Conditional Random Fields

- The **Conditional Random Field** i bilistic model for sequence labeling based on logistic r
- The name is der **elds**, a class of models in whic bles is proportional to a product of scores acros in a factor graph.

The probability model

- The basic probab

$$P(\boldsymbol{y}|\boldsymbol{w}) = \frac{\exp(\Psi(\boldsymbol{w}, \boldsymbol{y}))}{\sum_{\boldsymbol{y'} \in \mathcal{Y}(\boldsymbol{w})}}$$

- This is almost identical to Logistic Regressio label space is se

- This requires *t tag sequence* in decoding, and for *summing ov* in training

- The usual locality assumption on the scoring function:

$$\Psi(\boldsymbol{w}, \boldsymbol{y}) = \sum_{m=1}^{M+1} \psi(\boldsymbol{w}, y_m, y_{m-1}, m)$$

Decoding in CRFs

- The Viterbi algori sequence, just as it can be used for decoding HMM and Perpcetron models
- The decoding algorithm is identical to that of p because the denominator is the same for all tag s. can thus be ignored:

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \text{a} \ldots_{\boldsymbol{y}} \\
&= \operatorname*{argmax}_{\boldsymbol{y}} \Psi(\boldsymbol{y}, \boldsymbol{w}) - \text{lo} \ldots_{\boldsymbol{y'} \in \mathcal{Y}(\boldsymbol{w})} \ldots (\boldsymbol{y'}, \boldsymbol{w})) \\
&= \operatorname*{argmax}_{\boldsymbol{y}} \Psi(\boldsymbol{y}, \boldsymbol{w}) \\
&= \operatorname*{argmax}_{\boldsymbol{y}} \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})
\end{aligned}
$$

Learning in CRFs

- As with logistic regression, the weights can be learned by minimizing the regularized negative log-p

$$\mathcal{L} = \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 - \sum_{i=1}^{M} \log p(\boldsymbol{y}^{(i)} \mid \boldsymbol{w}^{(i)}; \boldsymbol{\theta})$$

$$= \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 - \sum_{i=1}^{M} \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}) + \log \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{w}^{(i)})} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y'})\right)$$

- The second term requires computing *the sum of all possible labelings*. There are $|\mathcal{Y}|^M$ possible labeling for an input of length M, so an efficient algorithm is required to compute this sum.

Computing the gradients of the loss function

- As in logistic regression, the gradient of the negative log likelihood with respect to the parameters is the difference between observed and expected feature co

$$\frac{\partial}{\partial} \quad {}^{N} \quad {}^{(i)}, \boldsymbol{y}^{(i)})$$

- Computing this gradient involves com r of a sequence that includes the transition $_{m-1}$ Y_m
- Recall the feature function for bigram tag sequences is of the form $f(Y_{m-1}, Y_m, \boldsymbol{w}, m)$. To compute the expected count of a feature, we need $P(Y_{m-1} = k', Y_m = k | \boldsymbol{w})$.

Marginal probabilities over tag bigrams

$$Pr(Y_{m-1} = k', Y_m = k \mid \boldsymbol{w}) = \frac{\sum_{\boldsymbol{y}: Y_m = k, Y_{m-1} = k'} \prod_{n=1}^{M} \exp s_n(y_n, y_{n-1})}{\sum_{\boldsymbol{y'}} \prod_{n=1}^{M} \exp s_n(y'_n, y'_{n-1})}$$

How do we compute t

- The naive way to compute this probabilit possible labelings for the sequence. Sinc $\mathcal{Y}^M$ possible labelings, this is prohibitively expensive for a typical tag set and sentence length.
- So we need find a more efficient way of doing this.

