Machine Learning Engineer Nanodegree

Capstone Proposal

Naren Doraiswamy
March 4th, 2017

Domain background

Ever wondered how google always almost predicts the question you ask in its search engine? Fascinating , is'nt it? The same question boggled me from a long time and after spending quiet a lot of on the internet researching and reading through awful lot of materials and then finally taking the machine learning nanodegree course, i could learn the finer nuances about machine learning and also deep learning. My capstone project will be based on a machine translation model using recurrent neural networks which can be applied to speech recognition The applications of language models are two-fold: First, it allows us to score arbitrary sentences based on how likely they are to occur in the real world. This gives us a measure of grammatical and semantic correctness. Such models are typically used as part of Machine Translation systems. Secondly, a language model allows us to generate new text (I think that's the much cooler application). Training a language model on Shakespeare allows us to generate Shakespeare-like text.

The goal of statistical language modeling is to predict the next word in textual data given context, thus we are dealing with sequential data prediction problem when constructing language models. Still, many attempts to obtain such statistical models involve approaches that are very specific for language domain - for example, assumption that natural language sentences can be described by parse trees, or that we need to consider morphology of words, syntax and semantics. This project is partly inspired by andrew karpathy's post (http://karpathy.github.io/2015/05/21/rnn-effectiveness/) and research paperhttp://www.fit.vutbr.cz/research/groups/speech/publi/2011/mikolov_icassp2011_5528.pdf

Problem statement

The basic aim here is to predict the word that will be inputed next by training the model on some training data. Sequential data prediction is considered by many as a key problem in machine learning and artificial intelligence. Developing a machine translation model that predicts the next word that will be typed is one of the intelligence problem that can be solved and it takes the next step towards developing smarter systems Even the most widely used and general models, based on n-gram statistics, assume that language consists of sequences of atomic symbols - words - that form sentences, and where the end of sentence symbol plays important and very special role.

Datasets and inputs

The dataset that will be used here for this project will be the 15,000 reddit comments downloaded from google's bigquery(now using another smaller dataset taken from a fictional novel as the reddit data was too huge to train with a trial account on floydhub due to time limit contraints). The input data is fragmented into sentences and then words and then into each characters and a vocabulary list is created. A word to vec dictionary is created and then used as inputs for the further computations. Different datasets can be used for training the model for different purposes. Generation of theme based scripts has become one of the most fascinating and advanced sequential data prediction problems that has been solved by intelligence.

Solution statement

I aim to create a generative language based model by using recurrent neural networks which makes us of the sequential information and predicts text. The traditional neural networks will not work in this case as they assume independence between the inputs to the network but language based models depend on the sequential information and hence recurrent neural networks work the best for us. In general, for image and video based processing of information, the convolutional networks seem to work the best for their location independence quality whereas for natural language processing, the recurrent neural networks (tweak: LSTM and GRU) work the best though conv-nets do seem to work efficiently on NLP tasks with some complex optimizations and minor tweaks.

Benchmark model

The initial benchmark model will be based on a reference paper for this project (https://github.com/udacity/deep-learning/blob/master/intro-to-rnns/Anna_KaRNNa.ipynb) from which my processes are taken and implemented in Keras abstract framework. I will be implementing the backpropagation through time technique along with stochastic gradient descent and using adam optimizer for optimization processes.

Evaluation metrics

To evaluate the performance of my machine translation model, i will use the cross-entropy loss method to calculate the deviation from the actual word to the predicted word and then try to minimize the loss by using the stochastic gradient descent method and back propagation. I will be implementing a basic SGD with adam optimization.

Project design

The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you will wanna know which words come before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later).
I will be using the dataset from google's big-query which consists of about 15,000 reddit comments (now using smaller dataset due to time constarints on trial floydhub account and huge time taken to tr ain the network just for a single epoch) and after training our model, hopefully we will be able to gen erate comments similar to fiction novel threads. Like any other project ,i would first like to pre-proces s the data which includes sentence and word tokenization of the data and then taking the most com mon words for training our model and assigning them to a list of vocabulary. Each word is given an i ndex and kept a tab on. Then the recurrent neural network is built and the model is trained. The loss incurred during training will be measured by the cross-entropy function and the softmax activation function will be used for the model.

Let's get concrete and see what the RNN for our language model looks like. The input x will be a sequence of words (just like the example printed above) and each x_t is a single word. But there's one more thing: Because of how matrix multiplication works we can't simply use a word index (like 36) as an input. Instead, we represent each word as a one-hot vector of size vocabulary size. For example, the word with index 36 would be the vector of all 0's and a 1 at position 36. So, each x_t

will become a vector, and x will be a matrix, with each row representing a word. We'll perform this transformation in our Neural Network code instead of doing it in the pre-processing. The output of our network o has a similar format. Each o_t is a vector of vocabulary size elements, and each element represents the probability of that word being the next word in the sentence.

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$o_t = \{softmax\}(Vs_t)$$

where x_t will be the inputs , s_t will be the memory from previous states o_t will be the outputs and the variables U,V,W will be the learning parameters which will be applied to the model. This builds the forward propagation part of the model. Then the backpropagation through time (BPTT) is run to learn the suitable learning parameters and is aided by the stochastic gradient descent algorithm for the learning of weights along the model.

Now after training the model on the dataset, it's time to generate the new text for which we will use a helper function which keeps a tab on the starting and the ending sentence token which is appended at the start and the end of a sentence..

Since the data that is being used for training the model is a bit huge,