CS540 Introduction to Artificial Intelligence

Lecture 10

Young Wu

Based on lecture slides by Jerry Zhu, Yingyu Liang, and Charles Dyer

June 23, 2022

Special Bayesian Network for Sequences

Motivation

- A sequence of features $X_1, X_2, ...$ can be modeled by a Markov Chain but they are not observable.
- A sequence of labels $Y_1, Y_2, ...$ depends only on the current hidden features and they are observable.
- This type of Bayesian Network is called a Hidden Markov Model.

HMM Applications Part 1

Motivation

- Weather prediction.
- Hidden states: $X_1, X_2, \ldots$ are weather that is not observable by a person staying at home (sunny, cloudy, rainy).
- Observable states: $Y_1, Y_2, \ldots$ are Badger Herald newspaper reports of the condition (dry, dryish, damp, soggy).
- Speech recognition.
- Hidden states: $X_1, X_2, \ldots$ are words.
- Observable states: $Y_1, Y_2, \ldots$ are acoustic features.

HMM Applications Part 2

Motivation

- Stock or bond prediction.
- Hidden states: $X_1, X_2, \ldots$ are information about the company (profitability, risk measures).
- Observable states: $Y_1, Y_2, \ldots$ are stock or bond prices.
- Speech synthesis: Chatbox.
- Hidden states: $X_1, X_2, \ldots$ are context or part of speech.
- Observable states: $Y_1, Y_2, \ldots$ are words.

Other HMM Applications

Motivation

- Machine translation.
- Handwriting recognition.
- Gene prediction.
- Traffic control.

Transition and Likelihood Matrices

Motivation

- An initial distribution vector and two-state transition matrices are used to represent a hidden Markov model.

1. Initial state vector: π.

$$\pi_i = \mathbb{P}\left\{X_1 = i\right\}, i \in 1, 2, ..., |X|$$

2. State transition matrix: A.

$$A_{ij} = \mathbb{P}\left\{X_t = j | X_{t-1} = i\right\}, i, j \in 1, 2, ..., |X|$$

3. Observation Likelihood matrix (or output probability distribution): B.

$$B_{ij} = \mathbb{P}\left\{Y_t = i | X_t = j\right\}, i \in 1, 2, ..., |Y|, j \in 1, 2, ..., |X|$$

Markov Property

Motivation

- The Markov property implies the following conditionally independent property.

$$\mathbb{P}\left\{x_t \middle| x_{t-1}, x_{t-2}, \ldots, x_1\right\} = \mathbb{P}\left\{x_t \middle| x_{t-1}\right\}$$
$$\mathbb{P}\left\{y_t \middle| x_t, x_{t-1}, \ldots, x_1\right\} = \mathbb{P}\left\{y_t \middle| x_t\right\}$$

Evaluation and Training

Motivation

- There are three main tasks associated with an HMM.

1. Evaluation problem: finding the probability of an observed sequence given an HMM: $y_1, y_2, \ldots$
2. Decoding problem: finding the most probable hidden sequence given the observed sequence: $x_1, x_2, \ldots$
3. Learning problem: finding the most probable HMM given an observed sequence: $\pi, A, B, \ldots$

Expectation-Maximization Algorithm

Description

- Start with a random guess of π, A, B.
- Compute the forward probabilities: the joint probability of an observed sequence and its hidden state.
- Compute the backward probabilities: the probability of an observed sequence given its hidden state.
- Update the model π, A, B using Bayes rule.
- Repeat until convergence.
- Sometimes, it is called the Baum-Welch Algorithm.

Evaluation Problem

Definition

- The task is to find the probability $\mathbb{P}\left\{y_1, y_2, ..., y_T \middle| \pi, A, B\right\}$.

$$\begin{aligned}
&\mathbb{P}\left\{y_1, y_2, ..., y_T \middle| \pi, A, B\right\} \\
&= \sum_{x_1, x_2, ..., x_T} \mathbb{P}\left\{y_1, y_2, ..., y_T \middle| x_1, x_2, ..., x_T\right\} \mathbb{P}\left\{x_1, x_2, ..., x_T\right\} \\
&= \sum_{x_1, x_2, ..., x_T} \left(\prod_{t=1}^{T} B_{y_t x_t}\right) \left(\pi_{x_1} \prod_{t=2}^{T} A_{x_{t-1} x_t}\right)
\end{aligned}$$

- This is also called the Forward Algorithm.