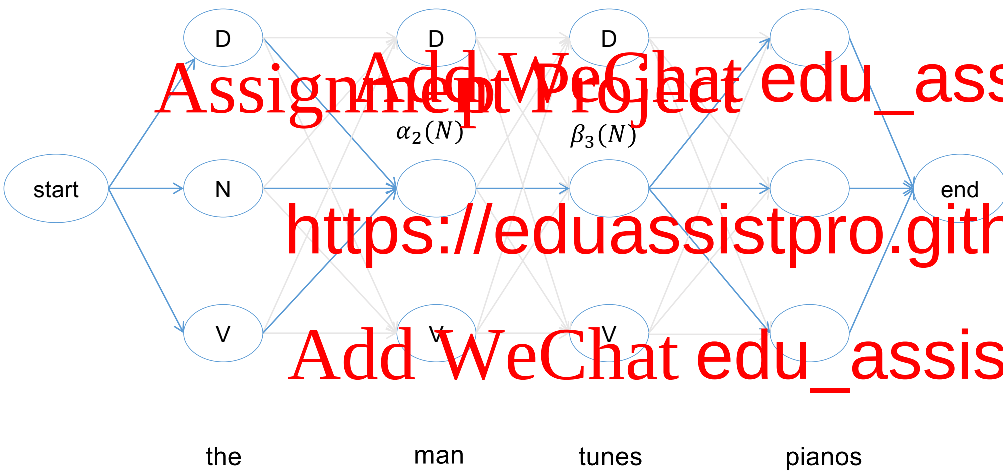
Computing the numerator

The numerator sums over all tag sequences that includes the transition $(Y_{m-1} = k') \to (Y_m = k)$. This sum can be decomposed into three parts: the prefix $\boldsymbol{y}_{1:m-1}$, terminating in $Y_{m-1} = k'$; the transition $(Y_{m-1} = k') \to (Y_m = k)$; and the suffix $\boldsymbol{y}_{m:M}$, beginning with the tag $Y_m = k$:

$$
\begin{aligned}
\sum_{\boldsymbol{y}:Y_m=k,Y_{m-1}=k'} \prod_{n=1}^{M+1} \exp s_n(y_n, y_{n-1}) = & \sum_{\boldsymbol{y}_{1:m-1}:Y_{m-1}=k'} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1}) \\
& \times \exp s_m(k, k') \\
& \times \sum_{\boldsymbol{y}_{m:M}:Y_m=k} \prod_{n=m+1}^{M+1} \exp s_n(y_n, y_{n-1})
\end{aligned}
$$

Trellis

We can illustrate the nu



We compute the numerator by positing the first term as a **forward variable** $\alpha_{m-1}(k')$, and the third term as a **backward variable** $\beta_m(k)$.

Defining a forward variable that can be cached

- A **forward variables** $\alpha_m(y_m)$ is equal to the sum of scores of all paths leading to the tag y_m at position m:

$$\alpha_m(y_m) \triangleq \quad \exp \quad {}_{n} \quad {}_{n} \quad {}_{n-1}$$

$$= \quad {}_{y_{1:m-1}} \quad {}_{n} \quad {}_{1})$$

- The forward recurrence is also known as the **sum-product algorithm** and can be computed through a recurrence

The forward recurrence

- Computing the forward variable at position m

$$
\begin{aligned}
\alpha_m(y_m) &= \sum_{y_1 \ldots y_{m-1}} \prod_{n=1}^{m} \exp s_n(y_n, y_{n-1}) \\
&= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \sum_{y_1 \ldots y_{m-2}} \prod_{n=1}^{m-1} \exp s_n(y_n, y_{n-1}) \\
&= \sum_{y_{m-1}} (\exp s_m(y_m, y_{m-1})) \times \alpha_{m-1}(y_{m-1})
\end{aligned}
$$

The backward recurrence

$$
\begin{aligned}
\beta_m(k) &\triangleq \sum_{\boldsymbol{y}_{m:M}: y_m = k} \prod_{n=m}^{M+1} \exp s_n(y_n, y_{n-1}) \\
&= \sum_{k' \in \mathcal{Y}} \sum_{\boldsymbol{y}_{m+1:M}: Y_{m+1}=k'} \prod_{n=m}^{M+1} \exp s_n(y_n, y_{n-1}) \\
&= \sum_{k' \in \mathcal{Y}} \exp s_m(k', k) \times \beta_{m+1}
\end{aligned}
$$

where k' is the label at position $m + 1$.

Computing the denominator

The denominator, the s

sequence, can be compu
backward recurrence, or at any given position m:

- The score of all possible labelings for the entire sequence is the value of the forward variable at position in state $\blacklozenge$:

$$\sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{w})} \Psi(\boldsymbol{w}, \boldsymbol{y}) = \alpha \quad (\ )$$

$$= \sum_{\boldsymbol{y}_{1:M}} (\exp s_{M+1} \quad \mathrm{p}\, s_m(y_m, y_{m-1})$$

- It can also be computed via backward recurrence as the value of the backward variable at position 0:

$$\sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{w})} \Psi(\boldsymbol{w}, \boldsymbol{y}) = \beta_0(\lozenge)$$

Features and their weights

f_k	y_{m-1}	y_m	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
f_1	D	D	-	-	-	-	-	-0.5
f_2	N	D	-	-	-	-	-	-0.8
f_3	V	D	-	-	-			
f_4	D	N	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
f_5	N	N	-	-	-			
f_6	V	N						3
f_7	D	V	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.4
f_8	N	V	-	-	-			
f_9	V	V	-	-	-			6
f_{10}	-	D	-	-	man	-	-	-0.5
f_{11}	-	N	-	-	man	-	-	2
f_{12}	-	V	-	-	man	-	-	1
f_{13}	-	D	-	the	-	-	-	-4
f_{14}	-	N	-	the	-	-	-	5
f_{15}	-	V	-	the	-	-	-	-2

