CSE 250B - Project 4

Lund, Sigurd Stoeen
sigurlu@stud.ntnu.no

Wolff, Thomas
thomawo@stud.ntnu.no

Introduction

This paper describes a way to learn the meaning of sentences using artificial neural networks (ANN) with recursive auto encoders (RAE). The goal is to predict if a movie review is either positive or negative.

Stochastic gradient descent (SGD) is used to learn the model. We use an algorithm called backpropagation (backprop) to find the partial derivatives needed to perform learning. For testing correctness of the partial derivatives, we also calculate them by using numerical differentiation.

Our implementation of backprop did not get the same derivatives as the numerical differentiation, and hence the weights were not be learned correctly during training. Therefore the model learned was only able to predict 50% of the reviews correctly.

Design and analysis of algorithms

In this section we describe the problem more formally and the algorithm used for solving it.

Defining the problem

Given an English sentence, we want to be able to predict its meaning. Here, ”meaning” refers to some vector representation where each element is a real number. From a sentence’s meaning vector it is possible to calculate a label for the sentence, which could be a binary or a multiclass classifier. E.g in the binary case the label could be ”positive” or ”negative”.

Representation

We represent a sentence with a binary tree, where each leaf node corresponds to a word, and every internal node of the three corresponds to a phrase in the sentence. The root node corresponds to the whole sentence. We assign to each node n in the tree a meaning vector $x_n \in \mathbb{R}^d$. Every node’s meaning vector represents the meaning of that node’s corresponding phrase, and

has dimension d. Typical values for d is 20 or 100. The meaning vector for leaf nodes can be known in advance, or it could be initialized to some arbitrary vector. The meaning vector of an internal node is a function of its two children's meaning vectors. More specifically, internal node k with children i and j has meaning vector x_k:

$$x_k = h(W[x_i; x_j] + b)$$

where x_i and x_j are the meaning vectors of node k's children i and j respectively, W is a matrix and b is a vector who's values are to be learned, and h is a point-wise sigmoid shaped function.

Doing supervised learning

To be able to do supervised learning on these sentences, there has to be some target meaning for each sentence. We associate with each sentence a target vector t, where t is the true meaning of the sentence. A simple loss function to use during training is square loss:

$$L_{square} = ||x_{root} - t||^2$$

If we can compute $\frac{\partial L_{square}}{\partial w_{ij}}$ for each w_{ij} in the matrix W and $\frac{\partial L}{\partial b_i}$ for each b_i in the vector b we can use any gradient following algorithm for minimizing L.

Recursive autoencoders

The above method for learning the meaning of sentences assumes that the structure of a sentence's tree is known. In this section we introduce an algorithm for finding a tree structure based purely on the meaning vector of the individual words using recursive autoencoders.

The goal of a recursive autoencoder is to reconstruct its inputs. In the case of the binary tree introduced in the above section, given the meaning vector x_k for internal node k, we want to be able to reconstruct its children's meaning vectors x_i and x_j. The approximated reconstruction of x_i and x_j is z_i and z_j respectively, and is found by using:

$$[z_i; z_j] = Ux_k + c$$

where $U \in \mathbb{R}^{2d \times d}$ and $c \in \mathbb{R}^d$ who's values are to be learned during training.

We can define a loss of this approximated reconstruction at node k as

$$L_{recon}(k) = ||z_i - x_i||^2 + ||z_j - x_j||^2$$

Furthermore, we can define an optimal tree structure as the tree that minimizes this loss function over all internal nodes in the tree. There are an exponential number of possible binary trees given the number of leaf nodes, and one could enumerate all these and choose the tree that has the lowest value for the above loss function. Since this is not a tractable way of finding an optimal tree, rather than finding the best tree we can find a good tree by using a greedy

algorithm. The greedy algorithm works as follows: given a sentence of n words, consider all $n-1$ pairs of consecutive words and their reconstruction loss. Choose the pair that has the lowest reconstruction loss and make those the children of the next internal node in the tree. Now, do the same again with the $n-2$ remaining pairs of nodes where the new internal node from the previous iteration has replaced its two children in the list of remaining nodes. Continue in this fashion until there is only one remaining node, namely the root.

We will make a refinement to the definition of reconstruction loss above, based on the intuition that longer phrases should contribute more to the reconstruction loss than shorter phrases. More specifically, if an internal node k has children i and j, the reconstruction loss at node k is:

$$L_{recon}(k) = \frac{n_i}{n_i + n_j}||z_i - x_i||^2 + \frac{n_j}{n_i + n_j}||z_j - x_j||^2$$

where z_i and z_j are the approximate reconstructions of x_i and x_j respectively, and n_i is the number of words in the phrase of node i and n_j is the number of words in the phrase of node j.