Decoding Problem

Definition

- The task is to find $x_1, x_2, ..., x_T$ that maximizes $\mathbb{P}\left\{x_1, x_2, ..., x_T \middle| y_1, y_2, ..., y_T, \pi, A, B\right\}$.
- Direct computation is too expensive.
- Dynamic programming needs to be used to save computation.
- This is called the Viterbi Algorithm.

Viterbi Algorithm Value Function

Definition

- Define the value functions to keep track of the maximum probabilities at each time t and for each state k.

$$
\begin{aligned}
V_{1,k} &= \mathbb{P}\left\{y_1 | X_1 = k\right\} \cdot \mathbb{P}\left\{X_1 = k\right\} \\
&= B_{y_1 k} \pi_k \\
V_{t,k} &= \max_x \mathbb{P}\left\{y_t | X_t = k\right\} \mathbb{P}\left\{X_t = k | X_{t-1} = x\right\} V_{t-1,k} \\
&= \max_x B_{y_t k} A_{kx} V_{t-1,k}
\end{aligned}
$$

Viterbi Algorithm Policy Function

Definition

- Define the policy functions to keep track of the x_t that maximizes the value function.

$$\text{policy}_{t,k} = \operatorname*{argmax}_{x} B_{y_t k} A_{kx} V_{t-1,k}$$

- Given the policy functions, the most probable hidden sequence can be found easily.

$$x_T = \operatorname*{argmax}_{x} V_{T,x}$$
$$x_t = \text{policy}_{t+1,x_{t+1}}$$

Dynamic Programming Diagram

Definition

Viterbi Algorithm Diagram

Definition

Expectation-Maximization Algorithm (for HMM), Part 1

Algorithm

- Initialize the hidden Markov model.

$$\pi \sim \text{D}\left(|X|\right), A \sim \text{D}\left(|X|, |X|\right), B \sim \text{D}\left(|Y|, |X|\right)$$

- Perform the forward pass.

$$\alpha_{i,t} \text{ represents } \mathbb{P}\left\{y_1, y_2, ..., y_t, X_t = i | \pi, A, B\right\}$$

$$\alpha_{i,1} = \pi_i B_{y_1,i}$$

$$\alpha_{i,t+1} = \sum_{j=1}^{|X|} \alpha_{j,t} A_{ji} B_{y_{t+1} i}$$

Expectation-Maximization Algorithm (for HMM), Part 2

Algorithm

- Perform the backward pass.

$\beta_{i,t}$ represents $\mathbb{P}\left\{y_{t+1}, y_{t+2}, ..., y_T \middle| X_t = i, \pi, A, B\right\}$

$$\beta_{i,T} = 1$$

$$\beta_{i,t} = \sum_{j=1}^{|X|} A_{ij} B_{y_{t+1} j} \beta_{j,t+1}$$

Expectation-Maximization Algorithm (for HMM), Part 3

Algorithm

- Define the conditional hidden state probabilities for each training sequence n.

$$\gamma_{n,i,t} = \text{ represents } \mathbb{P}\left\{X_t = i \middle| y_1, y_2, ..., y_T, \pi, A, B\right\}$$

$$\gamma_{n,i,t} = \dfrac{\alpha_{i,t}\beta_{i,t}}{\displaystyle\sum_{j=1}^{|X|} \alpha_{j,t}\beta_{j,t}}$$

Expectation-Maximization Algorithm (for HMM), Part 4

Algorithm

- Define the conditional hidden state probabilities for each training sequence n.

$\xi_{n,i,j,t}$ represents $\mathbb{P}\left\{X_t = i, X_{t+1} = j \middle| y_1, y_2, ..., y_T, \pi, A, B\right\}$

$$\xi_{n,i,j,t} = \frac{\alpha_{i,t} A_{ij} \beta_{j,t+1} B_{y_{t+1} j}}{\displaystyle\sum_{k=1}^{|X|} \sum_{l=1}^{|X|} \alpha_{k,t} A_{kl} \beta_{l,t+1} B_{y_{t+1} w}}$$

Expectation-Maximization Algorithm (for HMM), Part 5

Algorithm

- Update the model.

$$\pi'_i = \frac{\sum_{n=1}^{N} \gamma_{n,i,1}}{N}$$

$$A'_{ij} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T-1} \xi_{n,i,j,t}}{\sum_{n=1}^{N} \sum_{t=1}^{T-1} \gamma_{n,i,t}}$$

Expectation-Maximization Algorithm (for HMM), Part 6

Algorithm

- Update the model, continued.

$$B'_{ij} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{1}_{\{y_{n,t}=j\}} \gamma_{n,i,t}}{\sum_{n=1}^{N} \sum_{t=1}^{T} \gamma_{n,i,t}}$$

- Repeat until π, A, B converge.

