Lecture 1, COMS E6998-3, Spring 2011

Michael Collins

January 19, 2011

A Machine-Learning Example: Hand-Written Digit Recognition

- The problem: given a hand-written digit, decide whether it is $0, 1, 2, \ldots$ or 9
- A learning approach:
  1. Collect several hundred/thousand example digits, and label them by hand to form a *training set*
  2. Automatically learn a digit recognition *model* from the training set
  3. Apply the model to new, previously unseen hand-written digits
- Systems built in this way are in widespread use in the U.S. postal service (ZIP-code recognition), and in automatic check-reading

Today's Lecture

- Introduction:
  - Example problems from machine learning for NLP
  - Topics we'll cover in the course
  - Background required for the course
  - Projects/homework assignments
- Topic 1: Hidden Markov models
- Topic 2: Log-linear models

Related Problems

- Identifying faces within an image (see the Viola and Jones face detector)
- Text classification/spam filtering
- Medical applications: e.g., classification of cancer type
- Information retrieval: e.g., ranking web-pages in order of relevance to a given query

Supervised Learning Problems

- Goal: Learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- We have n training examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

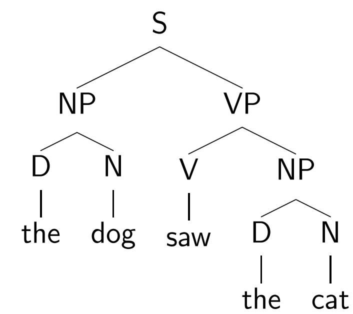
  where each $x_i \in \mathcal{X}$, and each $y_i \in \mathcal{Y}$
- Often (not always) $\mathcal{X} = \mathbb{R}^d$ for some integer d
- Some possibilities for $\mathcal{Y}$:
  - $\mathcal{Y} = \{-1, +1\}$ (binary classification)
  - $\mathcal{Y} = \{1, 2, \ldots, k\}$ for some $k > 2$ (multi-class classification)
  - $\mathcal{Y} = \mathbb{R}$ (regression)

Context-Free Parsing

- The task: learn a function that maps a sentence, e.g.,

  the dog saw the cat

  to a parse tree,



Sequence Labeling Problems

- Task: learn a function that maps an input sequence

$$x_1, x_2, \ldots, x_m$$

  to an output sequence

$$y_1, y_2, \ldots, y_m$$

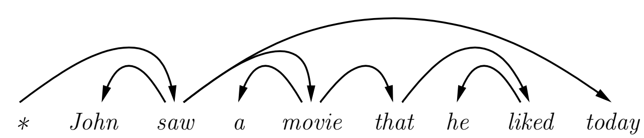
  Note: each $y_i \in \mathcal{Y}_i$ where $\mathcal{Y}_i$ is a **finite** set of possible labels at the i'th position
- This is a core problem in natural language processing
- Examples: part-of-speech tagging, named-entity recognition

Dependency Parsing

- The task: learn a function that maps a sentence, e.g.,

  John saw a movie that he liked today

  to a *dependency structure*,



Machine Translation

- The task: learn a function that maps a sentence in one language, e.g.,

  In wenigen Tagen finden Parlamentswahlen in Slowenian statt

  to a sentence in another language,

  In a few days elections take place in Slovenia

Topics Covered in the Class

- Probabilistic models for structured NLP data
  - e.g., hidden Markov models (HMMs), maximum-entropy Markov models (MEMMs), conditional random fields (CRFs), probabilistic context-free grammars, synchronous context-free grammars, dependency parsing models, etc.
- Inference algorithms
  - e.g., dynamic programming, belief propagation, methods based on linear programming and integer linear programming, dual decomposition/Lagrangian relaxation
- Semi-supervised learning
  - e.g., deriving lexical representations from unlabeled data, cotraining, entropy regularization, canonical correlation analysis (CCA)

Mapping Sentences to Logical Form

- The task: learn a function that maps a sentence e.g.,

  Show me the latest flight from Boston to Seattle on Friday

  to a expression in logical form that represents its meaning, e.g.,

  $argmax(\lambda x.flight(x) \wedge from(x, BOS) \wedge to(x, SEA) \wedge day(x, FRI), \lambda y.time(y))$

Admin

- Background required for the class: a prior class in machine learning and/or natural language processing
- Evaluation:
  - Final class project (65%)
  - 3 homeworks (25%)
  - Class participation (10%)

Lecture 1, COMS E6998-3: Hidden Markov Models

Michael Collins

January 19, 2011

Overview

- Markov models
- Hidden Markov models

Markov Sequences

- Consider a sequence of random variables $X_1, X_2, \ldots, X_m$ where m is the length of the sequence
- Each variable X_i can take any value in $\{1, 2, \ldots, k\}$
- How do we model the joint distribution

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m)$$

?

