Structured learning for sequence labeling
Part 2: Sequence Labeling and Hidden Markov Models

Hakan Erdogan
Sabanci University

August 2-5, 2010
Enterface'10 workshop, Amsterdam

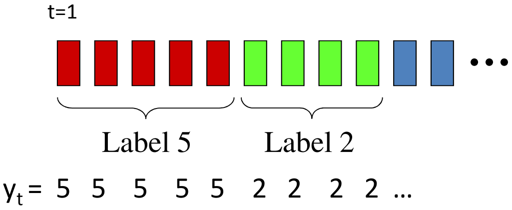
Outline

Sequence labeling problem definition I

- Given a sequence of features $\mathbf{x}_{1:T}$, find appropriate labels $y_{1:T}$ where each $y_t \in \mathcal{Y}$, we can assume wlog that $\mathcal{Y} = [M]$, a finite set
- This is a hard problem and the number of possible $y_{1:T}$ is too high, namely M^T and T is changeable
- We need additional assumptions on output labels y_t, such as being Markov
- Supervised learning problem: given training data sequences $\left\{(x_{1:T}^{(i)}, y_{1:T}^{(i)}) : i = 1, \ldots, N\right\}$, find a model that will predict $y_{1:T}$ given testing data $x_{1:T}$
- Note that, training and test sequences can be of different length T, but we do not explicitly indicate it to avoid clutter in our representation

Sequence labeling problem definition II

- Partially supervised learning: We do not know the label sequence $y_{1:T}^{(i)}$, but we know a sequence-specific grammar that the label sequence should obey (common case in speech recognition)



Sequence labeling applications

- Speech recognition
- Part-of-speech tagging
- Shallow parsing
- Handwriting recognition
- Protein secondary structure prediction
- Video analysis
- Facial expression dynamic modeling

Urns and balls example

- Assume there are two urns with black and white balls [Rabiner, 1989]
- One urn has more black than white (90% vs 10%) and vice versa
- Someone pulls out one ball at a time and shows us without revealing which urn he uses and puts it back into the urn
- He is more likely to use the same urn (90% chance) once he starts using one
- We are looking only at the sequence of balls and recording them

Questions about the urns and balls example

- Questions of interest:
  1. Can we predict which urn is used at a given time?
  2. What is the probability of observing the sequence of balls shown to us?
  3. Can we estimate/learn the ratio of balls in each urn by looking at a long sequence of balls if we did not know the ratios beforehand?

Jason Eisner's ice-cream example

- Example excel sheet online (illustrates forward backward algorithm)
- Example also adopted in [Jurafsky and Martin, 2008]
- Try to guess whether the weather was hot or cold by observing only how many ice-creams (0, 1, 2 or 3+) Jason ate each day in a sequence of 30 days
- Two states and observations with 4 distinct values (discrete observations)
- Question: Can we determine if a day was hot or cold given the sequence of ice-creams consumed by Jason?

Human activity labeling in an exercise video

- Assume we are given an exercise video of a single person and we are interested in labeling actions of the person as either "standing", "squatting" or "lying down" (assume for now that no other action is present)
- We track the subject and have a bounding box around her/him at each frame of the video
- We consider as features $\mathbf{x}_t = [h_t, w_t]^T$ where h_t is the height of the bounding box and w_t is the width of the bounding box
- So, we have continuous (real) observations and three labels
- Question: Given the height and width of the bounding boxes in all frames, can we determine the action type in each frame?

