EE 628
Deep Learning
Fall 2019

Lecture 9

03/26/2020

Sergul Aydore

Applied Scientist

Amazon Web Services

Announcements

- Midterms are graded
- Graded pdfs are pushed to your repositories
- Deadline for Project Proposals: **04/03/2020 Friday at 5pm ET**
  - This includes creation of a github repository with READ.md file that contains the summary of the project.
  - Late submissions or repositories with empty READ.md file will lose 30 points from their grade for the project.
  - Email me the link to your github repository before deadline
- Deadline for Projects **04/27/2020 Monday at 5pm ET**
  - project presentation on **04/30/2020** (40 %)
  - Details on grading: https://github.com/sergulaydore/EE-628-Spring-2020

Overview

- Last lecture we covered
  - Modern Convolutional Neural Networks
- Today, we will cover
  - Text Processing
  - Recurrent Neural Networks
- Source material:
  - Dive into Deep Learning (https://d2l.ai/)
  - https://github.com/sergulaydore/EE-628-Spring-2020

Recurrent Neural Networks

- So far, we encountered two types of data: generic vectors and images.
- Most importantly, we assume that our data is generated iid.
- This is not true for most data.
- For instance, the words written here are in sequence, and would be hard to understand if you permute randomly.
- Likewise, image frames in a video, the audio signal in a conversation, or the browsing behavior on a website, all follow sequential order.
- Thus we need specialized models for such data.

Recurrent Neural Networks

- Also, we might not only receive a sequence as an input but rather might be expected to continue the sequence.
- While CNNs process spatial information, RNNs are designed to better handle sequential information.
- These networks introduce state variables to store past information and, together with the current input, determine the current output.
- Many of the examples for using RNNs are based on text data. Hence, we will emphasize language models in this lecture.

Statistical Tools

- We need statistical tools and new deep networks architectures to deal with sequence data.
- To keep things simple, we use the stock price as an example.
- Let's denote the prices by $x_t \geq 0$.
- For traders to do well in the stock market on day t, they want to predict x_t via

$$x_t \sim p(x_t | x_{t-1}, \ldots x_1).$$

Autoregressive Models

- Our trader could use a regressor but there is a major problem.
- The number of inputs $x_1, \ldots, x_{t-1}$ varies, depending on t.
- There are two strategies to address this:
  1. We might content ourselves with some timespan τ and only use $x_{t-1}, \ldots, x_{t-\tau}$ observations. Such models are called autoregressive models as they perform regression on themselves.
  2. Keep some summary h_t of the past observations around and update that in addition to the actual prediction. This leads to models that estimate $p(x_t, x_{t-1}, h_{t-1})$ and moreover updates of the form $h_t = g(h_{t-1}, x_{t-1})$. These models are called latent autoregressive models. LSTMs and GRUs are examples of this.

Autoregressive Models

- Both cases raise the question how to generate training data.
- A common assumption is that while the specific values of x_t might change, at least the dynamics of the time series itself won’t.
- Statisticians call dynamics that don’t change stationary.
- Regardless of what we do, we will thus get an estimate of the entire time series via

$$p(x_1, \ldots x_T) = \prod_{t=1}^{T} p(x_t | x_{t-1}, \ldots x_1).$$

Markov Model

- Recall the approximation that in an autoregressive model we use only $(x_{t-1}, \dots, x_{t-\tau})$ instead of $(x_{t-1}, \dots, x_1)$ to estimate x_t.
- Whenever this approximation is accurate we say the sequence satisfies a Markov condition.
- In particular, if $\tau = 1$, we have a first order Markov model and $p(x)$ is given by

$$p(x_1, \dots x_T) = \prod_{t=1}^{T} p(x_t | x_{t-1}).$$

- Such models are particularly nice since dynamic programming can be used to compute values along the chain exactly.

$$p(x_{t+1} | x_{t-1}) = \sum_{x_t} p(x_{t+1} | x_t) p(x_t | x_{t-1}).$$

Causality

- In principle, there's nothing wrong with unfolding $p(x_1, \dots, x_T)$ in reverse order. After all, by conditioning we can always write it via

$$p(x_1, \dots x_T) = \prod_{t=T}^{1} p(x_t | x_{t+1}, \dots x_T).$$

- In many cases, however, there exists a natural direction for the data, namely going forward in time.
- It is clear that future events cannot influence the past.

