TP: Introduction to RNN

1 BPTT vs. Truncated BPTT

The goal of this TP is to study how to train a simple RNN by minimizing a loss L. Consider the following problem

$$\min_{\theta \in \mathbb{R}} L(\theta) = \sum_{t=1}^{T} L_t(\theta)$$

where

- $h_0 = x_0 \sim \mathcal{N}(\mu, 1)$
- $h_t = h_{t-1} + \theta$
- $L_t(\theta) = \mathbb{E}_{x_0 \sim p}(h_t - \mu)^2$

We shall compute the gradient of L with respect to θ and the truncated gradient in Pytorch, and then implement gradient descent methods : BPTT and Truncated BPTT.

1.1 BPTT

• Compute the loss L, the gradient $\nabla_\theta L$ at $T = 20$ and $\theta = 1$. Use 1024 samples of x_0 to estimate the expectation of each L_t.

• Implement the following gradient-descent method (GD) using pytorch's auto-differential function (backward), in order to minimize L with an initial $\theta = 1$,

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla_\theta L(\theta^{(k)}).$$

Set a proper learning rate for $T = 20$ to achieve a fast convergence.

• We denote the parameter at k-th iteration of GD by $\theta^{(k)}$. Make a plot of the loss $L(\theta^{(k)})$ as a function of k, as well as the gradient $\nabla_\theta L(\theta^{(k)})$. An example is given in Figure 1 with $\mu = 1$.

1.2 Truncated BPTT

• Compute the truncated gradient at each step k. Use 1024 samples of x_0 to estimate the expectation of L_k.

• Implement the truncated BPTT method

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \widetilde{\nabla}_\theta L_{k+1}(\theta^{(k)})$$

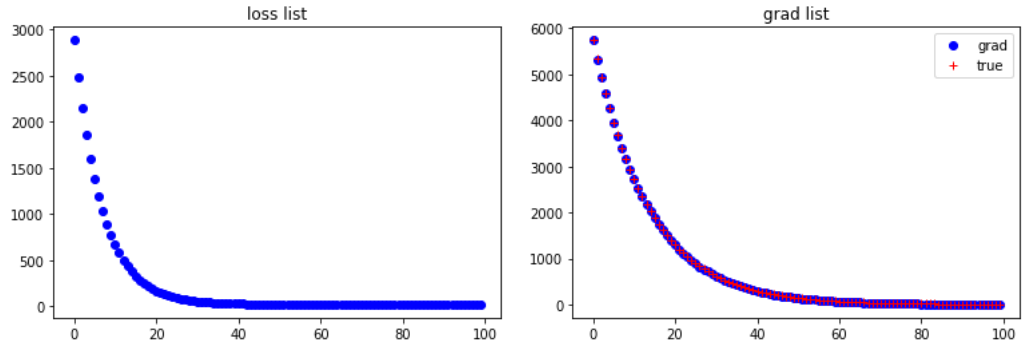

Figure 1 – The loss $L(\theta^{(k)})$ and the gradient $\nabla_\theta L(\theta^{(k)})$ as a function of k in BPTT.

• Start from $\theta = 1$, make a plot of how $L_{k+1}(\theta^{(k)})$ and $\widetilde{\nabla}_\theta L_{k+1}(\theta^{(k)})$ change over 1000 iterations.

• Set $\eta_k = 0.5/k$, make a plot of how $\theta^{(k)}$ changes over 1000 iterations. How far the θ at the last iteration is from being optimal ?

• Set $\eta_k = 0.5/k^{3/2}$, and check whether the algorithm converges faster. Implement this using 'torch.optim.lr_scheduler.LambdaLR'. You may try other learning rates as well. [1]

• Restart the truncated GD method from the last $\theta^{(k)}$, and run for another 1000 iterations. Repeat this twice. Is the θ at the last iteration getting closer to zero ?

1. `https://www.kaggle.com/isbhargav/guide-to-pytorch-learning-rate-scheduling`