Independent solution

- Simplest solution is to assume independence of sequence labels
- Find y_t such that $p(y_t|\mathbf{x}_t)$ is maximized independently
- This is suboptimal since it does not use the relation between neighboring labels
- This approach is prone to errors due to independence assumption not being valid most of the time
- One should consider the relation among neighboring labels
- A natural assumption is Markov assumption on labels which leads to hidden Markov models

Outline

1. Sequence Labeling

2. Hidden Markov Models

What is a hidden Markov model? I

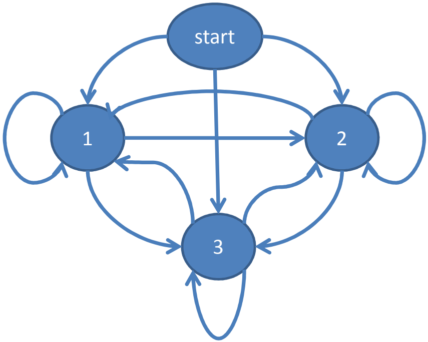
- A tool that helps us solve sequence labeling problems
- Observations $\mathbf{x}_{1:T}$ are modeled by a state machine (that is hidden) that generates them (generative model)
- States y_t correspond to labels, state sequence is $y_{1:T}$
- A finite set of labels is possible, $y_t \in \mathcal{Y}$ where $|\mathcal{Y}|$ is finite
- Markov assumption $p(y_t|y_{t-1}, y_{t-2}, \ldots, y_1) = p(y_t|y_{t-1})$
- Transition from one state (y_{t-1}) to another (y_t) occurs at each time instant
- Meanwhile an observation ($\mathbf{x}_t$) is emitted after the transition
- Parameters of the model:
    - Probabilities of transitions among states
    - Probabilities of emission of observations from states
    - Probabilities of starting at states

Three views of HMMs

An HMM can be viewed in three different ways

- State transition diagram
- Graphical model
- Trellis / lattice diagram

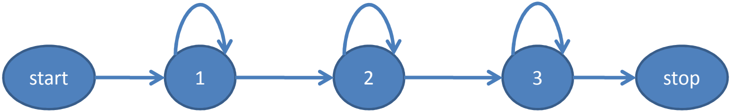
State transition diagram - fully connected



Time is not explicitly shown in this diagram, at each time instant a transition followed by an emission occurs

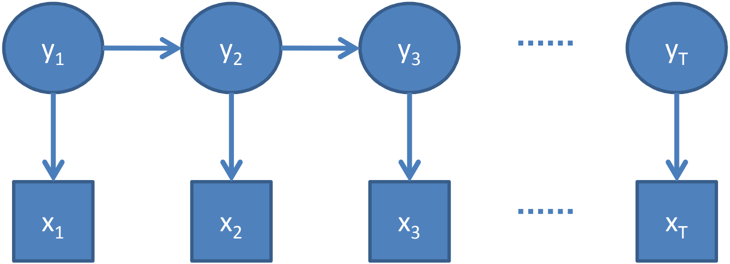
All transitions are possible with a certain probability in this example

State transition diagram - left-to-right

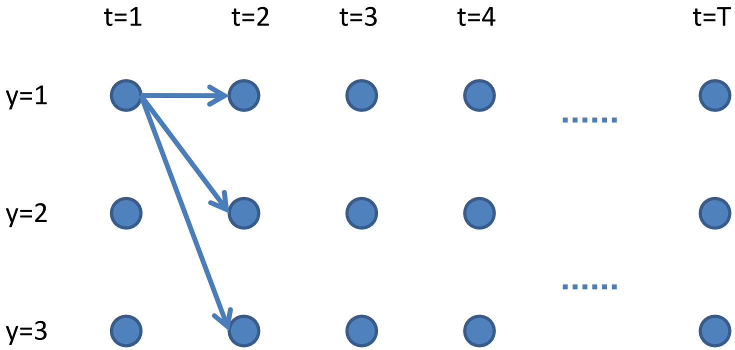


Some transitions are not possible (their probabilities are set to zero)

