Structured Perceptrons

CS114B Lab 8

Kenneth Lai

March 24, 2022

Structured Perceptrons

- Perceptrons for sequence labeling

Structured Perceptrons

- Perceptrons for sequence labeling
  - Given a sequence $X = [x_1, \ldots, x_T]$, predict labels $Y = [y_1, \ldots, y_T]$

Structured Perceptrons

- Perceptrons for sequence labeling
  - Given a sequence $X = [x_1, \ldots, x_T]$, predict labels $Y = [y_1, \ldots, y_T]$
  - We do not care about the probability $P(Y|X)$, just which Y has the highest score Z

Structured Perceptrons

- Perceptrons for sequence labeling
  - Given a sequence $X = [x_1, \ldots, x_T]$, predict labels $Y = [y_1, \ldots, y_T]$
  - We do not care about the probability $P(Y|X)$, just which Y has the highest score Z
- $\hat{Y} = \operatorname*{argmax}_{k \in K^T} Z_k$

Structured Perceptrons

- Score Z decomposes into a sum of local parts

Structured Perceptrons

- Score Z decomposes into a sum of local parts
  - At each time step i, for each possible combination of current tag y_i and previous tag y_{i-1}, compute a local score $z(y_i, y_{i-1})$

Structured Perceptrons

- Score Z decomposes into a sum of local parts
  - At each time step i, for each possible combination of current tag y_i and previous tag y_{i-1}, compute a local score $z(y_i, y_{i-1})$
  - Use the Viterbi algorithm to combine the local scores across the sequence, and find the argmax

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:
  - Previous tag y_{i-1}

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:
  - Previous tag y_{i-1}
    - At the beginning of the sentence, let the previous tag y_{i-1} be the start symbol `<S>`

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:
  - Previous tag y_{i-1}
    - At the beginning of the sentence, let the previous tag y_{i-1} be the start symbol <S>
  - Current word x_i

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:
  - Previous tag y_{i-1}
    - At the beginning of the sentence, let the previous tag y_{i-1} be the start symbol `<S>`
  - Current word x_i
- For simplicity, we will assume that these are the only features, and we will ignore the bias term

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:
  - Previous tag y_{i-1}
    - At the beginning of the sentence, let the previous tag y_{i-1} be the start symbol `<S>`
  - Current word x_i
- For simplicity, we will assume that these are the only features, and we will ignore the bias term
- Let $\mathbf{f}(X, y_i, y_{i-1}, i)$ be the feature vector at time step i

Structured Perceptrons

- Suppose that at each time step i, we want to predict the current tag y_i using the following features:
  - Previous tag y_{i-1}
    - At the beginning of the sentence, let the previous tag y_{i-1} be the start symbol `<S>`
  - Current word x_i
- For simplicity, we will assume that these are the only features, and we will ignore the bias term
- Let $\mathbf{f}(X, y_i, y_{i-1}, i)$ be the feature vector at time step i
  - Using $\mathbf{f}$ instead of $\mathbf{x}$, because features can include more than just the input

Structured Perceptrons

- We can arrange our weight matrix Θ as follows:

$$\begin{bmatrix} \rule{2cm}{0.4pt} \\ \\ \\ \rule{2cm}{0.4pt} \\ \\ \\ \end{bmatrix}$$

Structured Perceptrons

- We can arrange our weight matrix Θ as follows:

$$\left[\begin{array}{c} \pi \\ \hline \\ \\ \hline \\ \\ \end{array}\right]$$

- Initial features
  - $y_{i-1} = \text{<S>}, y_i = \ldots$

Structured Perceptrons

- We can arrange our weight matrix Θ as follows:

$$\left[\begin{array}{c} \pi \\ \hline \\ \mathbf{A} \\ \\ \hline \\ \\ \\ \end{array}\right]$$

- Initial features
  - $y_{i-1} = \texttt{<S>}, y_i = \ldots$
- Transition features
  - $y_{i-1} = \ldots, y_i = \ldots$

Structured Perceptrons

- We can arrange our weight matrix Θ as follows:

$$\left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right]$$

- Initial features
  - $y_{i-1} = \texttt{<S>}, y_i = \ldots$
- Transition features
  - $y_{i-1} = \ldots, y_i = \ldots$
- Emission features
  - $x_i = \ldots, y_i = \ldots$

Initialization Step

- We want to compute local scores $z(y_1, \texttt{<S>})$ for each possible y_1

Initialization Step

- We want to compute local scores $z(y_1, \texttt{<S>})$ for each possible y_1
  - These are the elements of $\mathbf{z}_1 = \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta$