Dynamic System

Motivation

- The hidden units are used as the hidden states.
- They are related by the same function over time.

$$h_{t+1} = f(h_t, w)$$
$$h_{t+2} = f(h_{t+1}, w)$$
$$h_{t+3} = f(h_{t+2}, w)$$
$$\ldots$$

Dynamic System with Input

Motivation

- The input units can also drive the dynamics of the system.
- They are still related by the same function over time.

$$
\begin{aligned}
h_{t+1} &= f\left(h_t, x_{t+1}, w\right) \\
h_{t+2} &= f\left(h_{t+1}, x_{t+2}, w\right) \\
h_{t+3} &= f\left(h_{t+2}, x_{t+3}, w\right) \\
&\ldots
\end{aligned}
$$

Dynamic System with Output

Motivation

- The output units only depend on the hidden states.

$$y_{t+1} = f\left(h_{t+1}\right)$$
$$y_{t+2} = f\left(h_{t+2}\right)$$
$$y_{t+3} = f\left(h_{t+3}\right)$$
$$\dots$$

Dynamic System Diagram

Motivation

Recurrent Neural Network Structure Diagram

Motivation

Activation Functions

Definition

- The hidden layer activation function can be the tanh activation, and the output layer activation function can be the softmax function.

$$
\begin{aligned}
z_t^{(x)} &= W^{(x)} x_t + W^{(h)} a_{t-1}^{(x)} + b^{(x)} \\
a_t^{(x)} &= g\left(z_t^{(x)}\right), g\left(\boxed{\cdot}\right) = \text{ tanh } \left(\boxed{\cdot}\right) \\
z_t^{(y)} &= W^{(y)} a_t^{(x)} + b^{(y)} \\
a_t^{(y)} &= g\left(z_t^{(y)}\right), g\left(\boxed{\cdot}\right) = \text{ softmax } \left(\boxed{\cdot}\right)
\end{aligned}
$$

Cost Functions

Definition

- Cross entropy loss is used with softmax activation as usual.

$$C_t = H\left(y_t, a_t^{(y)}\right)$$

$$C = \sum_t C_t$$

Multiple Sequential Data Notations

Definition

- There could multiple sequences in the training set index by $i = 1, 2, ..., n$. For one training instance, at time t, there are m features.
- x_{ijt} is the feature j of instance i at time t (position t of the sequence).
- y_{ijt} is the output j of instance i at time t (position t of the sequence).

Multiple Sequential Activations Notations

Definition

- $z_{ijt}^{(x)}$ denotes the linear part of instance i unit j at time t in the hidden layer.
- $a_{ijt}^{(x)}$ denotes the activation of instance i unit j at time t in the hidden layer.
- $z_{ijt}^{(y)}$ denotes the linear part of instance i output j at time t in the output layer.
- $a_{ijt}^{(y)}$ denotes the activation of instance i output j at time t in the output layer

Multiple Sequential Weights Notations, Part 1

Definition

- There are weights and biases between the input layer and the hidden layer, between the hidden layer and the output layer, as in usual neural networks.
- $w_{j'j}^{(x)}, j' = 1, ..., m, j = 1, ..., m^{(h)}$ denotes the weight from input feature j' to hidden unit j.
- $b_{j}^{(x)}, j = 1, ..., m^{(h)}$ denotes the bias of hidden unit j.
- $w_{jj'}^{(y)}, j = 1, ..., m^{(h)}, j' = 1, ..., K$ denotes the weight from hidden unit j to output unit j'.
- $b_{j'}^{(y)}, j' = 1, ..., K$ denotes the bias of output unit j'.

Multiple Sequential Weights Notations, Part 2

Definition

- There are also weights between units within the hidden layer through time.
- $w_{j'j}^{(h)}, j, j' = 1, ..., m^{(h)}$ denotes the weight from hidden unit j' at time t to hidden unit j at time $t + 1$.

