Generative approaches: Hidden Markov Models

Hidden Markov Models (HMM): the generative story

The generative story: first, the tags are drawn from a prior distribution; next, the tokens are drawn from a conditional likelihood.

$y_0 \leftarrow \Diamond, m \leftarrow 1$
repeat
  $y_m \sim Categorical(\lambda_{y_{m-1}})$ ▷ sample the current tag
  $w_m \sim Categorical(\phi_{y_m})$ ▷ sample the current word
until $y_m = \blacklozenge$ ▷ terminate when the stop symbol is generated

---

The independence assumptions of HMM

In addition to the usual independence assumptions for Naïve Bayes, two additional independence assumptions are needed for HMM:

- The probability of each word token only depends on its tag, not on any other element in the sequence:

Assignment Project Exam Help

$$P(\boldsymbol{w}|\boldsymbol{y}) = \prod_{m-1}^{M} P(w_m|y_m)$$

https://powcoder.com
Add WeChat powcoder

- Each tag y_m depends only on its predecessor

$$P(\boldsymbol{y}) = \prod_{m=1}^{M} P(y_m|y_{m-1})$$

when $y_m = \Diamond$ in all cases.

Parameter estimation of HMMs

The hidden Markov model has two groups of parameters:

- **Emission probabilities** ϕ. The probability $P_e(w_m|y_m;\phi)$ is the emission probability

- **Transition probabilities** λ. The probability $P_t(y_m|y_{m-1};\lambda)$

Both of these groups of parameters are typically computed from relative frequency estimation on a labeled corpus. The unsmoothed probabilities are,

$$\phi_{k,i} \triangleq Pr(W_m = i|Y_m = k) = \frac{count(W_m = i, Y_m = k)}{count(Y_m = k)}$$

$$\lambda_{k,k'} \triangleq Pr(Y_m = k'|Y_{m=1} = k) = \frac{count(Y_m = k', Y_{m-1} = k)}{count(Y_{m-1} = k)}$$

Inference for HMMs

- The goal of the inference in the hidden Markov model is to find the highest probability sequence:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \log P(\boldsymbol{y}|\boldsymbol{w})$$

- As $P(\boldsymbol{y}, \boldsymbol{w}) = P(\boldsymbol{y}|\boldsymbol{w}) \times P(\boldsymbol{w}) \propto P(\boldsymbol{y}|\boldsymbol{w})$, the inference problem can be reformulated as finding the joint probability of $\boldsymbol{w}$ and $\boldsymbol{y}$:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \log P(\boldsymbol{y}, \boldsymbol{w})$$

Inference for HMMs

- Applying the independent assumptions, we get:

$$
\begin{aligned}
\log P(\boldsymbol{y}, \boldsymbol{w}) &= \log P(\boldsymbol{y}) + \log P(\boldsymbol{w}|\boldsymbol{y}) \\
&= \sum_{m=1}^{M+1} \log P_Y(y_m|y_{m-1}) + \log P_{W|Y}(w_m|y_m) \\
&= \sum_{m=1}^{M+1} \log \lambda_{y_m, y_{m-1}} + \log \phi_{y_m, w_m} \\
&= \sum_{m=1}^{M+1} s_m(y_m, y_{m-1})
\end{aligned}
$$

- This allows us to apply a variant of the Viterbi algorithm, where the only parameters are the transition probabilities and emission probabilities, which correspond to two features at each position in the sequence. This limitation explains the performance disadvantage of HMMs.

Discriminative alternatives to HMMs

As with Naïve Bayes, one disadvantage of HMMs is that additional features (e.g., suffixes) cannot be applied without violating the independent assumptions. Discriminative models do not have this problem and additional features can be included:

$$
\begin{aligned}
&\boldsymbol{f}(\boldsymbol{w} = \text{the man who whistles tunes pianos}, \boldsymbol{y} = \text{DT NN WP VBZ VBZ NNS})\\
&= f(w_0 = the, y_0 = DT) + f(y_0 = DT, y_{-1} = \diamondsuit)\\
&\quad + f(w_0 = man, y_0 = NN) + f(y_0 = NN, y_{-1} = DT)\\
&\quad + f(w_0 = who, y_0 = WP) + f(y_0 = WP, y_{0-1} = NN)\\
&\quad + f(w_0 = whistles, y_0 = VBZ) + f(y_0 = VBZ, y_{-1} = WP) + f(suffix_0 = es)\\
&\quad + f(w_0 = tunes, y_0 = VBZ) + f(y_0 = VBZ, y_{-1} = VBZ) + f(suffix_0 = es)\\
&\quad + f(w_0 = pianos, y_0 = NNS) + f(y_0 = NNS, y_{-1} = VBZ)\\
&\quad + f(y_0 = \blacklozenge, y_{0-1} = NNS)
\end{aligned}
$$

Note that you do not need to add the same morphological feature for each word token.

Structured Perceptron for sequence labeling

Structured Perceptron

- Each tagging sequence is assigned a score with a linear model model

$$
\begin{aligned}
\Psi(\boldsymbol{w}, \boldsymbol{y}) &= \sum_{m=1}^{M+1} \psi(\boldsymbol{w}, y_m, y_{m-1}, m) \\
&= \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}, y_m, y_{m-1}, m) \\
&= \boldsymbol{\theta} \cdot \sum_{m=1}^{M+1} \boldsymbol{f}(\boldsymbol{w}, y_m, y_{m-1}, m) \\
&= \boldsymbol{\theta} \cdot \boldsymbol{f}^{(global)}(\boldsymbol{w}, \boldsymbol{y}_{1:M})
\end{aligned}
$$

  where $y_{M+1} = \blacklozenge$ and $y_0 = \Diamond$ by construction.
- The best tagging sequence can be found efficiently with the Viterbi algorithm.
- As a discriminative model, Perceptron can handle an arbitrary number of features at eash position.

Parameter estimation for structured perceptron

- In the training phase, the sentences in the training set are decoded one a time (with the Viterbi Algorithm).
- If the highest scoring tagging sequence is not the same as the correct tag sequence, the parameters are updated.

$$\hat{y} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{w})} \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}, \boldsymbol{y})$$

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} + \boldsymbol{f}(\boldsymbol{w}, \boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{w}, \hat{\boldsymbol{y}})$$

The averaged perceptron algorithm

procedure AVE_PERCEPTRON($\boldsymbol{x}^{1:N}, y^{1:N}$)
  $t \leftarrow 0, \boldsymbol{\theta}^{(0)} \leftarrow \mathbf{0}, \boldsymbol{m} \leftarrow \mathbf{0}$
  repeat
    $t \leftarrow t + 1$
    Select a sequence i ▷ Online training
    $\hat{\boldsymbol{y}} \leftarrow \text{argmax}_{\boldsymbol{y}}\, \boldsymbol{\theta}^{(t-1)} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y})$ ▷ Decoding by Viterbi
    if $\hat{\boldsymbol{y}} \neq \boldsymbol{y}^{(i)}$ **then**
      $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \boldsymbol{f}^{(global)}(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}) - \boldsymbol{f}^{(global)}(\boldsymbol{w}^{(i)}, \hat{\boldsymbol{y}})$
    else ▷ Feature count for entire sequence
      $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)}$
    $\boldsymbol{m} \leftarrow \boldsymbol{m} + \boldsymbol{\theta}^{(t)}$
  until tired
  $\bar{\boldsymbol{\theta}} = \frac{1}{t}\boldsymbol{m}$
  return $\bar{\boldsymbol{\theta}}$

Parameter Update example

- Correct tag sequence:
  - The_DT man_NN who_WP whistles_VBZ tunes_VBZ pianos_NNS
- Highest scoring sequence under the current model:
  - The_DT man_NN who_WP whistles_VBZ tunes_NNS pianos_NNS
- Which features need to be updated?

$$\theta_{(tunes,VBZ)} \leftarrow \theta_{(tunes,VBZ)} + 1 \quad \theta_{(tunes,NNS)} \leftarrow \theta_{(tunes,NNS)} - 1$$
$$\theta_{(VBZ,VBZ)} \leftarrow \theta_{(VBZ,VBZ)} + 1 \quad \theta_{(VBZ,NNS)} \leftarrow \theta_{(VBZ,NNS)} - 1$$
$$\theta_{(VBZ,NNS)} \leftarrow \theta_{(VBZ,NNS)} + 1 \quad \theta_{(NNS,NNS)} \leftarrow \theta_{(NNS,NNS)} - 1$$