Initialization Step

$$\mathbf{z}_1 = \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta$$

$$= \left[\quad \middle| \qquad \middle| \qquad\qquad \right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array} \right]$$

Initialization Step

$$\mathbf{z}_1 = \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta$$

$$= \left[\ 1 \ \middle| \qquad \middle| \qquad\qquad \right] \cdot \begin{bmatrix} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{bmatrix}$$

- We know that $y_{i-1} = \texttt{<S>}$

Initialization Step

$$\mathbf{z}_1 = \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta$$

$$= \left[\; 1 \;\middle|\; \mathbf{0} \;\middle|\; \qquad\qquad \right] \cdot \begin{bmatrix} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{bmatrix}$$

- We know that y_{i-1} cannot be any other tag

Initialization Step

$$
\begin{aligned}
\mathbf{z}_1 &= \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta \\
&= \left[\begin{array}{c|c|c} 1 & \mathbf{0} & \color{blue}{\mathbf{1}\{x_1 = o_1\}} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right]
\end{aligned}
$$

- One-hot vector of the first word

Initialization Step

$$
\begin{aligned}
\mathbf{z}_1 &= \mathbf{f}(X, y_1, \text{<S>}, 1) \cdot \Theta \\
&= \left[\begin{array}{c|c|c} 1 & \mathbf{0} & \mathbf{1}\{x_1 = o_1\} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right] \\
&= 1 \cdot \pi + \mathbf{0} \cdot \mathbf{A} + \mathbf{1}\{x_1 = o_1\} \cdot \mathbf{B}
\end{aligned}
$$

Initialization Step

$$
\begin{aligned}
\mathbf{z}_1 &= \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta \\
&= \left[\begin{array}{c|c|c} 1 & \mathbf{0} & \mathbf{1}\{x_1 = o_1\} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right] \\
&= 1 \cdot \pi + \mathbf{0} \cdot \mathbf{A} + \mathbf{1}\{x_1 = o_1\} \cdot \mathbf{B} \\
&= \pi + \mathbf{B}[o_1]
\end{aligned}
$$

Initialization Step

$$
\begin{aligned}
\mathbf{z}_1 &= \mathbf{f}(X, y_1, \texttt{<S>}, 1) \cdot \Theta \\
&= \left[\begin{array}{c|c|c} 1 & \mathbf{0} & \mathbf{1}\{x_1 = o_1\} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right] \\
&= 1 \cdot \pi + \mathbf{0} \cdot \mathbf{A} + \mathbf{1}\{x_1 = o_1\} \cdot \mathbf{B} \\
&= \pi + \mathbf{B}[o_1]
\end{aligned}
$$

- These local scores go into the first column of the Viterbi trellis

Recursion Step

- We want to compute local scores $z(y_i, y_{i-1})$ for each possible combination of y_i and y_{i-1}

Recursion Step

- We want to compute local scores $z(y_i, y_{i-1})$ for each possible combination of y_i and y_{i-1}
  - We can stack the feature vectors for each possible y_{i-1}, $\mathbf{f}(X, y_i, y_{i-1}, i)$, on top of each other, in order

Recursion Step

- We want to compute local scores $z(y_i, y_{i-1})$ for each possible combination of y_i and y_{i-1}
  - We can stack the feature vectors for each possible y_{i-1}, $\mathbf{f}(X, y_i, y_{i-1}, i)$, on top of each other, in order
    - Form a feature matrix $\mathbf{F}_i$

Recursion Step

- We want to compute local scores $z(y_i, y_{i-1})$ for each possible combination of y_i and y_{i-1}
  - We can stack the feature vectors for each possible y_{i-1}, $\mathbf{f}(X, y_i, y_{i-1}, i)$, on top of each other, in order
    - Form a feature matrix $\mathbf{F}_i$
  - Compute $\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$

Recursion Step

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$$

$$= \left[\begin{array}{c|c|c} \quad & \quad & \qquad \end{array} \right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array} \right]$$

Recursion Step

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$$

$$= \left[\begin{array}{c|c|c} \mathbf{0} & \quad & \quad \end{array} \right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array} \right]$$

- We know that $y_{i-1} \neq$ <S>

Recursion Step

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$$

$$= \left[\begin{array}{c|c|c} \mathbf{0} & \mathbf{I} & \phantom{\mathbf{0000000000}} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right]$$

- Identity matrix!

