Exploring additive and multiplicative similarity measures in multilayer perceptrons.

Santiago Aranguri aranguri@stanford.edu

Introduction

In this project, we are going to analyze the advantages and disadvantages of different ways of measuring the similarity between vectors in an embedding space that represents real information. This is important because having a good similarity measure enables us to solve most of the machine learningproblems, just by transforming the representation of the input to a convenient space and then measuring the similarity between the input and previously defined categories.

PITCH LINK: https://youtu.be/dNn0NvLVg9I

Datasets

Clustered vector dataset. We generated $m = 10000$ vectors drawn uniformly where $v \in [-1, 1]^n$ with $n = 100$. Then, for each vector, we created 2 vectors by adding to it random Gaussian noise with mean 0 and variance 1. Given two different random vectors, the correct output is 1 if they come from the same cluster and 0 if they don't.

bAbi dataset. The bAbI dataset is a set of tasks for natural language processing. Although it includes different types of tasks, in this project we focused only on the two supporting fact task. That task consists of receiving a set of facts and answering a question that required two facts from the list to be answered.

Methods

A similarity measure is a function $sim(q, M)$ that receives a query vector q and memory vectors M and outputs a probability distribution P over M where each $p \in P$ represents the similarity between the q and $m \in M$. In general, $sim(q, M)$ has two steps. First, we compute the similarity $f(q, m)$ for all $m \in M$. Secondly, we normalize the probabilities by using a softmax.

Additive interactions (A.I.) We define an additive similarity function as $f(q, m) = ||q - m||_1$.

Multiplicative interactions (M.I.) We use the element-wise product for the multiplicative similarity, which is defined as $f(q, m) = q \odot m$.

Parametric interactions An example of using trained parameters is $f(q, m) = MLP(q, m; \theta)$ where MLP is a multilayer perceptron with only one unit for the output and θ as the weights.

Experiment - Memory Networks

We used the bAbi dataset and the Dynamic Memory Network (DMN), a model that performs multiple passes through the input, computing a memory vector that is taken as input for the next pass. For each step that we compute the memory state, we need to know what facts to consider, so we have gates. We compute the values for the gates by passing z through a two layer MLP. z is defined as follows.

$$z = [c, m, q, |c - q|, |c - m|, c \odot q, c \odot m, c^T W q, c^T W m]$$

where c, m, and q are representations of the fact, memory state, and question respectively. The first experiment was using z as above. After around 200k training examples, the model reached a 99% accuracy. In the second experiment, we removed the four last entries in the nine entries Z has. Thus we only had A.I.. Here it took 300k training examples to reach 99%. Finally, we removed the fourth, fifth, eighth, and ninth entries, so as to have only element-wise M.I.. Surprisingly, the model never got a better accuracy than 13%, after going through >1M training examples.

Experiment - Multiplicative interaction

We used the dataset and task from "clustered vectors dataset." We tested whether M.I. were useful for this task by using two MLPs. MLP_1 receives the two random vectors as input and outputed its similarity. MLP_2 receives as input the two random vectors and their element-wise product, and it has one output unit. Both MLP had two layers and 300 hidden units with ReLU as activation function. MLP_2 reached an accuracy of 95% while MLP_1 couldn't capture underlying information from the input, getting an accuracy of around 50%. We thought that the higher the variance, the more difficult it is to distinguish between two vectors in the same cluster. Using a variance of .5, MLP_2 got around 90% and MLP_1 around 60% (see figure 1.) However, using a variance of 2 neither of the two $MLPs$ reached a high accuracy. While MLP_1 barely distinguished between the random vectors, MLP_2 achieved an accuracy of around 65%. See figure 2.

Figures

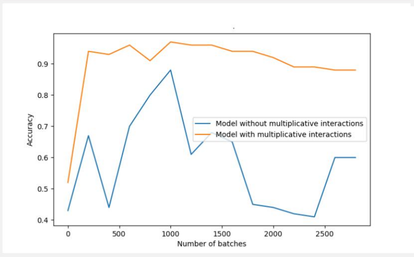

Figure 1: Random noise with variance 0.5.

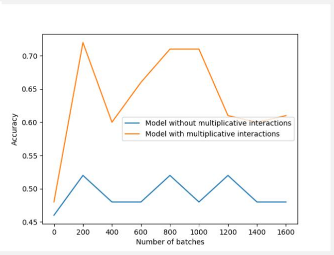

Figure 2: Random noise with variance 2.

Conclusion, discussion, future

In the M.I. experiment we saw that the model that receives the elementwise product always outperforms the model that doesn't receive it. But, in the DMN, the efficacy of the M.I. vs A.I. was the reverse. Both experiments suggest that depending on the problem one is trying to solve, using different interactions between the input data can prove useful. An exalanation for this is that the DMN is more complex and thus is able to model M.I. without receiving it as input. Future work could help in understanding why the model that didn't receive A.I. in the DMN performed so poorly and analyzing whether more complex model don't require M.I. as input. (As the DMN seems to do.)