Computing local transition matricies

Recall the local score bet

$\exp s_m(y_m, y_{m-1})$. The transition scores from each tag y_{m-1} to the tag y_m can be arranged in a $K \times K$ matrix, where K is the number of tags. Suppose we are computing the score at the pos "man" based on the feature weights in the previous sl

$$\exp(f_1\theta_1 + f \quad - 4) = 0.007$$

$$\begin{array}{c c} & \begin{array}{ccc} D & N & V \end{array} \\ \begin{array}{c} D \\ N \\ V \end{array} & \begin{bmatrix} 0.007 & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} \end{array}$$

Forward computation in CRF

$M_1(k',k)$

	D	N	V
D	1.2	0	0
N	0	2	0
V	0	0	0.5

M

	D	N	V
D	0.2	4	0.1
N	0.3	0.8	2.5
V	2	5	1

	D	N	V
D	0.3	4.5	0.8
N	0.4	1.2	3.5
V	3	6	3

	D	N	V
D	0.5	5	0.4
N	0.6	1.5	6

$M_5(k',k)$

	D	N	V
D	1.2	1.2	1.2
N	0.3	0.3	0.3
	4	4	4

Representing features as matricies

Some features span mul

$$f_1(k', k, \quad, m) = 1 \text{ iff } w_m = run \;\;\&\;\; k = V$$

$$f_2(k', k, \boldsymbol{w}, m) = 1 \text{ iff } w_m = run \;\;\&\;\; k = N$$

If represented as matricies, these features have mu
have a value of 1:

$$f_1(k', k, \boldsymbol{w}, m) = \begin{array}{c|ccc} & & & \\ D & & & \\ N & & & \\ V & 0 & 0 & 1 \end{array}$$

$$f_2(k', k, \boldsymbol{w}, m) = \begin{array}{c} \\ D \\ N \\ V \end{array}\begin{array}{c} \begin{array}{ccc} D & N & V \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

Feature expectations

- Let f be any loc… m). The count of the this feature in a particular sequence is

$$F(\boldsymbol{y}, \boldsymbol{w}) = \sum_{m}^{M+1} f(\ldots$$

- Expectatio… quence is:

$$\mathbb{E}[F(\boldsymbol{w}, \boldsymbol{y} \ldots {}_{m-1} \ldots f(k', k, \boldsymbol{w}, m)$$

$$= \sum_{m} \frac{\alpha_{m-1}(f \odot \ldots_{m} \ldots_{m}}{Z(\boldsymbol{w})}$$

where $Z(\boldsymbol{w})$ is the total score of all labelings, also known as the **partition function**.

- Given the feature expectations, we can now compute the gradient of each feature.

Example computation of feature expectations

- Assume the transition matrix at position m:

$M[k', k] \quad D$

https://eduassistpro.github.io/

$V \qquad 6.1$

Add WeChat edu_assist_pro

- Assume the following vectors:
  - $\alpha_{m-1}(k') = \begin{bmatrix} 40 & 30 & 65 \end{bmatrix}$
  - $\beta_m(k) = \begin{bmatrix} 45 & 65 & 30 \end{bmatrix}$

Computing Feature Expectations

$$\alpha_{m-1}(f_1 \odot M_m)\beta_m^\top = \begin{bmatrix} 40 & 30 & 65 \end{bmatrix} \begin{bmatrix} 0 & 0.1 & 0 \end{bmatrix} \begin{bmatrix} 45 \\ 65 \\ 30 \end{bmatrix}$$

$$= 37310.0$$

$$\alpha_{m-1}(f_1 \odot M_m)\beta_m^\top = \begin{bmatrix} 40 & 30 & 65 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0.4 \\ 0 & 0 & 0.3 \\ 0 & 0 & 0.01 \end{bmatrix} \begin{bmatrix} 45 \\ 65 \\ 30 \end{bmatrix}$$

$$= ?$$

CRFs

- Avoids the strong i
- Allows arbitrary features over observations (but not hidden states)
- Sound probabilistic model - gives us a distrib sequence labelings
- Uses the same e) for decoding as st
- Estimation/learning is harder (than sa use we have to compute the posterior for a sequ
- Empirical results generally show CRFs outperforms HMMs (and other classifiers)
- Feature estimation can be replaced with LSTM RNNs, resulting in what's called RNN-CRFs, of which LSTM-CRFs are the most widely used.