Recursion Step

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$$

$$= \left[\begin{array}{c|c|c} \mathbf{0} & \mathbf{I} & \mathbf{1}\{x_i = o_i\} \end{array} \right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array} \right]$$

- Stack of one-hot vectors

Recursion Step

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$$

$$= \left[\begin{array}{c|c|c} \mathbf{0} & \mathbf{I} & \mathbf{1}\{x_i = o_i\} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right]$$

$$= 0 \cdot \pi + \mathbf{I} \cdot \mathbf{A} + \mathbf{1}\{x_i = o_i\} \cdot \mathbf{B}$$

Recursion Step

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta$$

$$= \left[\begin{array}{c|c|c} \mathbf{0} & \mathbf{I} & \mathbf{1}\{x_i = o_i\} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right]$$

$$= 0 \cdot \pi + \mathbf{I} \cdot \mathbf{A} + \mathbf{1}\{x_i = o_i\} \cdot \mathbf{B}$$

$$= \mathbf{A} + \mathbf{B}[o_i]$$

Recursion Step

$$
\begin{aligned}
\mathbf{Z}_i &= \mathbf{F}_i \cdot \Theta \\
&= \left[\begin{array}{c|c|c} \mathbf{0} & \mathbf{I} & \mathbf{1}\{x_i = o_i\} \end{array}\right] \cdot \left[\begin{array}{c} \pi \\ \hline \mathbf{A} \\ \hline \mathbf{B} \end{array}\right] \\
&= 0 \cdot \pi + \mathbf{I} \cdot \mathbf{A} + \mathbf{1}\{x_i = o_i\} \cdot \mathbf{B} \\
&= \mathbf{A} + \mathbf{B}[o_i]
\end{aligned}
$$

- Use the Viterbi algorithm to combine these local scores with scores from the rest of the sequence

Recursion Step

- Use the Viterbi algorithm to combine these local scores with scores from the rest of the sequence

Recursion Step

- Use the Viterbi algorithm to combine these local scores with scores from the rest of the sequence

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta = \mathbf{A} + \mathbf{B}[o_i]$$

Recursion Step

- Use the Viterbi algorithm to combine these local scores with scores from the rest of the sequence

$$\mathbf{Z}_i = \mathbf{F}_i \cdot \Theta = \mathbf{A} + \mathbf{B}[o_i]$$

$$\text{Viterbi}[:, i] = \max(\text{Viterbi}[:, i-1:i] + \mathbf{A} + \mathbf{B}[o_i], \text{axis}{=}0)$$

Termination Step

- Best last tag is the argmax of the last column of the Viterbi trellis

Termination Step

- Best last tag is the argmax of the last column of the Viterbi trellis
- Follow backpointers in the backpointer trellis

Termination Step

- Best last tag is the argmax of the last column of the Viterbi trellis
- Follow backpointers in the backpointer trellis
  - Nothing fancy; we'll let you figure it out on your own

Termination Step

- Best last tag is the argmax of the last column of the Viterbi trellis
- Follow backpointers in the backpointer trellis
  - Nothing fancy; we'll let you figure it out on your own
  - For HW4, you do not have to return the path (log-)probability/score, just the backtrace path

Structured Perceptron Learning Algorithm

- Use the Viterbi algorithm to compute the best tag sequence

Structured Perceptron Learning Algorithm

- Use the Viterbi algorithm to compute the best tag sequence
- If $\hat{Y} = Y$, then do nothing

Structured Perceptron Learning Algorithm

- Use the Viterbi algorithm to compute the best tag sequence
- If $\hat{Y} = Y$, then do nothing
- Else:
  - Increment weights for features in Y, decrement weights for features in $\hat{Y}$

Structured Perceptron Learning Algorithm

- Use the Viterbi algorithm to compute the best tag sequence
- If $\hat{Y} = Y$, then do nothing
- Else:
  - Increment weights for features in Y, decrement weights for features in $\hat{Y}$
  - In other words, for each time step i:

Structured Perceptron Learning Algorithm

- Use the Viterbi algorithm to compute the best tag sequence
- If $\hat{Y} = Y$, then do nothing
- Else:
  - Increment weights for features in Y, decrement weights for features in $\hat{Y}$
  - In other words, for each time step i:
    - Increment weights for features in y_i, decrement weights for features in $\hat{y}_i$

Structured Perceptron Learning Algorithm

- Use the Viterbi algorithm to compute the best tag sequence
- If $\hat{Y} = Y$, then do nothing
- Else:
  - Increment weights for features in Y, decrement weights for features in $\hat{Y}$
  - In other words, for each time step i:
    - Increment weights for features in y_i, decrement weights for features in $\hat{y}_i$
  - Nothing fancy; no Numpy tricks needed