The Markov Assumption

$$\begin{aligned} & P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) \\ = \; & P(X_1 = x_1) \prod_{j=2}^{m} P(X_j = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) \\ = \; & P(X_1 = x_1) \prod_{j=2}^{m} P(X_j = x_j | X_{j-1} = x_{j-1}) \end{aligned}$$

- The first equality is exact (by the chain rule).
- The second equality follows from *the Markov assumption*: for all $j = 2 \ldots m$,

$$P(X_j = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) = P(X_j = x_j | X_{j-1} = x_{j-1})$$

Homogeneous Markov Chains

- In a *homogeneous* Markov chain, we make an additional assumption, that for $j = 2 \ldots m$,

$$P(X_j = x_j | X_{j-1} = x_{j-1}) = q(x_j | x_{j-1})$$

where $q(x'|x)$ is some function

- Idea behind this assumption: the transition probabilities do not depend on the position in the Markov chain (do not depend on the index j)

A Generative Story for Markov Models

- A sequence $x_1, x_2, \ldots, x_m$ is generated by the following process:

1. Pick x_1 at random from the distribution $q(x)$
2. For $j = 2 \ldots m$:
   - Choose x_j at random from the distribution $q(x|x_{j-1})$

Markov Models

- Our model is then as follows:

$$p(x_1, x_2, \ldots x_m; \underline{\theta}) = q(x_1) \prod_{j=2}^{m} q(x_j | x_{j-1})$$

- Parameters in the model:
  - $q(x)$ for $x = \{1, 2, \ldots, k\}$
    Constraints: $q(x) \geq 0$ and $\sum_{x=1}^{k} q(x) = 1$
  - $q(x'|x)$ for $x = \{1, 2, \ldots, k\}$ and $x' = \{1, 2, \ldots, k\}$
    Constraints: $q(x'|x) \geq 0$ and $\sum_{x'=1}^{k} q(x'|x) = 1$

Today's Lecture

- Markov models
- Hidden Markov models

Modeling Pairs of Sequences

- In many applications, we need to model *pairs* of sequences
- Examples:
  1. Part-of-speech tagging in natural language processing (assign each word in a sentence to one of the categories noun, verb, preposition etc.)
  2. Speech recognition (map acoustic sequences to sequences of words)
  3. Computational biology: recover gene boundaries in DNA sequences

Probabilistic Models for Sequence Pairs

- We have two sequences of random variables: $X_1, X_2, \ldots, X_m$ and $S_1, S_2, \ldots, S_m$
- Intuitively, each X_i corresponds to an "observation" and each S_i corresponds to an underlying "state" that generated the observation. Assume that each S_i is in $\{1, 2, \ldots k\}$, and each X_i is in $\{1, 2, \ldots o\}$
- How do we model the joint distribution

$$P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)$$

?

Hidden Markov Models (HMMs)

- In HMMs, we assume that:

$$P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)$$
$$= P(S_1 = s_1) \prod_{j=2}^{m} P(S_j = s_j | S_{j-1} = s_{j-1}) \prod_{j=1}^{m} P(X_j = x_j | S_j = s_j)$$

Independence Assumptions in HMMs

- By the chain rule, the following equality is exact:

$$\begin{aligned} & P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m) \\ = \quad & P(S_1 = s_1, \ldots, S_m = s_m) \times \\ & P(X_1 = x_1, \ldots, X_m = x_m | S_1 = s_1, \ldots, S_m = s_m) \end{aligned}$$

