Backpropagation Through Time (BPTT) \cite{webBack} consist in a variant of the previous algorithm to train RNNs. Refers to the backpropagation train algorithm when it take into account time to compute the gradient descent from a sequence data. The idea behind this algorithm is similar to the previous presented one:

1. **Initialization**. As in backpropagation case, it is assumed no prior weights information available.

2. Presentation of **Training sequence of time steps** samples which will follow the following operations.

3. Forward. The algorithm goes through all layers but in that case it accumulate errors across each time-step by unrolling the network. The error is derivate at each time step.

4. Backward. It “roll-up” the network and it update from different time step for each weights. Therefore, each weight will require same number of derivations than time-step for single update of itself.

**5. Iterate** until a error criteria is reached. BPTT can be computationally expensive if the number of time-step is high.

Conceptually, this training algorithm is not much complex as backpropagation but due to the high level of computation complexity, with long sequence can cause weights to vanish and make slow (or inefficient) learning.

Truncated Backpropagation Through Time (TBPTB) is a modified version of BPTT this problem. Instead of taking the whole set of sequence, TBPTT process one time step at a time and periodically (usually named as k1) the algorithm updates back for a fixed number of time steps (k2). K2 needs to be large to capture temporal structure to learn, but not large enough to cause vanishing weights.

It is important to mentioned that Tensorflow truncated backpropagation using k1=k2 which cause slightly differences with the original one. It means that when Tensorflow compute the TBPTT uses all the information from the time steps sequence, instead of only using truncated part.

Batch training

Training a Neural Network is the process of applying the algorithm backpropagation to compute the optimum error from a training data set and with that finding a set of weights and bias values that fit in the same way new data and the network can make predictions to them (test data set). There are two main ways of training called mini-batch training and stochastic training. The major difference between them is that the batch algorithm keeps the system weighs constant while computing the error associated with each sample in the input. Since the on-line version is constantly updating its weights, its error calculation uses different weights for each input sample. In other words,

A priori, knowing our resources and our needs, we discard other options than not training with batches.