Text Preprocessing

- Text is an important example of sequence data.
- Common preprocessing steps for text data consists of:
    1. Loads texts as strings into memory
    2. Splits strings into tokens, a token could be a word or a character
    3. Builds a vocabulary for these tokens to map them into numerical indices
    4. Maps all tokens in the data into indices to facilitate to feed into models

Open notebook
IN_CLASS Text_Data_Example.ipynb

Language Models and Data Sets

- Tokens can be viewed as a time series of discrete observations.
- Assuming the tokens in a text of length T are in turn $x_1, x_2, \ldots, x_T$, then, in the discrete time series, x_t can be considered as the output or label of time step t.
- Given such a sequence, the goal of a language model is to estimate the probability

$$p(x_1, x_2, \ldots, x_T).$$

- Language models are incredibly useful. For instance, an ideal language model would be able to generate a natural text just on its own by drawing one word at a time

$$w_t \sim p(w_t | w_{t-1}, \ldots w_1).$$

Estimating a language model

- The obvious question is how we should model a document, or even a sequence of words.
- Let’s start by applying basic probability rules:

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_1, \ldots, w_{t-1}).$$

- For example, the probability of a text sequence containing four tokens consisting of words and punctuation would be given as:

$$p(\text{Statistics}, \text{is}, \text{fun}, .) = p(\text{Statistics})p(\text{is}|\text{Statistics})p(\text{fun}|\text{Statistics}, \text{is})p(.|\text{Statistics}, \text{is}, \text{fun})$$

- In order to compute the language model, we need to calculate the probability of a word given the previous few words.

Estimating a language model

- The probability of words can be calculated from the relative word frequency of a given word in the training dataset.
- For example, $p(\text{Statistics})$ can be calculated as the count of all occurrences of the word ‘statistics’ and divide it by the total number of words in the corpus. This works fairly well, particularly for frequent words.
- Moving on, we could attempt to estimate $\hat{p}(\text{is}|\text{Statistics}) = \dfrac{n(\text{Statistics is})}{n(\text{Statistics})}.$
- Estimating the probability of a word pair is somewhat difficult, since the occurrences of ‘Statistics is’ are a lot less frequent. Things get worse for 3 word combinations.

Estimating a language model

- Laplace Smoothing: add a small constant to all counts.
- Unfortunately, models like this get unwieldy rather quickly:
  - we need to store all counts
  - This entirely ignores the meaning of the words

Markov Models and n-grams

- Recall our discussion of Markov models.
- Let's apply this to language modeling.
- A distribution over sequences satisfies the Markov property of first order if $p(w_{t+1}|w_t, \ldots w_1) = p(w_{t+1}|w_t)$.
- Higher orders correspond to longer dependencies.
- This leads to a number of approximations that we could apply to a model sequence:

$$p(w_1, w_2, w_3, w_4) = p(w_1)p(w_2)p(w_3)p(w_4) \longrightarrow \text{unigram}$$

$$p(w_1, w_2, w_3, w_4) = p(w_1)p(w_2|w_1)p(w_3|w_2)p(w_4|w_3) \longrightarrow \text{bigram}$$

$$p(w_1, w_2, w_3, w_4) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)p(w_4|w_2, w_3) \longrightarrow \text{trigram}$$

Open IN_CLASS_Natural_Language_Statistics.ipynb

Open IN_CLASS_Training_Data_Preparation.ipynb

Recurrent Neural Networks

- In n-gram models, conditional probability of word x_t depends on the n-1 previous words.
- If we want to check the possible effect of words earlier than $t-(n-1)$ on x_t, we need to increase n.
- However, the number of parameters would also increase with exponentially with it.
- Hence, rather than modeling $p(x_t|x_{t-1}, \dots, x_{t-n+1})$, it is preferable to use a latent variable model in which we have

$$p(x_t|x_{t-1}, \dots x_1) \approx p(x_t|x_{t-1}, h_t).$$

Recurrent Neural Networks

- Here h_t is the latent variable that stores the sequence information.
- A latent variable is also called as hidden variable, hidden state or hidden state variable.
- The hidden state at time t could be computed based on both input x_{t-1} and hidden state h_{t-1} at time $t-1$, that is

$$h_t = f(x_{t-1}, h_{t-1}).$$

- Recurrent neural networks are neural networks with hidden states.

Recurrent Neural Networks with Hidden States

- Assume we have $\mathbf{X}_t \in R^{n \times d}, t = 1, \dots, T$.
- And $\mathbf{H}_t \in R^{n \times h}$ is the hidden variable of time step t from the sequence.
- Unlike the MLP, here we save the hidden variable $\mathbf{H}_{t-1}$ from the previous step and introduce a new weight parameter $\mathbf{W}_{hh} \in R^{h \times h}$.
- Specifically, the calculation of the hidden variable of the current step is determined by

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h).$$

- Since the hidden state uses the same definition of the previous time step in the current time step, the computation of the equation above is recurrent, hence the name recurrent neural network (RNN).