Using meaning vectors to predict labels

In the section "defining the problem" we introduced a method for learning the weights in W and b assuming that we had some target meaning t for the given sentence. Suppose we, rather than having a meaning for the sentence, had a target label. This label could be of the types mentioned above, binary or multiclass. Suppose we had a binary valued target label $t \in \{0, 1\}^r$ for the given sentence. In order to compare the meaning vector x_{root} with this target label, we have to transform x_{root} to an output p that can be compared with t. One way of doing that is using:

$$p = softmax(Vx_{root})$$

Element p_i of p would then be the probability that the sentence belongs to class i, and $V \in \mathbb{R}^{r \times d}$ which is another matrix whos values are to be learned during training.

There are several possible loss functions that can be defined over t and p. Among them are square loss

$$L_{square}(p, t) = \sum_{i=1}^{d} (p_i - t_i)^2$$

and log loss

$$L_{log}(p, t) = -\sum_{i=1}^{d} t_i \log p_i$$

Now that we have expressions for the loss due to reconstruction when finding a tree structure, and the loss due to difference in predicted label and the true label, we can define a total loss for the current model for a given sentence s with true label t:

$$J = \sum_{k \in T(s)} \alpha L_{recon}(k) + (1 - \alpha) L_{square}(k)$$

where T(s) is the set of internal nodes in the tree, and alpha is a hyperparameter that controls the importance of the reconstruction loss compared to the loss at output nodes.

Notice that there is, in addition to the two approximated reconstruction nodes z_i and z_j at internal node k, defined an output node p for k as well, and that the total loss is the sum of loss at all these output nodes.

The Backprop algorithm

Recall that the meaning vector x_k for internal node k was calculated using:

$$x_k = h(W[x_i; x_j] + b)$$

We can incorporate b in the matrix multiply by rewriting the expression to:

$$x_k = h([W; b][x_i; x_j; 1])$$

We call $[W; b]$ an extended weight matrix. The same thing can be done when calculating the reconstructions z_i and z_j by writing:

$$[z_i; z_j] = [U; c][x_k; 1]$$

To be able to minimize the loss function J for a sentence, we have to calculate the partial derivative

$$\frac{\partial J}{\partial W_{ij}}$$

for every element W_{ij} in every extended weight matrix used in the network. This section describes and algorithm that computes the needed partial derivatives in linear time in terms of the number of weights.

Given that node j's value is $h(a) = h(W[q^1; \ldots; q^m; 1])$, where there are m nodes feeding into node j, weight W_{ij} affects J only through a's i'th element a_i, i.e

$$\frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial a_i} \frac{\partial a_i}{\partial W_{ij}}$$

We define

$$\delta^i = \frac{\partial J}{\partial a_i}$$

and each node has a δ vector.

$\frac{\partial a_i}{\partial W_{i:}}$ can easily be seen to be

$$\frac{\partial a_i}{\partial W_{i:}} = [q^1 : \ldots; q^m; 1]^T$$

and we only need to derive an expression for each node's δ vector to be able to describe the Backprop algorithm.

For each output node with output vector o we assume that it's contribution to the total loss function J is a sum of the contributions from each of it's elements, where the contribution from element o_i is L_i. Then we have

$$\delta^i = \frac{\partial J}{\partial a_i} = \frac{\partial L_i}{\partial a_i} = \frac{\partial L_i}{\partial r_i}\frac{\partial r_i}{\partial a_i} = \frac{\partial L_i}{\partial r_i}h'(a_i)$$

For output nodes, the partial derivative $\frac{\partial L_i}{\partial r_i}$ can be computed directly, given it's loss function, and the same goes for $h'(a_i)$.

For all other nodes, it's δ vector can be calculated using:

$$\delta = h'(a) \circ [\sum_k (\delta^k)^T V^k]^T$$

where $\circ$ means pointwise multiplication.

Now that we have an expression for every node's δ vector we can describe the Backprop algorithm for computing the partial derivatives $\frac{\partial J}{\partial W_{ij}}$:

Step 0: Compute vector value of each node in the tree

Step 1: Compute δ vector for each output node

Step 2: Compute δ vector for each non-output node

Step 3: Compute for all nodes $\frac{\partial J}{\partial W_{ij}} = \delta[q^1; \ldots; ^m; 1]$

Numerical differentiation

Numerical differentiation is a second way to compute the derivatives, and will be used to verify that the values from backprop is correct. We use the central differences which is defined as:

$$\frac{\partial J}{\partial w_{ij}} = \frac{J(w_i j + \epsilon) - J(w_i j - \epsilon)}{2\epsilon} + O(\epsilon^2)$$

For a weight m. a small value ϵ is added and the total loss is calculated, then the total loss is calculated for the same weight m, but now subtracted by ϵ and the difference is seen by dividing by 2ϵ. This is done for every weight in each weight matrices. The lower ϵ is, the more accurate the derivatives will be. However when computing you need to keep in mind that the value will be rounded to 0 if it is too low.

The running time for each weight is $O(m)$ time, so computing the derivatives for every weight requires $O(m^2)$ time. This is not feasible in practise, so it will only be used on a small dataset to verify the derivatives.

Experiments

Design of Experiments

First experiment is to run the training algorithm with both backprop and numerical differentiation, and see if the partial derivatives matches for both methods. This will not guarantee that any of the methods calculating the derivatives are implemented correctly, but it is very unlikely that two different methods will produce the same wrong result.

