Recurrent Neural Networks (RNNs) and the Exploding & Vanishing Gradient Problems

Le Gia Khang	21522189
Bui Manh Hung	21522110
Nguyen Hoang Tan	21521413
Pham Tram Anh	21520587
Le Thi Kim Yen	21521695

University of Information Technology

December 26, 2022

References

The contents of this document are taken mainly from the follow sources:

- Robin M.Schmidt, Recurrent Neural Networks (RNNs): a gentle introductionand overview[1];
- Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training RNNs[2]
- Yoshua Bengio, Patrice Simard, Paolo Frasconi, Learning long-term dependencies with gradients descent is difficult[3]

[1] https://arxiv.org/pdf/1912.05911.pdf

[2] https://arxiv.org/pdf/1211.5063.pdf

[3] https://www.researchgate.net/profile/Y-Bengio/publication/5583935_Learning_long-term_dependencies_with_gradient_descent_is_difficult/links/546b702d0cf2397f7831c03d/Learning-long-term-dependencies-with-gradient-descent-is-difficult.pdf

Table of Contents

Table of Contents

Sequential Data

- Sequential Data refers to any data that contain elements that are ordered into sequences.
- Examples include time series, DNA sequences (see biomedical informatics) and sequences of user actions.

Question ?

Named-entity Recognition for person names:

Harry Potter and Hermione Granger invented a new spell.

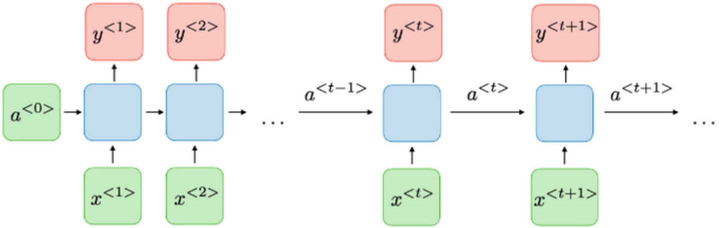
- Machine learning models that input or output data sequences are known as sequence models.

What's Recurrent Neural Networks (RNNs)

- Recurrent Neural Networks (RNNs) are a type of neural network architecture which is mainly used to detect patterns in a sequence of data.
- However, they are also applicable to images if these get respectively decomposed into a series of patches and treated as a sequence.
- What differentiates Recurrent Neural Networks from Feedforward Neural Networks also known as Multi-Layer Perceptrons (MLPs) is how information gets passed through the network.

Model Architecture

- The RNN has cycles and transmits information back into itself. This enables them to extend the functionality of Feedforward Networks to also take into account previous inputs $X_{0:t-1}$ and not only the current input X_t.



Model Architecture

- We denote the hidden state and the input at time step t respecively as $\mathbf{H}_t$ and X_t Further, we use $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$, $\mathbf{W}_{ho}$ and a bias b_h as shared parameters. Lastly, all these informations get passed to a activation function ϕ which is usually a logistic sigmoid or tanh function.

Hidden state at the time step t

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

Output at time step t

$$\mathbf{O}_t = \phi_o(\mathbf{H}_t \mathbf{W}_{ho} + \mathbf{b}_o)$$

Loss Function

- Loss function evaluates performance of the network by comparing the output $\mathbf{y}_t$ with the corresponding target $\mathbf{z}_t$ defined as:

Loss Function

$$L(\mathbf{y}, \mathbf{z}) = \sum_{t=1}^{T} L_t(\mathbf{y_t}, \mathbf{z_t})$$

- Selection of the loss function is problem dependent. Some popular loss function are Euclidean distance and Hamming distance and cross-entropy.

Table of Contents

Initializations

- A general rule is to assign small values to the weights. A Gaussian draw with a standard deviation of 0.001 or 0.01 is a reasonable choice.
- The biases are usually set to zero, but the output bias can also be set to a very small value.
- However, the initialization of parameters is dependent on the task and properties of the input data such as dimensionality.

Gradient-based Learning Methods

- Gradient descent (GD) adjusts the weights of the model by finding the error function derivatives with respect to each member of the weight matrices.

The Gradient Descent or Batch GD

Computing the gradient for the whole dataset in each optimization iteration to perform a single update as

$$\theta_{t+1} = \theta_t - \frac{\lambda}{U} \sum_{k=1}^{U} \frac{\partial \ell_k}{\partial \theta}$$

- However, computing error-derivatives through time is difficult. This is mostly due to the relationship among the parameters and the dynamics of the RNN, that is highly unstable and makes GD ineffective.

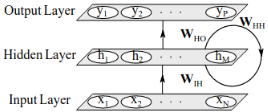
Gradient-based Learning Methods

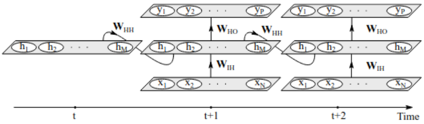
- Gradient-based algorithms have difficuly in capturing dependencies as the duratiton of dependencies increases.
- The derivatives of the loss function with respect to the weights only consider the current state, without using the history information for weights updating.
- RNNs cannot learn long-range temporal dependencies when GD is used for training.

Back-propagation through time (BPTT)

- Backpropagation Through Time (BPTT) is the adaption of the backpropagation algorithm for RNNs.