Graphical model

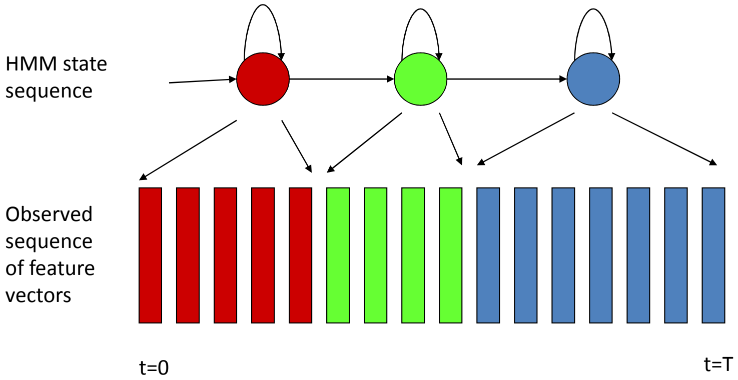


Trellis / lattice



Observations are not shown, the labels (states) are explicitly shown
Graphical model is expanded at each time instant to reveal all possible states

A possible alignment



Depicting a possibility of alignment of observed data to an underlying left-to-right HMM

Variables

- Observations $\mathbf{x}_{1:T}$
  - $\mathbf{x}_t \in \mathbb{R}^d$ for continuous observations HMM
  - $\mathbf{x}_t \in [N_o]$ for discrete observations HMM
- $y_{1:T}$ state sequence, $y_t \in [M]$ is the state at time t
- $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$: model parameters
  - $\mathbf{A}$ where $A_{ij} = p(y_{t+1} = j | y_t = i)$ is the transition matrix
  - For discrete observations $\mathbf{B}$ is a matrix where $B_{ik} = p(x_t = k | y_t = i)$ are emission probabilities
  - For continuous observations with Gaussian emission distributions we have $p(\mathbf{x}_t | y_t = i) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, we may think of $\mathbf{B}$ as the set of mean and (co)variance parameters $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)_{i=1}^M$
  - π where $\pi_i = p(y_1 = i)$ initial state probabilities, we can remove π if we introduce a "start" state which has initial probability of one

Rabiner's three problems of HMMs

- Problem 1: Probability/likelihood calculation: Given an observation sequence, how can I calculate the probability of observing it given an underlying HMM model $p(\mathbf{x}_{1:T}|\lambda)$
- Problem 2: Alignment/decoding/inference: What is the most likely state sequence given an observation sequence and an HMM model? $y^*_{1:T} = \arg\max_{y_{1:T}} p(y_{1:T}|x_{1:T}, \lambda)$
  - We may also be interested in $y^*_t = \arg\max_{y_t} p(y_t|x_{1:T}, \lambda)$
- Problem 3: Training/learning: How can I train the parameters of an HMM given training data $\mathbf{x}^{(i)}_{1:T}$? How to choose λ to maximize $\prod_i p(\mathbf{x}^{(i)}_{1:T}|\lambda)$?
  - Note that, if we are given $(\mathbf{x}^{(i)}_{1:T}, y^{(i)}_{1:T})$ (aka fully supervised training), maximum-likelihood training becomes just a counting process

Problem 1: Computing $P(\mathbf{x}_{1:T}|\lambda)$

$$\begin{aligned} p(\mathbf{x}_{1:T}|\lambda) &= \sum_{y_{1:T}} p(\mathbf{x}_{1:T}, y_{1:T}|\lambda) \\ &= \sum_{y_{1:T}} p(\mathbf{x}_{1:T}|y_{1:T}, \lambda) p(y_{1:T}|\lambda) \end{aligned}$$

where $p(\mathbf{x}_{1:T}|y_{1:T}, \lambda) = \prod_t p(\mathbf{x}_t|y_t, \lambda)$ is the multiplication of emission probabilities and $p(y_{1:T}|\lambda) = \prod_t p(y_t|y_{t-1}, \lambda)$ is the multiplication of transition probabilities

- Hard to enumerate all state sequences $y_{1:T}$
- Almost impossible to find the result using this way
- Instead, we use an iterative method (dynamic programming) called the forward algorithm

