Sequential Machine Learning

Lecturer: Haim Permuter *Scribe: Ziv Aharoni*

Throughout this lecture we discuss the task of modeling sequential data. This task comprises on overcoming the fact that training examples are no longer i.i.d. Therefore, we have to build models that are capable of modeling the time dependencies between samples in order to model the data optimally in the sense of maximum likelihood. First, we define the problem mathematically and derive the necessity of Recurrent Neural Network (RNN) models. Then, we will elaborate on two types of Recurrent Neural Networks (RNNs): the Elman Network and the Long-Short-Term-Memory (LSTM) cell.

I. Introduction

Given a training set of $n \in \mathbb{N}$ examples $\{(x_i, y_i)\}_{i=1}^n$ drawn from the joint probability P_{X^n,Y^n}, we want to model the generating distribution of the data. In order to do so, we seek to estimate P_{X^n,Y^n} with a parametric model $Q^\theta_{X^n,Y^n}$, whose parameters are denoted by θ.

We distinct the sample elements into two groups: (1) the features X, those elements that we sample freely from the environment, and (2) the labels Y, those elements that we can not sample from the environment and seek to predict. We now decompose the joint probability, namely P_{X^n,Y^n} , using the chain rule of probabilities as follows,

$$P_{X^n,Y^n}(x^n, y^n) = \prod_{i=1}^{n} P_{X_i,Y_i|X^{i-1},Y^{i-1}}\left(x_i, y_i|x^{i-1}, y^{i-1}\right) \tag{1}$$

$$= \prod_{i=1}^{n} P_{X_i|X^{i-1},Y^{i-1}}\left(x_i|x^{i-1}, y^{i-1}\right) P_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i, y^{i-1}\right). \tag{2}$$

Generally, want to model only $P_{Y_i|X^i,Y^{i-1}}$ since we sample X^n freely from the real distribution, i.e $P_{X_i|X^{i-1},Y^{i-1}}$. Hence we can model $Q^\theta_{X^n,Y^n}(x^n, y^n)$ by

$$Q^\theta_{X^n,Y^n}(x^n, y^n) = \prod_{i=1}^{n} P_{X_i|X^{i-1},Y^{i-1}}\left(x_i|x^{i-1}, y^{i-1}\right) Q^\theta_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i, y^{i-1}\right). \tag{3}$$

In the rest of the lecture we use this setting, but the derivation could be extended to estimating the term $P_{X_i|X^{i-1},Y^{i-1}}\left(x_i|x^{i-1},y^{i-1}\right)$ as well.

Now, we find the model parameters θ by the maximum likelihood estimator which is given by

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}} \left\{\log Q^{\theta}_{X^n,Y^n}\left(x^n,y^n\right)\right\} \tag{4}$$

$$= \underset{\theta}{\operatorname{argmax}} \left\{\log \prod_{i=1}^{n} P_{X_i|X^{i-1},Y^{i-1}}\left(x_i|x^{i-1},y^{i-1}\right) Q^{\theta}_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i,y^{i-1}\right)\right\} \tag{5}$$

$$= \underset{\theta}{\operatorname{argmax}} \left\{\sum_{i=1}^{n} \log Q^{\theta}_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i,y^{i-1}\right)\right\} \tag{6}$$

If the samples were sampled i.i.d the term $Q^{\theta}_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i,y^{i-1}\right)$ would collapse to $Q^{\theta}_{Y_i|X_i}\left(y_i|x_i\right)$, which is feasible to approximate with a parametric model. Unfortunately, when modeling $Q^{\theta}_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i,y^{i-1}\right)$, there are two major difficulties with the modeling task: (1) the model needs more parameters to combine the features from past times, and (2) the function $Q^{\theta}_{Y_i|X^i,Y^{i-1}}$ varies (even in the number of arguments) as i changes.

For example, if we use a logistic model, in the i.i.d case, if $x \in \mathbb{R}^d, y \in \mathbb{R}$, we can parameterize $Q^{\theta}_{Y_i|X_i}\left(y_i|x_i\right)$ the model with $d+1$ parameters. However, in the time dependent case, if we want to model $Q^{\theta}_{Y_i|X^i,Y^{i-1}}$ we would need $di + d(i-1) + 1$ parameters.

These problems encourage us to make assumptions on the distribution that generated the data, that eventually lead us to build feasible models which are capable on encapsulating time dependencies. In the next section we develop a model with shared weights by approximating the underlying distribution of the data as Markov process.

