Structured Perceptron

October 6, 2015

Two Kinds of Models

Generative

$$p(\boldsymbol{x}, \boldsymbol{y})$$

Two Kinds of Models

Generative

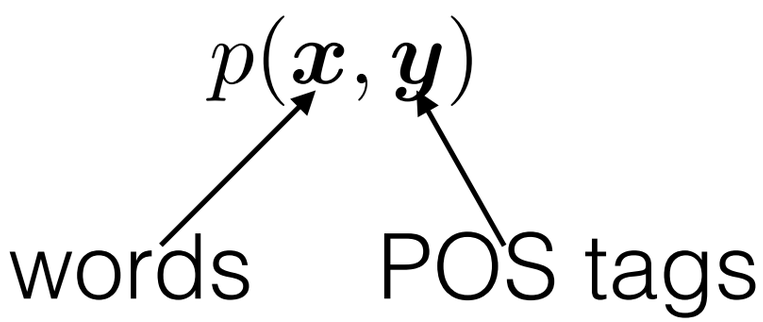
$$p(\boldsymbol{x}, \boldsymbol{y})$$

Discriminative

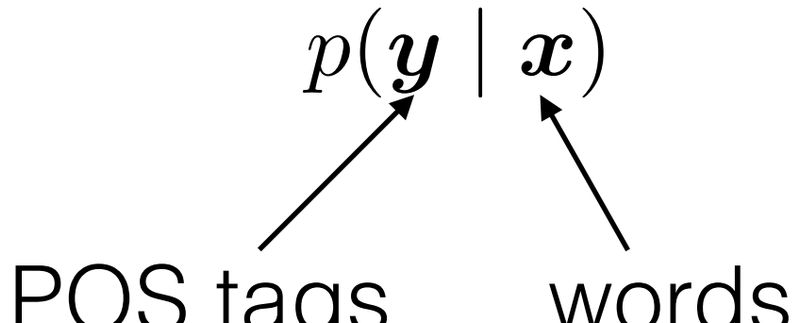
$$p(\boldsymbol{y} \mid \boldsymbol{x})$$

Two Kinds of Models

Generative



Discriminative



Two Kinds of Models

Generative

$p(\boldsymbol{x}, \boldsymbol{y})$

words POS tags

Discriminative

$p(\boldsymbol{y} \mid \boldsymbol{x})$

POS tags words

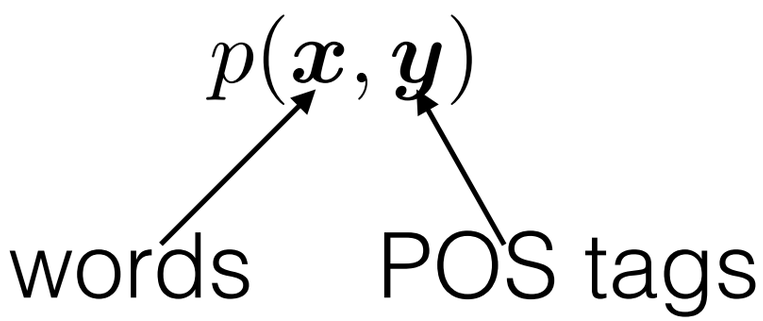
Things we can compute:

$p(\boldsymbol{x})$ marginal probability
good for EM

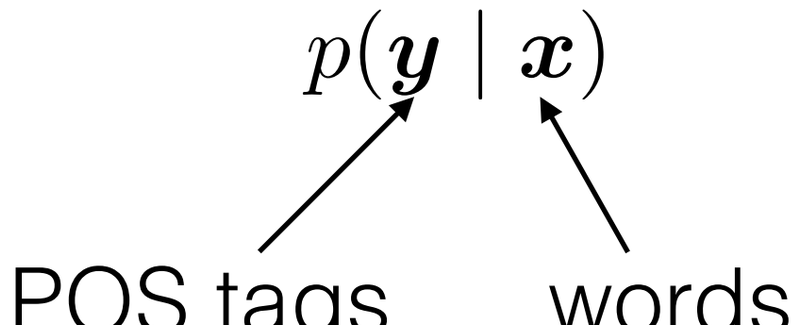
$p(\boldsymbol{y} \mid \boldsymbol{x})$ classification

Two Kinds of Models

Generative



Discriminative



Things we can compute:

$p(\boldsymbol{x})$ marginal probability
good for EM

$p(\boldsymbol{y} \mid \boldsymbol{x})$ classification

~~$p(\boldsymbol{x})$ marginal probability~~
~~good for EM~~

$p(\boldsymbol{y} \mid \boldsymbol{x})$ classification

Why Discriminative?

- Often smaller modeling space - simpler inference
- Model what you care about (classification)
- Learn with features to encode **knowledge** about the problem

Why Features?

- Proper nouns (PNP) often start with Capital Letters
- Words that end in `-ly` are usually RB
- Words that end in `-ing` are usually VBG
- Words that end in `-ed` are usually VBD
- Words that match the regular expression `(0|1|2|3|4|5|6|7|8|9)*` are usually CD

Why Features?

