Neural network knowledge distillation in tensor networks

Dereck Piché

January 25, 2023

Abstract

This is the paper's abstract ...

1 Introduction

This is time for all good men to come to the aid of their party!

Outline

2 Layer-by-layer approach

It has been common to each layer of a neural network as a certain abstracted representation of the previous information and their for generalisibility. Thus, we propose to change the cost, we use a tensor mapping and train each mapping individualy.

$$x \longrightarrow H_1 \longrightarrow H_2 \longrightarrow (\dots) \xrightarrow{g} \hat{y}$$

Each hidden layer is of the form;

$$a^l = \sigma^l\big(W^l a^{l-1}\big)$$

In tensor layer form, it will be defined as

$$a^l = T^l \cdot \Phi\big(a^{l-1}\big)$$

The main difference is that in the tensor approach, the "heavy" part is done by the non-linear transformation while a little work is done with the linear mapping. The opposite is true with neural networks. Here, $\Phi(X)$ (X begin the input vector) is a tensor product of several identical non-linear mappings of each element x_i. Thus, we have

$$\Phi(X) = \phi(x_1)\phi(x_2)\ldots\phi(x_n)$$

Were each $\phi : R \rightarrow R^d$, and each $d > 1$. Thus, our $\Phi(X) : R^n \rightarrow R^{(d\times)^{n-1}d}$. In other words, our Φ returns a tensor of order n, where each indices run from 1 to d.

3 Going further with local transformations

In previous works, the non-linear mappings were independant and equal for each (x_i). It would be interesting if these linear mappings were codependant. Here is an example of transformation to desambiguate what we mean

$$\phi_i(x_i, x_{i+1}) = \begin{bmatrix} x_i \\ x_i \cdot x_{i+1} \end{bmatrix}$$

It would be interesting to apply this transformation for a set of randomised pairs before the training.