- Assumption 1: the state sequence forms a Markov chain

$$P(S_1 = s_1, \ldots, S_m = s_m) = P(S_1 = s_1) \prod_{j=2}^{m} P(S_j = s_j | S_{j-1} = s_{j-1})$$

Independence Assumptions in HMMs

- By the chain rule, the following equality is exact:

$$\begin{aligned} & P(X_1 = x_1, \ldots, X_m = x_m | S_1 = s_1, \ldots, S_m = s_m) \\ = & \prod_{j=1}^{m} P(X_j = x_j | S_1 = s_1, \ldots, S_m = s_m, X_1 = x_1, \ldots X_{j-1} = x_j) \end{aligned}$$

- Assumption 2: each observation depends only on the underlying state

$$\begin{aligned} & P(X_j = x_j | S_1 = s_1, \ldots, S_m = s_m, X_1 = x_1, \ldots X_{j-1} = x_j) \\ = & P(X_j = x_j | S_j = s_j) \end{aligned}$$

A Generative Story for Hidden Markov Models

- Sequence pairs $s_1, s_2, \ldots, s_m$ and $x_1, x_2, \ldots, x_m$ are generated by the following process:

  1. Pick s_1 at random from the distribution $t(s)$. Pick x_1 from the distribution $e(x|s_1)$
  2. For $j = 2 \ldots m$:
     - Choose s_j at random from the distribution $t(s|s_{j-1})$
     - Choose x_j at random from the distribution $e(x|s_j)$

The Model Form for HMMs

- The model takes the following form:

$$p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta}) = t(s_1) \prod_{j=2}^{m} t(s_j | s_{j-1}) \prod_{j=1}^{m} e(x_j | s_j)$$

- Parameters in the model:

  1. Initial state parameters $t(s)$ for $s \in \{1, 2, \ldots, k\}$
  2. Transition parameters $t(s'|s)$ for $s, s' \in \{1, 2, \ldots, k\}$
  3. Emission parameters $e(x|s)$ for $s \in \{1, 2, \ldots, k\}$ and $x \in \{1, 2, \ldots, o\}$

Today's Lecture

- More on Hidden Markov models:
  - parameter estimation
  - The Viterbi algorithm

Parameter Estimation with Fully Observed Data

- We'll now discuss parameter estimates in the case of *fully observed data*: for $i = 1 \ldots n$, we have pairs of sequences $x_{i,j}$ for $j = 1 \ldots m$ and $s_{i,j}$ for $j = 1 \ldots m$. (i.e., we have n training examples, each of length m.)

Parameter Estimation: Emission Parameters

- Assume we have fully observed data: for $i = 1 \ldots n$, we have pairs of sequences $x_{i,j}$ for $j = 1 \ldots m$ and $s_{i,j}$ for $j = 1 \ldots m$
- Define $\text{count}(i, s \rightsquigarrow x)$ to be the number of times state s is paired with emission x. More formally:

$$\text{count}(i, s \rightsquigarrow x) = \sum_{j=1}^{m} [[s_{i,j} = s \wedge x_{i,j} = x]]$$

- The maximum-likelihood estimates of emission probabilities are then

$$e(x|s) = \frac{\sum_{i=1}^{n} \text{count}(i, s \rightsquigarrow x)}{\sum_{i=1}^{n} \sum_{x} \text{count}(i, s \rightsquigarrow x)}$$

Parameter Estimation: Transition Parameters

- Assume we have fully observed data: for $i = 1 \ldots n$, we have pairs of sequences $x_{i,j}$ for $j = 1 \ldots m$ and $s_{i,j}$ for $j = 1 \ldots m$
- Define $\text{count}(i, s \rightarrow s')$ to be the number of times state s' follows state s in the i'th training example. More formally:

$$\text{count}(i, s \rightarrow s') = \sum_{j=1}^{m-1} [[s_{i,j} = s \wedge s_{i,j+1} = s']]$$

(We define $[[\pi]]$ to be 1 if π is true, 0 otherwise.)