BackPropogation Through Time

Definition

- The gradient descent algorithm for recurrent neural networks is called BackPropogation Through Time (BPTT). The update procedure is the same as standard neural networks using the chain rule.

$$w = w - \alpha \frac{\partial C}{\partial w}$$

$$b = b - \alpha \frac{\partial C}{\partial b}$$

Unfolded Network Diagram

Definition

Backpropagation Diagram 1

Definition

Backpropagation Diagram 2

Definition

Backpropagation, Part 1

Definition

- The cost derivative is the same as softmax neural networks.

$$\frac{\partial C}{\partial C_t} = 1$$

$$\frac{\partial C_t}{\partial z_{ijt}^{(y)}} = z_{ijt}^{(y)} - \mathbb{1}_{\{y_{it}=j\}}$$

Backpropagation, Part 2

Definition

- The other derivatives are similar to fully connected neural networks.

$$\frac{\partial z_{ij't}^{(y)}}{\partial a_{ijt}^{(x)}} = w_{jj'}^{(y)}$$

$$\frac{\partial z_{ij't}^{(y)}}{\partial w_{jj'}^{(y)}} = a_{ijt}^{(x)}$$

$$\frac{\partial z_{ij't}^{(y)}}{\partial b_{j'}^{(y)}} = 1$$

Backpropagation, Part 3

Definition

- The other derivatives are similar to fully connected neural networks.

$$\frac{\partial a_{ijt}^{(x)}}{\partial z_{ijt}^{(x)}} = g'\left(z_{ijt}^{(x)}\right) = 1 - \left(a_{ijt}^{(x)}\right)^2$$

$$\frac{\partial z_{ijt}^{(x)}}{\partial w_{j'j}^{(x)}} = x_{ij't}$$

$$\frac{\partial z_{ijt}^{(x)}}{\partial b_{j}^{(x)}} = 1$$

Backpropagation, Part 4

Definition

- The chain rule goes through time, so each gradient involves a long chain of the partial derivatives between $a_t^{(x)}$ and $a_{t-1}^{(x)}$ for $t = 1, 2, ..., T$.

$$\frac{\partial a_{ijt}^{(x)}}{\partial z_{ijt}^{(x)}} = 1 - \left(a_{ijt}^{(x)}\right)^2$$

$$\frac{\partial z_{ijt}^{(x)}}{\partial a_{ij't-1}^{(x)}} = w_{j'j}^{(h)}$$

Vanishing and Exploding Gradient

Discussion

- If the weights are small, the gradient through many layers will shrink exponentially. This is called the vanishing gradient problem.
- If the weights are large, the gradient through many layers will grow exponentially. This is called the exploding gradient problem.
- Fully connected and convolutional neural networks only have a few hidden layers, so vanishing and exploding gradient is not a problem in training those networks.
- In a recurrent neural network, if the sequences are long, the gradients can easily vanish or explode.

RNN Variants

Discussion

- Long Short Term Memory (LSTM): gated units to keep track of long term dependencies.
- Gated Recurrent Unit (GRU): different gated units.
- Transformers (BERT, GPT): no recurrent units, positional encoding, attention mechansim.

Long Term Memory

Discussion

- It is also very hard to detect that the current output depends on an input from many time steps ago.
- Recurrent neural networks have difficulty dealing with long-range dependencies.

Long Short Term Memory

Discussion

- Long Short Term Memory (LSTM) network adds more connected hidden units for memories controlled by gates. The activation functions used for these gates are usually logistic functions.
- An LSTM unit usually contains an input gate, an output gate, and a forget gate, to keep track of the dependencies in the input sequence.

Gated Recurrent Unit

Discussion

- Gated Recurrent Unit (GRU) does something similar to an LSTM unit.
- A GRU contains input and forget gates, and does not contain an output gate.

Transformers

Discussion

- There are no recurrent units, and positional encoding are used instead so that the features contain information about both the word type of the current token and its position.
- Only attention units are used, they are also called scaled dot product attention units: they keep track of which parts of the sentence is important and pay attention to. They can be multiple parallel attention units called multi-head attention.
- Layer normalization trick is used so that the means and variances of the units between attention layers and fully connected layers stay the same.