Lecture 13: Discriminative Sequence Tagging

Nathan Schneider

ENLP | 17 October 2016

HMM + features

- There are variants of the generative HMM that emit features instead of just words.

- However, these suffer from similar problems as features in naïve Bayes (too strong independence assumptions).

- Can we be **discriminative** instead?
  - Yes! In fact, we can reuse the same machinery for discriminative learning with **linear models**.

Recasting HMM as a Linear Model

- Recall that a linear model is one that scores candidate outputs y with $\mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}, y)$. Decoding = $\arg\max_{y'} \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}, y')$.

- Not just classification: we can be predicting a structured output $\mathbf{y}$. Thus $\arg\max_{\mathbf{y}'} \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}, \mathbf{y}')$.

- How can we express an HMM in this framework?
  - transitions = features over tag n-grams
  - emissions = tag + word features
  - weights = log probabilities
  - $\arg\max_{\mathbf{y}'}$ = Viterbi decoding

Viterbi for Linear Models

- Essentially, the Viterbi algorithm stays the same:
  - *transition probabilities* replaced by linear score of **transition (multi-tag) features**
  - *emission probabilities* replaced by linear score of **non-transition (single-tag) features**

D

Generative → Discriminative

- If we want to estimate the weights without making independence assumptions about the features…
- …we can use a discriminative learning algorithm!
- However, the algorithm has to take the **structure** of the output into account. Tag n-gram features mean the prediction of one tag influences what the model thinks about other tags.
- Machine learning with models where the outputs are interrelated is called **structured prediction**.

Review: Perceptron Learner

$\mathbf{X} \rightarrow$

$\mathbf{Y} \rightarrow$

$\mathbf{w} \leftarrow \mathbf{0}$

for i = 1 … I:

  for t = 1 … T:

    select $(\mathbf{x}, y)_t$

    # run current classifier

    $\hat{y} \leftarrow \arg\max_{y'} \mathbf{w}^\intercal \boldsymbol{\Phi}(\mathbf{x}, y')$

    if $\hat{y} \neq y$ then *# mistake*

      $\mathbf{w} \leftarrow \mathbf{w} + \boldsymbol{\Phi}(\mathbf{x}, y) - \boldsymbol{\Phi}(\mathbf{x}, \hat{y})$

return $\mathbf{w}$

L

$\rightarrow \boxed{\mathbf{w}}$

Review: Perceptron Learner

$\mathbf{X} \rightarrow$

$\mathbf{Y} \rightarrow$

$\mathbf{w} \leftarrow \mathbf{0}$

for i = 1 … I:

  for t = 1 … T:

    select $(\mathbf{x}, y)_t$

    # run current classifier

    $\hat{y} \leftarrow$ $\boxed{C}$ $\leftarrow \mathbf{x}$ **decoding** is a subroutine of learning

    if $\hat{y} \neq y$ then *# mistake*

      $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{\Phi}(\mathbf{x}, y) - \mathbf{\Phi}(\mathbf{x}, \hat{y})$

return $\mathbf{w}$

L

$\rightarrow$ $\boxed{\mathbf{w}}$

Structured Perceptron Learner

$\mathbf{X} \rightarrow$

$\mathbf{Y} \rightarrow$

$\mathbf{w} \leftarrow \mathbf{0}$

for i = 1 … I:

  for t = 1 … T:

    select $(\mathbf{x}, \mathbf{y})_t$

    # run structured decoding

    $\hat{\mathbf{y}} \leftarrow \boxed{D} \leftarrow \mathbf{x}$ **decoding** is a subroutine of learning

    if $\hat{\mathbf{y}} \neq \mathbf{y}$ then *# mistake*

      $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{\Phi}(\mathbf{x}, \mathbf{y}) - \mathbf{\Phi}(\mathbf{x}, \hat{\mathbf{y}})$

return $\mathbf{w}$

L

$\rightarrow \boxed{\mathbf{w}}$

Structured Perceptron Learner

$\mathbf{X} \rightarrow$

$\mathbf{Y} \rightarrow$

$\mathbf{w} \leftarrow \mathbf{0}$

for i = 1 … I:

  for t = 1 … T:

    select $(\mathbf{x}, \mathbf{y})_t$

    # run structured decoding

    $\hat{\mathbf{y}} \leftarrow \boxed{D} \leftarrow \mathbf{x}$ For sequence tagging, **decoder = Viterbi!**

    if $\hat{\mathbf{y}} \neq \mathbf{y}$ then *# mistake: incorrect tag(s)*

      $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{\Phi}(\mathbf{x}, \mathbf{y}) - \mathbf{\Phi}(\mathbf{x}, \hat{\mathbf{y}})$

return $\mathbf{w}$ update affects weights of features that fire for mistagged tokens

L

$\rightarrow \boxed{\mathbf{w}}$

Structured Perceptron

- What are the constraints on the kinds of features we can use? (tag bigrams? trigrams? word bigrams? trigrams?)
  - Remember that discriminative = we don't care about modeling the probability of the language. Thus, every model feature should involve at least one tag.
  - As a sequence model, **Markov order** is still relevant: if we want to use the *bigram* Viterbi algorithm, which is $O(T^2N)$, we can have features over *tag bigrams*, but not trigrams.
  - **local feature** = feature which respects the independence assumptions of the decoding algorithm (e.g., tag bigram Viterbi). Using nonlocal features would require fancier algorithms.
  - Unlike the generative HMM, **no constraint on which words** can be in a feature. E.g., there could be a feature that relates the first tag to the last token! (In POS tagging, perhaps ending with "?" correlates with certain kinds of initial words.)

Discriminative Classifiers: Non-probabilistic

- The structured counterpart of the perceptron classifier is called…the structured perceptron.
  - Also: structural SVM (max-margin).

Discriminative Classifiers: Probabilistic

- The structured counterpart of the logistic regression classifier: **conditional random field (CRF)**.
  - Most common: **linear-chain** structure, i.e., sequence
  - Probabilistic—linear score is exponentiated & normalized
  - Training requires forward-backward algorithm (expensive!)
  - Generally state-of-the-art
  - Downloadable implementations include CRF++
  - If you want the gory details: Sutton & McCallum, http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf
- There is also the **Maximum Entropy Markov Model (MEMM)**, which makes simplifying assumptions to reduce computation and is nearly as accurate in practice.

Final Projects

- See how NLP components fit together in a system
  - off-the-shelf tools such as spaCy, Stanford CoreNLP
  - + substantial new code
- Work in an interdisciplinary team of 3–4 people
  - Each team should have at least 2 departments/programs represented
  - Design the project to suit the team’s strengths! (programming, data collection, analysis)
- Build something cool!
  - artistic, scientific, or practical
  - using data (existing or new) & concepts from this course
  - start simple, then iterate
- Instructor & TA will help you scope the project, find relevant literature, design evaluation, etc.