- The maximum-likelihood estimates of transition probabilities are then

$$t(s'|s) = \frac{\sum_{i=1}^{n} \text{count}(i, s \rightarrow s')}{\sum_{i=1}^{n} \sum_{s'} \text{count}(i, s \rightarrow s')}$$

Parameter Estimation: Initial State Parameters

- Assume we have fully observed data: for $i = 1 \ldots n$, we have pairs of sequences $x_{i,j}$ for $j = 1 \ldots m$ and $s_{i,j}$ for $j = 1 \ldots m$
- Define $\text{count}(i, s)$ to be 1 if state s is the initial state in the sequence, and 0 otherwise:

$$\text{count}(i, s) = [[s_{i,1} = s]]$$

- The maximum-likelihood estimates of initial state probabilities are:

$$t(s) = \frac{\sum_{i=1}^{n} \text{count}(i, s)}{n}$$

Today's Lecture

- Hidden Markov models:
  - parameter estimation
  - the Viterbi algorithm

The Viterbi Algorithm

- Goal: for a given input sequence $x_1, \ldots, x_m$, find

$$\arg\max_{s_1,\ldots,s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta})$$

- The *Viterbi algorithm* is a dynamic programming algorithm. Basic data structure:

$$\pi[j, s]$$

will be a table entry that stores the maximum probability for any state sequence ending in state s at position j. More formally: $\pi[1, s] = t(s)e(x_1|s)$, and for $j > 1$,

$$\pi[j, s] = \max_{s_1 \ldots s_{j-1}} \left[t(s_1)e(x_1|s_1) \left(\prod_{k=2}^{j-1} t(s_k|s_{k-1})e(x_k|s_k) \right) t(s|s_{j-1})e(x_j|s) \right]$$

The Viterbi Algorithm

- Goal: for a given input sequence $x_1, \ldots, x_m$, find

$$\arg\max_{s_1,\ldots,s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta})$$

- This is the most likely state sequence $s_1 \ldots s_m$ for the given input sequence $x_1 \ldots x_m$

The Viterbi Algorithm

- Initialization: for $s = 1 \ldots k$

$$\pi[1, s] = t(s)e(x_1|s)$$

- For $j = 2 \ldots m$, $s = 1 \ldots k$:

$$\pi[j, s] = \max_{s' \in \{1 \ldots k\}} [\pi[j-1, s'] \times t(s|s') \times e(x_j|s)]$$

- We then have

$$\max_{s_1 \ldots s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta}) = \max_s \pi[m, s]$$

- The algorithm runs in $O(mk^2)$ time

Viterbi as a Shortest-Path Algorithm

- The input sequence $x_1 \ldots x_m$ is fixed
- Have vertices in a graph labeled (j, s) for $s \in \{1 \ldots k\}$ and $j = 1 \ldots m$. In addition have a source vertex labeled 0
- For $s \in \{1 \ldots k\}$, we have a directed edge from vertex 0 to vertex $(1, s)$, with weight $t(s)e(x_1|s)$
- For each $j = 2 \ldots m$, and $s, s' \in \{1 \ldots k\}$, have a directed edge from $(j-1, s)$ to (j, s') with weight $t(s'|s)e(x_j|s')$ (the weight of any path is the product of weights on edges in the path)
- $\pi[j, s]$ is the highest weight for any path from vertex 0 to vertex (j, s)

The Viterbi Algorithm: Backpointers

- Initialization: for $s = 1 \ldots k$

$$\pi[1, s] = t(s)e(x_1|s)$$

- For $j = 2 \ldots m$, $s = 1 \ldots k$:

$$\pi[j, s] = \max_{s' \in \{1 \ldots k\}} \left[\pi[j-1, s'] \times t(s|s') \times e(x_j|s)\right]$$

  and

$$bp[j, s] = \arg\max_{s' \in \{1 \ldots k\}} \left[\pi[j-1, s'] \times t(s|s') \times e(x_j|s)\right]$$

- The bp entries are backpointers that will allow us to recover the identity of the highest probability state sequence

Viterbi Algorithm: Backpointers (continued)

- Highest probability for any sequence of states is

$$\max_s \pi[m, s]$$

- To recover identity of highest-probability sequence:

$$s_m = \arg\max_s \pi[m, s]$$

  and for $j = m \ldots 2$,

$$s_{j-1} = bp[j, s_j]$$

- The sequence of states $s_1 \ldots s_m$ is then

$$\arg\max_{s_1, \ldots, s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta})$$