II. Stationary Markov Process Modeling

In order to deal with the parameterization problem, we approximate the process that generated X^n, Y^n as a stationary Markov process.

Let us assume that there exists a R.V $S_i \triangleq f(X^{i-1}, Y^{i-1})$, for some deterministic function $f(\cdot)$, that summarizes the history of inputs and labels, such that, $P_{X_i,Y_i|X^{i-1},Y^{i-1}} = P_{X_i,Y_i|S_i}$. That is, we assume that the following Markov chain holds,

$$(S_1, X_1, Y_1) - (S_2, X_2, Y_2) - \cdots - (S_n, X_n, Y_n). \tag{7}$$

Next, we assume stationarity of this Markov chain, that is,

$$\mathbb{P}(X_i = x, Y_i = y, S_i = s) = \mathbb{P}(X_j = x, Y_j = y, S_j = s) \quad i \neq j, \tag{8}$$

where $(x, y, s) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{S}$, the alphabet of inputs, targets and states respectively. The last assumption is that this stationary Markov chain is Ergodic, which basically means that we could recover the stationary distribution of the Markov chain by sampling observation of the data.

Under the three preceding assumptions, and by the law of large numbers, the optimization criteria in (6) becomes

$$\sum_{i=1}^{n} \log Q^{\theta}_{Y_i|X^i,Y^{i-1}}\left(y_i|x^i, y^{i-1}\right) = \tag{9}$$

$$\sum_{i=1}^{n} \log Q^{\theta}_{Y_i|X_i,S_{i-1}}\left(y_i|x_i, s_{i-1}\right) \xrightarrow{n\to\infty} \tag{10}$$

$$\mathbb{E}_{P_{Y|X,S}}\left[Q^{\theta}_{Y|X,S}(Y|X,S)\right] = \tag{11}$$

$$H\left(P_{Y|X,S}\right) + D_{KL}\left(P_{Y|X,S} \| Q^{\theta}_{Y|X,S}\right) \tag{12}$$

Using this approximation, we can build a feasible model that would be able to encapsulate time dependencies in the data.

III. Recurrent Neural Network (RNN)

In this section we show the equations of the Elman (vanilla) RNN. Then, we delve into the error propagation analysis of the Elman RNN and explain the vanishing/exploding gradient phenomena, which will motivate us finally derive the architecture of the Long-Short-Memory-Term (LSTM) cell.

A. Elman Network

1) Description: The Elman network uses the Markov settings from the preceding section. That is, the Elman networks generates a state S that summarizes the history and is used to generate along with the input X the prediction for Y and the next state S'.

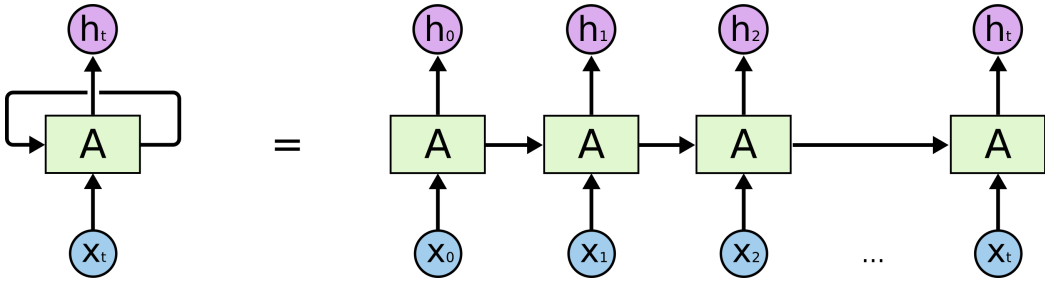

Fig. 1. Depiction of single-layered Elman network and its time unrolling representation

We introduce the equation of a single layered Elman network, as depicted in Figure 1, for simplicity, even though that the generalization for multi-layered network (or stacked Elman network) is direct. The equations of the Elman network are given by

$$z_t = W_x x_t + W_h h_{t-1} + b_x \tag{13}$$

$$h_t = \sigma\left(z_t\right) \tag{14}$$

$$z_t^y = W_y h_t + b_y \tag{15}$$

$$y_t = \sigma\left(z_t^y\right) \tag{16}$$

where $x_t \in \mathbb{R}^d, b_x, z_t, h_t \in \mathbb{R}^m, W_x \in \mathbb{R}^{m\times d}, W_h \in \mathbb{R}^{m\times m}, W_y \in \mathbb{R}^{1\times m}, z_y, y_t, b_y \in \mathbb{R}$ and $h_0^l = \mathbf{0}$. Here the state at time step t is denoted by h_t, the input at time t is denoted by x_t, and the $\sigma(\cdot)$ denote a element-wise non-linearity function. For our analysis we use the sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$.