Structured Support Vector Machines

Structured Support Vector Machines

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Classification with SVMs enforces a large-margin constraint that requires that there is a margin of at least 1 between the score of the correct label and all incorrect labels. Formally, this means

$$\forall y \neq y^{(i)}, \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y^{(i)}) - \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y) \geq 1$$

- This can be minimally extended to sequence labeling as

$$\forall \boldsymbol{y} \neq \boldsymbol{y}^{(i)}, \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}, \boldsymbol{y}^{(i)}) - \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}, \boldsymbol{y}) \geq 1$$

Extending SVMs to sequences

- A "structured" Support Vector Machine (SVM) outputs a structured object such as a sequence

- When extending SVMs to sequence labeling, two issues need to be addressed.

  - Some errors are more serious than others. Two labelings might have the same "margin" from the correct labeling, but have different number of tokens that are incorrectly labeled. We woould prefer to train "against" the latter rather than the former

  - Having a fixed margin of 1 would suggest we need to enumerate all possible sequences, which is infeasible as the number of sequences is exponential to the length of the sequence.
- The solution requires an adjustment of how the margin is computed.

Extending SVMs to sequences

- Instead of using a fixed margin, we use a cost function that reflects how the serious the errors are

$$\forall \boldsymbol{y} \neq \boldsymbol{y}^{(i)}, \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}) - \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}) \geq c(\boldsymbol{y}, \boldsymbol{y}^{(i)})$$

- Next, instead of using a delta function $c(\boldsymbol{y}, \boldsymbol{y}^{(i)}) = \delta(\boldsymbol{y}, \boldsymbol{y}^{(i)})$ to compute the error for the entire sequence, we use **Hamming cost**, which counts the number of errors in $\boldsymbol{y}$:

$$c(\boldsymbol{y}, \boldsymbol{y}^{(i)}) = \sum_{m=1}^{M+1} \delta(y_m \neq y_m^{(i)})$$

- Instead of training against all labelings $\boldsymbol{y}$ that have a margin that satisfies the above constraint, we focus on the prediction that **maximally** violates the margin constraint. We can identify this prediction by solving:

$$\begin{aligned}\hat{\boldsymbol{y}} &= \underset{\boldsymbol{y} \neq \boldsymbol{y}^{(i)}}{\operatorname{argmax}} \, \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}) - \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}) + c(\boldsymbol{y}, \boldsymbol{y}^{(i)}) \\ &= \underset{\boldsymbol{y} \neq \boldsymbol{y}^{(i)}}{\operatorname{argmax}} \, \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}) + c(\boldsymbol{y}, \boldsymbol{y}^{(i)})\end{aligned}$$

Extending SVMs to sequence labeling

- Reformulating the margin constraint, we get:

$$\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}) - \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{w})} \left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}) + c(\boldsymbol{y}, \boldsymbol{y}^{(i)}) \right) \geq 0$$

- This means that the constraint will be met (and training will be complete) when the score for $\boldsymbol{f}(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)})$ is equal or greater than the **cost-augmented** scores for all alternatives.

- In the training process, we identify predictions that are strong (scores highly according to the model) and wrong (incurs a large cost according to the truth), and reducing the scores of these predictions by adjusting weights.

- Note that Hamming cost can be reduced to local parts by adding a feature $f_m(y_m) = \delta(y_m \neq y_m^{(i)})$, and can be incorporated into the Viterbi algorithm for purposes of the identifying the prediction to train against.

A comparison between Structured Perceptron and Structured SVM

- In the training process, the perceptron algorithm simply finds the prediction that has the highest score according to the model to train against, while Structured SVM needs to find the prediction that has a high score with a heavy cost. Both can be identified with the Viterbi algorithm
- No cost needs to (or can be) computed during decoding for both models
- In practice, with a large training set, the perceptron algorithm works pretty well, obviating the need for more complicated SVM models.