Advance$_M L_H W02$

hitesh.khandelwal$_u g21$

April 2021

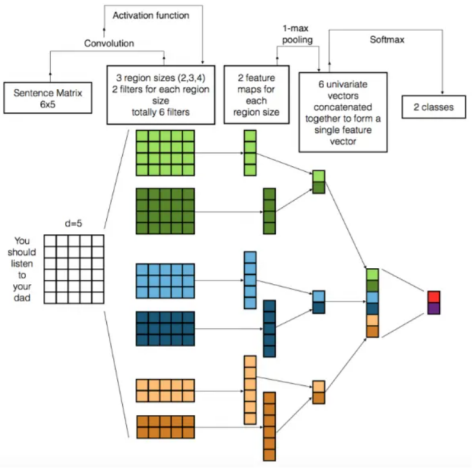
1 Question 1

A :
CNN or any neural nets works with numbers, therefore there is a need to transform our input data from text format to numbers. In CNN specifically we need a matrix which our filters can later operate on. To transform the texts into numbers we use word embedding like word2vec or glove. Here, Each words are transformed into a vector which is obtained using its word embedding. We obtain embedding for each word in our input data. This array of vectors(word embedding) forms the matrix that serves as the input for our CNN

B :
Let's understand how filters that we use in the case of computer vision in CNN can be used here as well. For this consider the following example shown in the diagram.
In case of computer vision we slide over local regions of the images using our filters but in NLP we usually slide over the entire matrix. Therefore, the width of the filter is normally same as the width of our input matrix but the height can vary. In the above example, The first layer has six filters, where 2 filters has height 2, 2 filters has height 3 and last 2 filters has height 4 and all filters has same width as that of input matrix. Here height indicates the number of words that filter is operating at a time. These filters are then convolved accordingly to get output matrix with the size : input_size - filter_size + 1 (this is without padding and stride).

C :
Since unlike RNN'S, in the CNN we are operating on entire data (matrix) at once, the processing/computation time is significantly better than those of RNN'S but at the cost of accuracy.

2 Question 2



Significance of MCMC in ML : MCMC provides a class of algorithms that helps in approximating some parameters in a probabilistic distribution. This is done by repeatedly drawing a sample where the next sample drawn is dependent on the current sample, thus forming a Markov chain. This in the end narrows down the quantity that one is trying to approximate from the distribution.

Depending on how one constructs a markov chan, we get different MCMC algorithms. Various algorithms using MCMC for samling are gibbs sampling, metroplois sampling etc.

Explanation of algorithm : In Metropolis-hastings algorithm we do the following :

- let P(x) be the probability distribution that we are trying to approximate
- First we choose a random point x_t and choose a random probability density g(x/y) that gives or suggests the next possiblt candidate x given the previous sample value y. one important condition here is that g(x/y) needs to be symmetric. More often then not, g(x/y) corresponds to a normal distribution centered around y.
- Now for each iteration till t (convergence):
  we generate the next candidate x from the probabaility density g(x/x_t). Then the acceptance ratio f(x)/f(x_t) is calculated and depending on the threshold if the ratio is greater than the threshold, the sample is accepted and generated else we move on with the same previous sample and try and see if new sample can be generated from here.

3 Question 3

A :
Pros :

- Can work with input of variable length
- weights are shared across the time
- historical information is taken into consideration during computation.

Cons :

- This approach runs into the problem of vanishing gradients or exploding gradient because during backpropagation the gradient is multiplied along the long chain of previous time steps
- Computationally inefficient because it has to remember all the previous words irrespective of how relevant or important they are to the sentence
- Rnn's tend to overfit and applying regularisation techniques like dropout is not easy with the given approach in the paper
- Therefore, because of the above limitation, RNN works well with small data as working with large data is both computationally inefficient and runs into problems of vanishing or exploding gradient

B :
Pros :

- Since the paper uses LSTM approach, working with large data becomes easy as LSTM uses memory cells to retrieve data that is important or relevant.
- Vanishing gradient or exploding gradient problem is solved with this approach to some extent because using LSTM approach we skip many time steps during backpropogation to compute gradients
- Tradition lstm cannot regularize dropout to prevent overfitting but in the paper the discussed method makes room for some kind of regularisation using dropout that helps in addressing the problem of overfitting.

Cons :

- Since the approach uses LSTM, it is computionally very heavy as it needs more memory.
- Though the approach in the paper has addresed the problem of vanishing gradient by using LSTM, The problem may still persist if the sentence is such that most of its wods needs to be remembered in which case during backpropogation you rarely skip time steps to compute gradient.
- Just like vanishing gradient problem, thought the problem of overfitting is addressed it is not completely eliminated