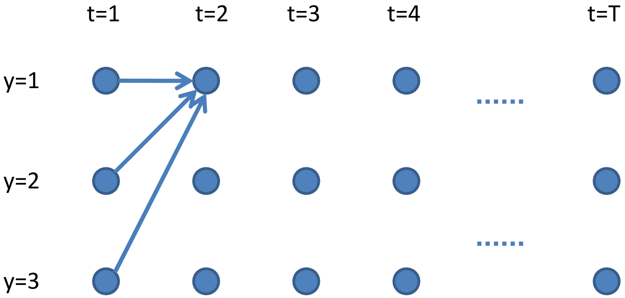
Forward algorithm

- Define partial probabilities $\alpha_t(j) = p(\mathbf{x}_{1:t}, y_t = j|\lambda)$, note that $\sum_j \alpha_T(j)$ is the desired probability of observation $p(\mathbf{x}_{1:T}|\lambda)$
- Iteratively update α's in time $\alpha_t(j) = \sum_{i=1}^{M} \alpha_{t-1}(i) a_{ij} p(\mathbf{x}_t|j)$
- We can visualize this on a trellis

The algorithm

1. Initialize $\alpha_1(j) = \pi_j p(\mathbf{x}_1|j)$ for $j = 1, \ldots, M$
2. Update $\alpha_t(j) = \sum_{i=1}^{M} \alpha_{t-1}(i) a_{ij} p(\mathbf{x}_t|j)$ for $j = 1, \ldots, M$
3. Terminate: $p(\mathbf{x}_{1:T}|\lambda) = \sum_{j=1}^{M} \alpha_T(j)$

Forward algorithm on a trellis



$$\alpha_t(j) = \sum_{i=1}^{M} \alpha_{t-1}(i) a_{ij} p(x_t \mid j)$$

$$\alpha_2(1) = \left[\alpha_1(1) a_{11} + \alpha_1(2) a_{21} + \alpha_1(2) a_{31} \right] p(x_2 \mid 1)$$

Problem 2: Alignment/decoding/inference

- We would like to find optimal $y^*_{1:T} = \arg\max_{y_{1:T}} p(y_{1:T}|x_{1:T}, \lambda)$
- Use another dynamic programming algorithm called Viterbi algorithm
- Simply replace the sum in the forward algorithm with a max operation
- Also, hold a backpointer at each state to remember the maximum scoring path

Viterbi algorithm

- Define partial maximal probabilities $V_t(j) = \max_{y_{1:t-1}} p(\mathbf{x}_{1:t}, y_{1:t-1}, y_t = j|\lambda)$
- Iteratively update V's in time $V_t(j) = \max_{i=1}^{M} V_{t-1}(i) a_{ij} p(\mathbf{x}_t|j)$
- We can visualize this on a trellis (same picture as forward algorithm, replace sum with max)

The algorithm

1. Initialize $V_1(j) = \pi_j p(\mathbf{x}_1|j)$
2. Update
   - $V_t(j) = \max_{i=1}^{M} V_{t-1}(i) a_{ij} p(\mathbf{x}_t|j)$
   - Hold a backpointer $\psi_t(j) = \arg\max_i V_{t-1}(i) a_{ij} p(\mathbf{x}_t|j)$
3. Terminate
   - Perform the update at step T
   - Trace back the path from $\psi_T(y_T^*)$ where y_T^* is the maximum likely end state

Problem 3: Training I

- Given $(\mathbf{x}^{(i)}_{1:T_i})^N_{i=1}$, maximum likelihood training requires finding

$$\hat{\lambda} = \arg\max_{\lambda} \sum_{i=1}^{N} \log\left(p(\mathbf{x}^{(i)}_{1:T_i}|\lambda)\right)$$