- Locations (LOC) are often in knowledge bases (e.g., Wikipedia, DBPedia, etc.)

- Persons (PER) often contain words that are in lists of names (e.g., the US Social Security List of Popular Names)

- …

Why Features?

- Look at the predictions that a classifier is making.
- Can you identify a pattern in the errors?
    - Then you can probably engineer a feature to fix the problem!

Viterbi with Features

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \prod_{i=1}^{n} (p(y_i \mid y_{i-1}) \times p(x_i \mid y_i))$$

Viterbi with Features

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \prod_{i=1}^{n} (p(y_i \mid y_{i-1}) \times p(x_i \mid y_i))$$

$$= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{n} (\log p(y_i \mid y_{i-1}) + \log p(x_i \mid y_i))$$

Viterbi with Features

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \prod_{i=1}^{n} \left(p(y_i \mid y_{i-1}) \times p(x_i \mid y_i) \right)$$

$$= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{n} \left(\log p(y_i \mid y_{i-1}) + \log p(x_i \mid y_i) \right)$$

$$= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(x_i, y_{i-1}, y_i)$$

with **w** and **f** appropriately defined.

Viterbi with Features

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \prod_{i=1}^{n} \left(p(y_i \mid y_{i-1}) \times p(x_i \mid y_i) \right)$$

$$= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{n} \left(\log p(y_i \mid y_{i-1}) + \log p(x_i \mid y_i) \right)$$

$$= \arg\max_{\boldsymbol{y}} \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(x_i, y_{i-1}, y_i)$$

with **w** and **f** appropriately defined.

But: Viterbi works with any **w** and **f**

Viterbi with Features

$\mathbf{f} : \Sigma \times \Omega \times \Omega \rightarrow \mathbb{R}^d$ feature functions

$\mathbf{w} \in \mathbb{R}^d$ weight vector

Viterbi with Features

$\mathbf{f} : \Sigma \times \Omega \times \Omega \rightarrow \mathbb{R}^d$ feature functions

$\mathbf{w} \in \mathbb{R}^d$ weight vector

$$f_{5123}(x_i, y_i, y_{i-1}) = \begin{cases} 1 & \text{if } x_i \text{ ends with -ly } \wedge \ y_i = \texttt{RB} \\ 0 & \text{otherwise} \end{cases}$$

Viterbi with Features

$\mathbf{f} : \Sigma \times \Omega \times \Omega \rightarrow \mathbb{R}^d$ feature functions

$\mathbf{w} \in \mathbb{R}^d$ weight vector

$$f_{5123}(x_i, y_i, y_{i-1}) = \begin{cases} 1 & \text{if } x_i \text{ ends with -ly } \wedge \ y_i = \texttt{RB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{12048}(x_i, y_i, y_{i-1}) = \begin{cases} 1 & \text{if } y_i = \texttt{VBD } \wedge \ y_{i-1} = \texttt{RB} \\ 0 & \text{otherwise} \end{cases}$$

Weights?

- We know where features come from - we write down the things we believe (hope?) to be correlated with good predictions

- **Where do weights come from?**

  - We will learn these from data

  - Intuitively, we want the score of the “right answer” to be higher than the score of all other answers

Structured Perceptron

- Initialize the weight vector $\mathbf{w}=\mathbf{0}$
  - Pick an example from the training data
    - Run the Viterbi algorithm to get $\boldsymbol{y}^-$
    - Is it correct? Go pick another example and repeat.
    - $\mathbf{w} \leftarrow \mathbf{w} + \sum_{i=1}^{n} \mathbf{f}(x_i, y_i^+, y_{i-1}^+) - \sum_{i=1}^{n} \mathbf{f}(x_i, y_i^-, y_{i-1}^-)$
  - Go pick another example and repeat.

How does it work?

- Intuitively, when you make a mistake, the algorithm updates **w** so that the score of the mistake is *lower* and the score of the correct answer is *higher*

- Does the algorithm converge?
  - Yes! … If there is a setting of **w** that makes all answers correct
  - If not, it will oscillate
  - In practice
    - stop after a number of iterations
    - better: return the average of **w** across iterations

Another View

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(x_i, y_i, y_{i-1})$$

$$\mathcal{L} = \max\{0, score(\boldsymbol{x}, \hat{\boldsymbol{y}}) - score(\boldsymbol{x}, \boldsymbol{y}^*)\}$$

Minimize the loss $\mathcal{L}$ by improving **w**

How do we minimize a function?
Differentiate and take a step in the opposite direction of the gradient.

$$\mathcal{L} = \max\{0, \gamma + score(\boldsymbol{x}, \hat{\boldsymbol{y}}) - score(\boldsymbol{x}, \boldsymbol{y}^*)\}$$

a margin says score must be better by a certain amount or it counts as "wrong"

Structured?

- The **classic** perceptron makes classification decisions

- The **structured** perceptron produces outputs that consist of smaller pieces (e.g., strings and trees)