The derivatives did not match in our case, but it is still a hope that the SGD method could learn well enough, if at least the derivatives has the correct sign. Therefore we will still run tests on the dataset to predict and see what the error rate is. The dataset consists of 2000 movie reviews, and each review consists of several sentences. Each review has a true label of either positive or negative and the goal is to predict that label. Our model will predict sentence by sentence, so the predicted label for a review is the dominating label of the sentences in that review.

We show the ten most positive and negative phrases in the validation set with our learned model. In addition we retrieve a random phrase and find a similar phrase in the validation set, based on their meaning vectors. This similarity will be calculated using Euclidean distance. We will also pick a phrase and show the tree structure that is learned.

Results of Experiments

The partial derivatives calculated by the backprop algorithm and numerical differentiation method did not match. That means that there is a high possibility that the derivatives calculated by backprop is wrong. However the derivatives did match with a precision of three and up to five digits on sentences of two words. With a sentence of three, the derivatives started to be slightly wrong, but the sign matched in all cases. With long sentences, the derivatives did not match on either sign or value, and that made the SGD unable to learn a model.

The model was only able to predict an average of 50% correct labels, which means that it did not learn anything at all, since the data was binary with equal size of positive and negative examples. In Table 1, the 10 most positive phrases are shown. These phrases does not seem to have a trend of being all positive or all negative, rather they seem somewhat randomly chosen. The same goes for the 10 most negative phrases in Table 2.

The two phrases that have a similar meaning vector, does not seem to have very similar meaning seen from a human perspective, Table 3.

Most positive phrases
miguel ferrer are invading china
the fairy godmother standin leonardo da vinci patrick godfrey is fun to watch as the eccentric old man who advises
he's referred to as the mariner but they the meaner would have been a more appropriate title because he's cold rigid
it features a moses who in the interests of making the character more human lacks any divinity whatsoever which isn't convincing at all to anyone
this time he falls for charms of a wily paramedic gloria sullivan kim dickens who may be involved
even more so when considering the fact it's all stuffed rather nonchalantly
anyone can read yet they have skidoos and airplanes
lost his ear but they are so flat and so plastic that they come off as prefabricated and unbelievable
this is only one of the minor chores involved
the thin red line will doubtless be compared to 1998's other

Table 1: 10 most positive phrases

Most negative phrases
most of all it contains an attempt to commercialize homogenize and massmarket a story about a manifestation
marley robert forster his new second officer nick vanzant james spader head medical officer kaela evers angela bassett medical technicians yerzy penalosa lou
this is the ultimate game of six degrees of separation
immediately the film's biggest flaw is apparent are these people primitive
primitive people that somehow have some of the technology we have today but also some of the lowtech tools used by pirates and vikings
since i didn't understand anything about these characters' motivations did i care when troy abruptly starts murdering them onebyone and he does knocking off three
the phantom allegedly based on the longrunning comic
but in something of a pseudo groundhog day' approach director tom tykwer gives lola three
on my second viewing i realized that the film follows the three
furthermore the end of the film looks like they took what were originally intended to be the opening credits and spliced them in right before the real end credits

Table 2: 10 most negative sentences

Two similar phrases
director griffin dunne is known mainly as an actor in such films as an american werewolf in london and after hours
so a team of scientists and filmmakers travel to the amazon to search for a legendary indian tribe

Table 3: Two similar phrases

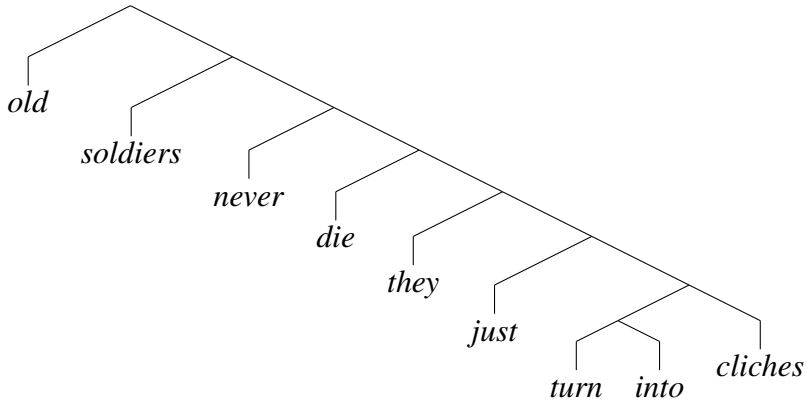

Figure 1: The tree structure shown in this figure is an example of a learned three for a sentence.

Findings and lessons learned

The results presented confirms that it is extremely important to calculate the partial derivatives correctly to be able to learn a good model of the data. An implementation with wrong derivatives will just move the weights in random directions, and therefore not be able to use the weights to predict new data.

References

1. http://cseweb.ucsd.edu/ elkan/250B/learningmeaning.pdf

2. *Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions* by Richard Socher *et al*

3. http://www.cs.cornell.edu/people/pabo/movie-review-data/