- For simplicity, assume single sequence $\mathbf{x}_{1:T}$ for training, generalization to multiple sequences is trivial
- Direct maximization is not easy, use Expectation Maximization (EM) algorithm
- Latent data is the label sequence $(y_{1:T})$
  1. Start with an initial λ^{old}
  2. Expectation step (E-step): Compute posterior probability of the latent variables $p(y_{1:T}|\mathbf{x}_{1:T}, \lambda^{old})$

Problem 3: Training II

3. Maximization step (M-step): Find λ that maximizes the auxiliary function which is the expected log-likelihood of the complete data under the posterior found in the E-step

$$Q(\lambda, \lambda^{old}) = \sum_{y'_{1:T}} p(y'_{1:T} | \mathbf{x}_{1:T}, \lambda^{old}) \log p(\mathbf{x}_{1:T}, y'_{1:T} | \lambda)$$

- Initialization is very important and it can be more art than science
- In case of HMMs, EM algorithm is called the forward-backward algorithm
- Need to propagate forward and backward variables for the E-step

Backward Algorithm

Similar to forward algorithm, we need a backward algorithm where we define

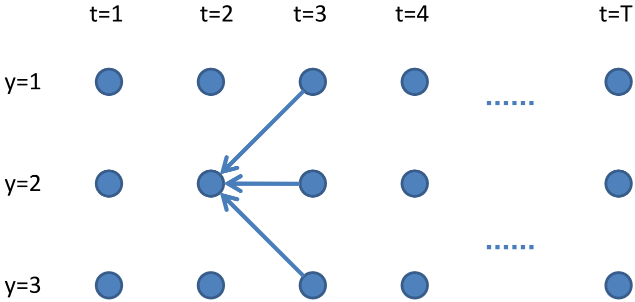
$$\beta_t(i) = p(x_{t+1:T} | y_t = i, \lambda)$$

The update is from final time to the beginning time and the update rule becomes (follows from probabilities and graphical model of HMMs)

$$\beta_t(i) = \sum_{j=1}^{M} a_{ij} p(x_{t+1}|j) \beta_{t+1}(j), \quad \forall i = 1, \ldots, M$$

We can visualize this on a trellis

Backward algorithm on a trellis



$$\beta_t(i) = \sum_{j=1}^{M} \beta_{t+1}(j) a_{ij} p(x_{t+1} \mid j)$$

$$\beta_2(2) = \beta_3(1) a_{21} p(x_3 \mid 1) + \beta_3(2) a_{22} p(x_3 \mid 2) + \beta_3(3) a_{23} p(x_3 \mid 3)$$

Posterior probabilities I

- For the EM algorithm, we need to sum over exponentially many $\sum_{y'_{1:T}} p(y'_{1:T}|\mathbf{x}_{1:T}, \lambda^{old}) \log p(\mathbf{x}_{1:T}, y'_{1:T}|\lambda)$, but both terms in the sum can be factorized due to the graphical model of the HMM
- Using the forward-backward algorithm we obtain local posteriors:

$$\xi_t(i,j) = p(y_{t-1} = i, y_t = j|\mathbf{x}_{1:T}, \lambda^{old})$$

and

$$\gamma_t(j) = p(y_t = j|\mathbf{x}_{1:T}, \lambda^{old})$$

then it is easy to maximize the auxiliary function $Q(\lambda, \lambda^{old})$ which factorizes as follows [Bishop, 2006]

$$\sum_{j=1}^{M} \gamma_1(j) \log \pi_j + \sum_{t=2}^{T} \sum_{i,j=1}^{M} \xi_t(i,j) \log a_{ij} + \sum_{t=1}^{T} \sum_{j=1}^{M} \gamma_t(j) \log p(\mathbf{x}_t|j)$$

Posterior probabilities II

- Once we can obtain the posterior probabilities using previous iteration's parameters (λ^{old}), we can update the emission parameters using $\gamma_t(j)$ and transition parameters using $\xi_t(i,j)$
- We can obtain these two sets of variables using forward-backward probabilities
- After performing one forward and one backward pass, we have all α and β parameters