2) Error Propagation Analysis: Let us denote the error signal by

$$\epsilon_t = d_t - y_t, \tag{17}$$

where d_t denotes the true label and the term y_t denotes the model prediction. For computational simplicity we assume that the target is a scalar. The goal is to minimize the MSE loss, namely $J(\theta)$ which is given by

$$J(\theta) = \sum_{t=1}^{T} J_t(\theta) \tag{18}$$

$$= \sum_{t=1}^{T} \frac{1}{2}\epsilon_t^2 \tag{19}$$

We would like to adjust the network weights by propagating the error back in time. Let us calculate the error signal as it propagates backwards in the network. The error propagated to the network output is denoted by $\delta_t^y \in \mathbb{R}^{1\times n_y}$ and is given by

$$\delta_t^y \triangleq \frac{\partial}{\partial z_t^y} J_t(\theta) \tag{20}$$

$$= \frac{\partial}{\partial z_t^y} \frac{1}{2} \left(d_t - y_t\right)^2 \tag{21}$$

$$= \frac{\partial}{\partial y_t} \frac{1}{2} \left(d_t - y_t\right)^2 \frac{\partial y_t}{\partial z_t^y} \tag{22}$$

$$= \left(d_t - y_t\right) \sigma'\left(z_t^y\right)^T \tag{23}$$

where

$$\sigma'\left(z_t^y\right) = \left[\sigma'(z_{t\,1}^y), \sigma'(z_{t\,2}^y), \ldots, \sigma'(z_{t\,n_y}^y)\right]^T. \tag{24}$$

The error that is propagated to the RNN state is denoted by $\delta_t \in \mathbb{R}^{1\times m}$ and is given by

$$\begin{aligned}
\delta_t &\triangleq \frac{\partial}{\partial z_t} J_t(\theta) \\
&= \frac{\partial}{\partial z_t^y} J_t(\theta) \frac{\partial z_t^y}{\partial z_t} \\
&= \delta_t^y \frac{\partial}{\partial z_t} \left(W_y h_t + b_y\right) \\
&= \delta_t^y W_y \text{diag}\left(\sigma'\left(z_t\right)\right)
\end{aligned} \tag{25}$$

Next, we can calculate the error propagated backwards in time inside the network by $\delta_{t-1} \in \mathbb{R}^{1\times n_L}$ and $\delta_t \in \mathbb{R}^{1\times n}$ respectively. The time propagated error is given by

$$\delta_{t-1} \triangleq \frac{\partial}{\partial z_{t-1}} J_t(\theta) \tag{26}$$

$$= \delta_t \frac{\partial z_t}{\partial z_{t-1}} \tag{27}$$

$$= \delta_t \frac{\partial}{\partial z_{t-1}} \left(W_x x_t + W_h h_{t-1} + b_x \right) \tag{28}$$

$$= \delta_t W_h \text{diag} \left(\sigma' \left(z_{t-1} \right) \right), \tag{29}$$

Now, we calculate the error that is propagated k time steps backwards in time. We will get that

$$\delta_{t-k} = e_t \sigma' \left(z_t^y \right)^T W_y \text{diag} \left(\sigma' \left(z_t \right) \right) W_h \text{diag} \left(\sigma' \left(z_{t-1} \right) \right) \cdots W_h \text{diag} \left(\sigma' \left(z_{t-k} \right) \right) \tag{30}$$

Let us examine the norm of the error as it propagates backwards in time. Since that σ' is bounded, say by $M > 0$, and assume that W_h has maximal eigen value of λ we can claim that

$$\|\delta_{t-k}\| \leq |e_t| \left\| M \mathbf{1}^T W_y \right\| \|M\lambda I\| \cdots \|M\lambda I\| \tag{31}$$

$$= |e_t| \left| (\lambda M)^{2k} \right| \left\| M^2 \mathbf{1}^T W_y \right\| \tag{32}$$

We can see that if $|\lambda M| > 1.0$ the error explodes as we keep propagating the error back in time, and if $|\lambda M| < 1.0$ the error vanishes in time. The naive approach is to set $\lambda M = 1$ but this does not work in practice since it causes saturated units that drives the activation to zero. This property of the Elman network is the main motivation for the LSTM cell, which addresses the vanishing/exploding gradient problem by enforcing constant error propagation in time.