Recurrent Neural Networks with Hidden States

- There are many different RNN construction methods.
- RNNs with a hidden state defined by the equation in the previous slide are very common.
- For time step t, the output of the output layer is similar to the computation in the MLP:

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$

- RNN parameters include the weight $\mathbf{W}_{xh} \in R^{d \times h}$, $\mathbf{W}_{hh} \in R^{h \times h}$, $\mathbf{b}_h \in R^{1 \times h}$, $\mathbf{W}_{hq} \in R^{h \times q}$, $\mathbf{b}_q \in R^{1 \times q}$
- RNNs always use these model parameters, even for different time steps. Therefore, the number of RNN model parameters does not grow as the number of time steps increases.

Recurrent Neural Networks with Hidden States

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h). \qquad \mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$

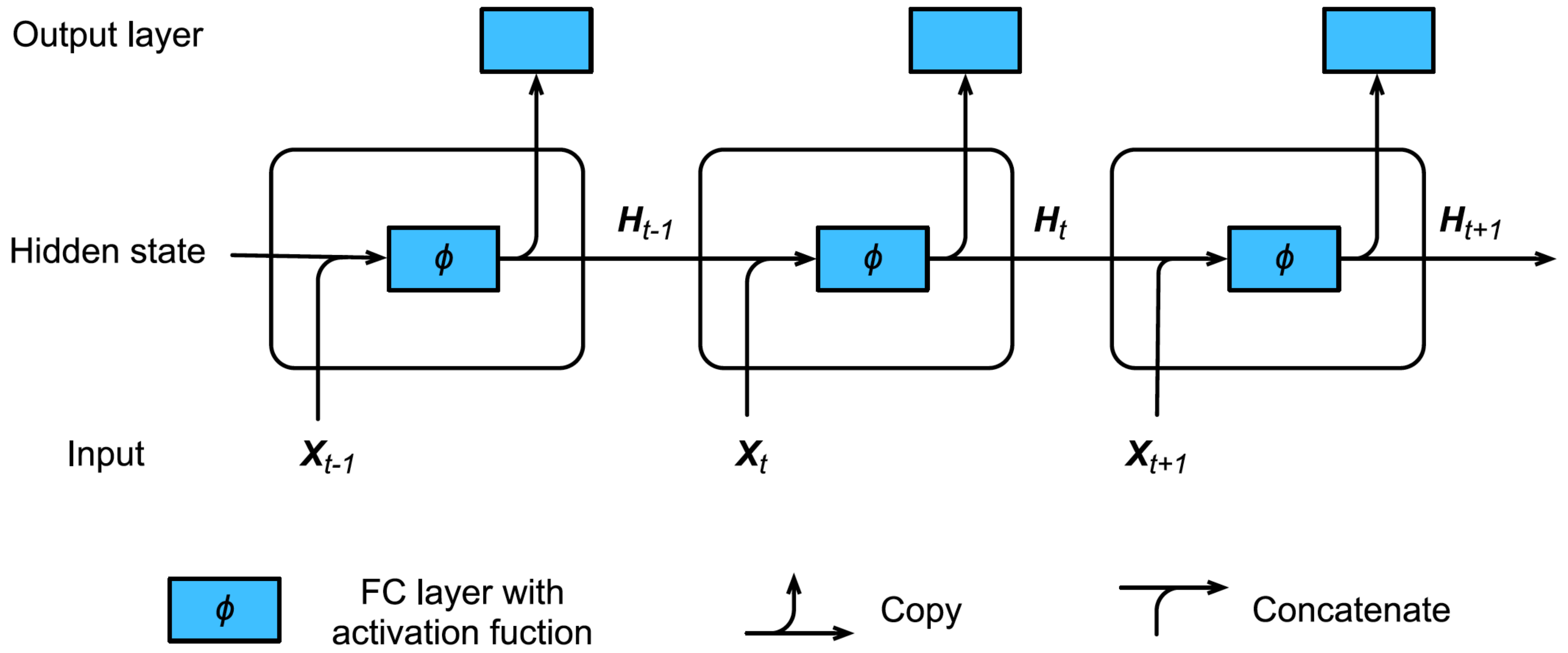


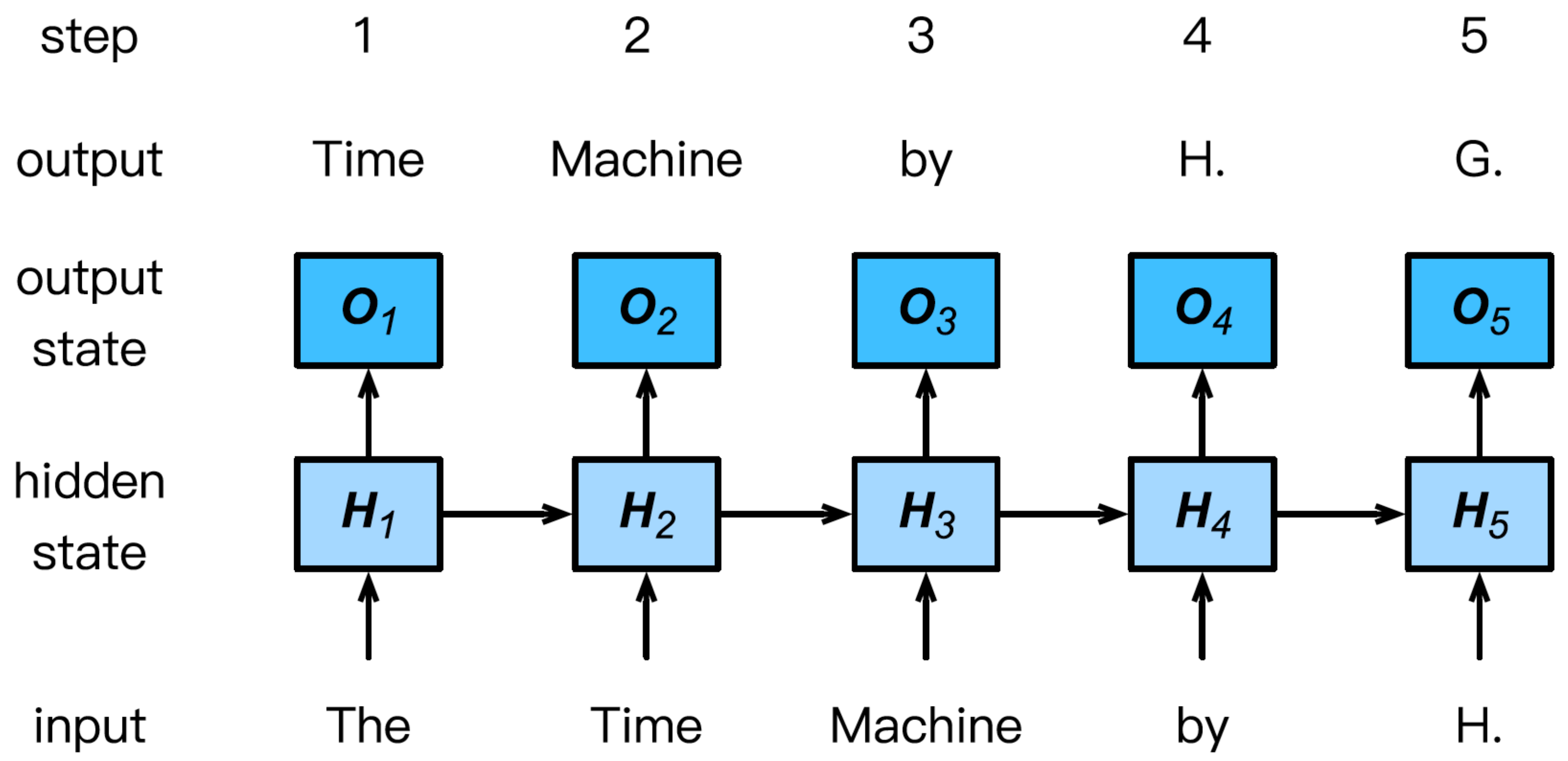
Fig. 10.4.1: An RNN with a hidden state.

Steps in a Language Modeling

- Now we illustrate how RNNs can be used to build a language model.
- Let the number of mini-batch examples be 1, and the sequence of the text be the beginning of our dataset*: the time machine by h. g. wells*

Steps in a Language Modeling

- Now we illustrate how RNNs can be used to build a language model.
- Let the number of mini-batch examples be 1, and the sequence of the text be the beginning of our dataset*: the time machine by h. g. wells*



Perplexity

- Last, let’s discuss about how to measure the sequence model quality.
- One way is to check how surprising the text is.
- A good language model is able to predict with high accuracy what we will see next.
- Consider the following continuations of the phrase *It is raining*, as proposed by different language models:

1. `It is raining outside`
2. `It is raining banana tree`
3. `It is raining piouw;kcj pwepoiut`

Perplexity

- We can measure the quality of the model by

$$\frac{1}{n}\sum_{t=1}^{n} -\log p(x_t|x_{t-1},\ldots x_1)$$

- For historical reasons scientists in natural language processing prefer to use a quantity called perplexity

$$\mathrm{PPL} := \exp\left(-\frac{1}{n}\sum_{t=1}^{n}\log p(x_t|x_{t-1},\ldots x_1)\right)$$

- What are the best, worst and baseline cases?

Open notebook IN_CLASS_implementation_of_RNNs_from_scratch