Project proposal

We want to look more into predicting NN, tree based models and K-nearest neighbors with in context learning. We also want to see how the performance changes if we have multiple of these models in the training data, compared to just one. To start, we will just use the same models as in the paper “What can transformers learn in context” (Garg et.al.), and then increase complexity with the models and have multiple models in the training data, to see the impact on performance.

The actual project proposal is on page 3.

Do you see the chat?

You mean in the text group?

No in google docs. It is right next to the icon with a C at the top.

Oh yeah

Nice!

What you up to now? We could split some work in writing the proposal

Ok, I could come up with a draft version of proposal by 8:30, then we can discuss it, to save you some time, yep.

That would be very nice of you, thank you a lot!!!

Ok, then i will come back as soon as I am done, thank you again!

Yeah, that would be good. I have some other work i need to do before I go to bed, but i can prioritise this. But i think it is better to discuss in the chat.

Project proposal

Do you agree with the following points?

-Looking more into predicting NN and tree based models with in context learning (and other interesting models/functions). Here especially seeing at what complexity of the models we start getting bad predictions, even with a lot of context. Furthermore, it might be interesting to see how the model performs if it is given inputs which are far away from context, as it behaves differently than linreg. Might also see if this changes if you include linear regression in the training data, and you for example have a decision tree that is similar to linreg on an interval. Could also see how it is affected when we change distribution of inputs.

-do we also want to look at k-nearest neighboors?Might be interesting, as he suggested it, and no other group has looked into it as far as i know.

So could have a function class consisting NN, CNN, decision trees, random forest and k-nn.

Could also look more into fine tuning vs. in context learning when it comes to social science problems or sentiment analysis.

Try to get the predictions as good as possible

- Can start with the code base in the “What can transformers learn in context”. And build on that.

-Test/train data will mostly be generated from randomly initializing. (We might need to do something more advanced here though, as we might need to ensure that the resulting models are interesting, especially if we use ensemble models). With ensemble models, we could scale the random outputs by sqrt(d) where d is the number of models, as we then would have variance 1, which is most interesting. For CNN and other types of neural nets

-Use curriculum training. Start with decision trees, increase depth, and number of trees to increase complexity.

-For neural nets we can increase depth at latent dimensions.

-Could do final test with actual trained random forests and neural nets to see if examples are more relevant, or if performance is different in this regard. Might be interesting if performance is similar or better than random forest or decision trees of the same size as the model.

Project Proposal--In-Context Learning

1. Overview

As is illustrated in the paper “What Can Transformers Learn In-Context? A Case Study of Simple Function Classes”, transformer architecture has the ability to in-context learn specific function classes given enough training data and relevant prompt information. We noticed that the paper explored the performance of such model on data generated by tree based models and simple neural nets and would like to elaborate on the following directions:

1) The upper bound of the complexity of patterns that the transformer model can learn. To be more specific, we want to train the same transformer model with data generated from base models of different complexity, and see when the transformer model starts to yield bad predictions. The models we want to consider when exploring this direction are decision trees, random forests, neural nets, and k-nearest neighbors, and add complexity by changing the parameters of these. For example, in the case of a Random Forest, we could add complexity to the base model by increasing the depth and number of trees.

2) The original paper trained the transformer model with data generated from the same function class, and we would like to explore whether the model could learn from data generated from multiple function classes (linear functions/polynomial functions/decision tree/k-nearest neighbors), or even better learn something more than the data represents, and test its performance under different distributions of data.

2. Code Base

We want to use the same model as the paper, which is a decoder-only Transformer architecture(implemented by HuggingFace [Wolf et al., 2020]) with 12 layers, 8 attention heads, and a 256-dimensional embedding space. By appending 2 linear transformations, the model can perform next-token prediction and the dimension of the output corresponds to that of the prompt input.

3. Data Generation

We intend to use similar data generation methods as the paper, which means every training prompt is generated by first sampling a random function f from the function class we are training on, then sampling inputs xi from the isotropic Gaussian distribution N(0, Id). We might want to shift the distribution on testing sets to study whether the in-context learning ability extrapolates beyond training distribution in the case of tree structured models. When creating the random forest, we will scale the outputs of each individual tree, by , where n is the number of trees, to keep the output normally distributed with variance 1. We will also use a similar scaling for k-nearest neighbors, in this case scaling by .

4. Training Process

Apply curriculum training to speed up the training process. We might want to see without curriculum training, how the pattern of the loss curves change as the problem dimension increases. (this was briefly mentioned in the paper)

5. Compute Power

We have a Google colab pro account with an A100 GPU. If necessary we can also get 300 dollars in credit per person by signing up for google cloud compute.