B. Long Short-Term Memory (LSTM)

1) Motivation - Constant Error Flow: For simplicity, let us examine the naive elman RNN with a single unit. In that case, the propagated error from time t to time $t-1$ is given by

$$\delta_{t-1} = \delta_t \frac{\partial z_t}{\partial z_{t-1}} \tag{33}$$

$$= \delta_t \frac{\partial}{\partial z_{t-1}} \left(w_x x_t + w_h h_{t-1} + b \right) \tag{34}$$

$$= \delta_t w_h f' \left(z_{t-1} \right). \tag{35}$$

In order to achieve constant error flow we demand that the activation function will satisfy

$$w_h f'(z_{t-1}) = 1. \tag{36}$$

By solving the differential equation we get that $f(z) = \frac{z}{w_h^l}$, i.e the activation function must be linear. So by setting f to be the identity mapping and $w_h = 1$ we can ensure a constant error flow, namely Constant Error Carousel (CEC). However a unit is connected to other units except itself that introduce two conflicts.

1) **Input weight conflict:** Excitation of input units enter the state by a linear projection and can contaminate the state if the input is not correlated with the targets.
2) **Output weight conflict:** State units are used to predict the target by a linear transformation. Nevertheless, at different time steps different units are correlated with the target. Hence, a linear transformation of the states essentially uses uncorrelated units to predict the target.

These conflicts, and the conclusion that linear activation can enforce constant error flow motivates us for the architecture of the LSTM cell.

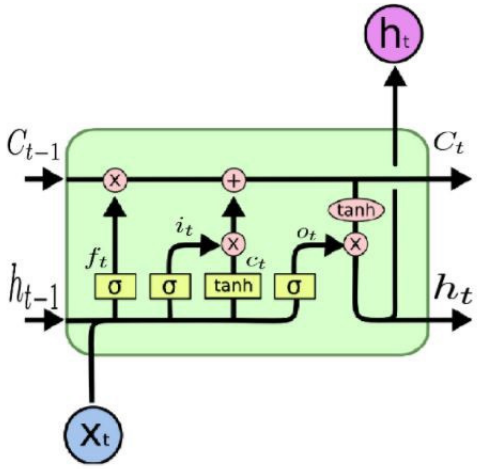

Fig. 2. Depiction of the LSTM cell architecture

2) The Original Model: The architecture of the LSTM is depicted in Figure 2, an its equations are given by

$$a_t = \tanh\left(U_a x_t + W_a h_{t-1} + b_a\right) \tag{37}$$

$$i_t = \sigma\left(U_i x_t + W_i h_{t-1} + b_i\right) \tag{38}$$

$$o_t = \sigma\left(U_o x_t + W_o h_{t-1} + b_o\right) \tag{39}$$

$$c_t = a_t \odot i_t + c_{t-1} \tag{40}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{41}$$

These equations were modified to the last common LSTM architecture by adding a "forget" gate that can forget certain cell units when propagating the error through time.

3) The Current Model: The LSTM equations are given by

$$a_t = \tanh\left(U_a x_t + W_a h_{t-1} + b_a\right) \tag{42}$$

$$i_t = \sigma\left(U_i x_t + W_i h_{t-1} + b_i\right) \tag{43}$$

$$f_t = \sigma\left(U_f x_t + W_f h_{t-1} + b_f\right) \tag{44}$$

$$o_t = \sigma\left(U_o x_t + W_o h_{t-1} + b_o\right) \tag{45}$$

$$c_t = a_t \odot i_t + f_t \odot c_{t-1} \tag{46}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{47}$$

This structure enables constant gradient flow through time by removing the non-linear activation from the memory update as depicted in figure 3, namely $c_t$tre of the LSTM cell enab

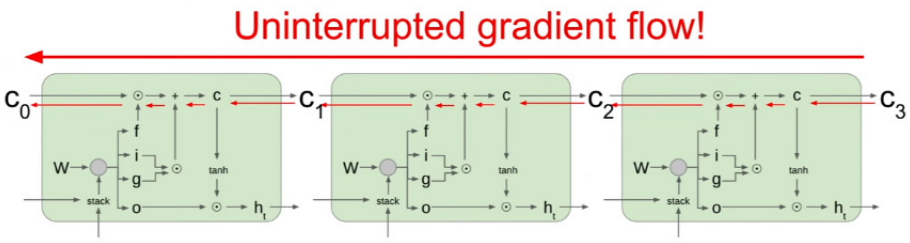

Fig. 3. Depiction of gradient flow in time of the LSTM cell.