Then,

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{p(x_{1:T}|\lambda)}$$

and

$$\xi_t(i,j) = \frac{\alpha_{t-1}(i)a_{ij}p(x_t|j)\beta_t(j)}{p(x_{1:T}|\lambda)}$$

Updating the parameters I

- Assume there is only a single training sequence $(x_{1:T})$
- After $\gamma_t(j)$ and $\xi_t(i,j)$ parameters are found, the parameter estimation becomes like a weighted counting procedure
- For transition parameters $\hat{a}_{ij} = \dfrac{\sum_{t=2}^{T} \xi_t(i,j)}{\sum_{t=2}^{T} \sum_{j=1}^{M} \xi_t(i,j)}$
- For emission parameters:
  - Discrete case: $p(x|j) := \dfrac{\sum_{t=1}^{T} \gamma_t(j) \delta_{x_t,x}}{\sum_{t=1}^{T} \gamma_t(j)}$

Updating the parameters II

- Gaussian case: The means and variances are updated using weighted sample averages where weights are $\gamma_t(j)$ for each state j
- So, when there is one training sequence, mean update is as follows

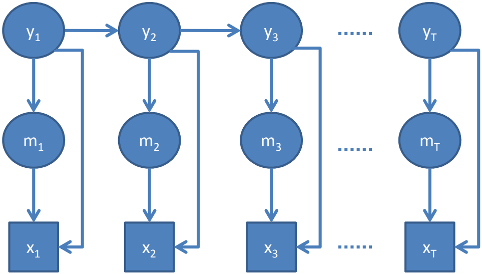
$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(j)}$$

- And the covariance update is similarly

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{x}_t\mathbf{x}_t^T}{\sum_{t=1}^{T} \gamma_t(j)} - \hat{\boldsymbol{\mu}}_j\hat{\boldsymbol{\mu}}_j^T$$

Gaussian mixture observations I

- Gaussian mixture model (GMM) distributions are used a lot in HMMs (e.g. for speech recognition)
- The emission probabilities are represented as a GMM



- $p(\mathbf{x}|y) = \sum_m p(\mathbf{x}|m, y)p(m|y) = \sum_m \mathcal{N}(\mathbf{x}; \mu_{y,m}, \Sigma_{y,m})c_{y,m}$

Gaussian mixture observations II

- The emission parameter updates will depend on mixture posteriors

$$
\begin{aligned}
\gamma_t(j,m) &= p(y_t = j, m_t = m | \mathbf{x}_{1:T}) \\
&= p(y_t = j | \mathbf{x}_{1:T}) p(m_t = m | y_t = j, \mathbf{x}_{1:T}) \\
&= \gamma_t(j) \frac{c_{j,m} p(\mathbf{x}_t | j, m)}{\sum_{m'} c_{j,m'} p(\mathbf{x}_t | j, m')}
\end{aligned}
$$

- Then, when there is a single sequence for training, mean updates will be as follows:

$$
\hat{\boldsymbol{\mu}}_{j,m} = \frac{\sum_{t=1}^{T} \gamma_t(j,m) \mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(j,m)}
$$

- (co)variances can be updated in a similar fashion

Other related models

- Hidden Semi-Markov models: assigns a single label to a segment instead of labeling each observation separately, enables explicit duration model
- Factorial HMM: multiple states explain the observation at the same time
- Multi-stream HMM: the observations are handled in separate streams each of which are independently modeled by a different emission model
- Coupled HMM: two state sequences generate two streams, they interact through their states

References I

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Daniel Jurafsky and James H Martin. *Speech and language processing (second edition)*. Prentice Hall, New Jersey, 2008.

Lawrence R Rabiner. A tutorial on hidden Markov models and selected application in speech recognition. *Proc. IEEE*, 77(2):257–285, February 1989.