dots t_n | o_1 \dots o_n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n p(o_i | t_i) \cdot p(t_i | t_{i-1})$
- $v_i(k) = p(w_i | k) \cdot \max_{k' \in T} v_{i-1}(k') \cdot p(k | k')$

For an MEMM:

- $p(t_i | o_i, t_{i-1}) = \frac{1}{Z(t_{i-1}, o_i)} \exp(\sum_i w_i f_i(t_i, o_i))$
- Likelihood of sequence:  $p(t_1 \dots t_n | o_1 \dots o_n) = \prod_{i=1}^n p(t_i | o_i, t_{i-1})$
- $\hat{t}_i^n = \operatorname{argmax}_{t_1^n} p(t_1 \dots t_n | o_1 \dots o_n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n p(t_i | o_i, t_{i-1})$
- $v_i(k) = \max_{k' \in T} v_{i-1}(k') \cdot p(k | o_i, k')$

“United States President Barack Obama discussed changes in the U.S. economy. Obama presented his ideas to Congress.”

- ### Identifying sequence spans

- ## Pipelines

- 5