- BPTT unfolds the RNN to construct a traditional Feedfoward Neural Network where we can apply backpropagation.





Back-propagation through time (BPTT)

Since we have three weight matrices Wxh, Whh and Who we need to compute the partial derivative to each of these weight matrices using the chain rule.

Partial derivative with respect to $\mathbf{W}_{ho}$

$$\frac{\partial L}{\partial \mathbf{W}_{ho}} = \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\partial \mathbf{W}_{ho}} = \sum_{t=1}^{T} \frac{\partial l_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} . \mathbf{H}_t$$

Partial derivative with respect to $\mathbf{W}_{hh}$

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{W}_{hh}} &= \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\partial \mathbf{H}_t} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{hh}} \\ &= \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \cdot \frac{\partial \mathbf{H}_t}{\partial \phi_h} \cdot \frac{\partial \phi_h}{\partial \mathbf{W}_{hh}} \end{aligned}$$

Back-propagation through time (BPTT)

- Since each $\mathbf{H}_t$ depends on the previous time step we can substitute the last part from above equation

Partial derivative with respect to $\mathbf{W}_{hh}$

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} \cdot \frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{hh}}$$

- Similarly, we have the partial derivative with respect to $\mathbf{W}_{xh}$ as below

Partial derivative with respect to $\mathbf{W}_{xh}$

$$\frac{\partial L}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial \mathbf{O}_t} \cdot \frac{\partial \mathbf{O}_t}{\partial \phi_o} \cdot \mathbf{W}_{ho} \sum_{k=1}^{t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} \cdot \frac{\partial \mathbf{H}_k}{\partial \mathbf{W}_{xh}}$$

Back-propagation through time (BPTT)

- In order to transport the error through time from timestep t back to timestep k we can have

$$\frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{H}_i}{\partial \mathbf{H}_{i-1}}$$

- We can consider previous equation as a Jacobian matrix for the hidden state parameter as

$$\prod_{i=k+1}^{t} \frac{\partial \mathbf{H}_i}{\partial \mathbf{H}_{i-1}} = \prod_{i=k+1}^{t} \mathbf{W}_{hh}^{T} diag \mid \phi_h'(\mathbf{H}_{i-1}) \mid$$

Table of Contents

Vanishing Gradient

- This problem refers to the exponential shrinking of gradient magnitudes as they are propagated back through time.
- This phenomena causes memory of the network to ignore long term dependencies and hardly learn the correlation between temporally distant events
- There are two reasons for that:
  1. Standard nonlinear functions have a gradient which is almost everywhere close to zero;
  2. The magnitude of gradient is multiplied over and over by the recurrent matrix as it is back-propagated through time.

Vanishing Gradient

- It is possible to use singular values to generalize it to the non-linear function ϕ'_h by bounding it with $\gamma \in R$ such as

$$|| \; diag(\phi'_h(\mathbf{H}_k)) \; || \leq \gamma$$

- The Jacobian matrix $\frac{\partial \mathbf{H}_{k+1}}{\partial \mathbf{H}_k}$ and the bound in previous equation, we can have

$$\forall k, \left|\left|\frac{\partial \mathbf{H}_{k+1}}{\partial \mathbf{H}_k}\right|\right| \leq ||\mathbf{W}_{hh}^T|| \; || \; diag(\sigma'(\mathbf{H}_k)) \; || < 1$$

Vanishing Gradient

- We can consider $|| \frac{\mathrm{d}\mathbf{H}_{k+1}}{\mathrm{d}\mathbf{H}_k} || \leq \eta < 1$ such as $\eta \in R$ for each step k. By continuing it over different timesteps and adding the loss function component we can have

$$|| \frac{\partial \ell_t}{\partial \mathbf{H}_t} (\prod_{i=k}^{t-1} \frac{\partial \mathbf{H}_{i+1}}{\partial \mathbf{H}_i}) || \leq \eta^{t-k} || \frac{\partial \ell_t}{\partial \mathbf{H}_t} ||$$

- Finally, we can see that the sufficient condition for the gradient vanishing problem to appear is that the largest singular value of the recurrent weights matrix $\mathbf{W}_{hh}$ satisfies $\lambda_1 < \frac{1}{\gamma}$

Exploding Gradient

- Gradients in training RNNs on long sequences may explode as the weights become larger and the norm of the gradient during training largely increases.
- As it is stated before, the necessary condition for this situation to happen is $\gamma > \frac{1}{\gamma}$.

Conclusion for the 2 problems

Vanishing Gradient

$$|| \frac{\partial \mathbf{H}_{i+i}}{\partial \mathbf{H}_i} ||< 1$$

Exploding Gradient

$$|| \frac{\partial \mathbf{H}_{i+1}}{\partial \mathbf{H}_i} ||> 1$$

- To deal with the exploding gradients problem, we propose a solution that involves clipping the norm of the exploded gradients when it is too large.

$$\begin{aligned}
&\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta} \\
&\textbf{if} \ \ \|\hat{\mathbf{g}}\| \geq \mathit{threshold} \ \textbf{then} \\
&\quad \hat{\mathbf{g}} \leftarrow \frac{\mathit{threshold}}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}} \\
&\textbf{end if}
\end{aligned}$$

- To deal with the vanishing gradients problem, we propose using LSTMs layers rather than normal Recurrent layers.