Classical Machine Learning

Week 0

Plan

- Setting up your learning and programming environment

Getting started

- Setting up your ML environment (Setup_NYU.ipynb)
    - Choosing an ML environment (Choosing_an_ML_Environment_NYU.ipynb)
- Quick intro to the tools (Getting_Started.ipynb)

Week 1

Plan

We give a brief introduction to the course.

We then present the key concepts that form the basis for this course

- For some: this will be review
- For others: it will be a preview

Intro to Advanced Course

- Introduction to Advanced Course (Intro_Advanced.ipynb)
- Review and Preview (Review_Advanced.ipynb)

You may want to run your code on Google Colab in order to take advantage of powerful GPU's.

Here are some useful tips:

Google Colab tricks (Colab_practical.ipynb)

Review/Preview of concepts from Intro Course

Transfer Learning: Review (Review_TransferLearning.ipynb)

Transformers: Review (Review_Transformer.ipynb)

Suggested reading

- Attention
  - Attention is all you need (https://arxiv.org/pdf/1706.03762.pdf)
- Transfer Learning
  - Sebastian Ruder: Transfer Learning (https://ruder.io/transfer-learning/)

Further reading

- Attention
  - Neural Machine Translation by Jointly Learning To Align and Translate (https://arxiv.org/pdf/1409.0473.pdf)
  - Geron Chapter 16
  - An Analysis of BERT's Attention (https://arxiv.org/pdf/1906.04341.pdf)

Week 2: Review (continued)

Preview

There is lots of interest in Large Language Models (e.g., ChatGPT). These are based on an architecture called the Transformer. We will introduce the Transformer and demonstrate some amazing results achieved by using Transformers to create Large Language Models.

Attention is a mechanism that is a core part of the Transformer. We will begin by first introducing Attention.

We will then take a detour and study the Functional model architecture of Keras. Unlike the Sequential model, which is an ordered sequence of Layers, the organization of blocks in a Functional model is more general. The Advanced architectures (e.g., the Transformer) are built using the Functional model.

Once we understand the technical prerequisites, we will examine the code for the Transformer.

Plan

We continue the review/preview of key concepts that we started last week.

- We will *re-start* the module on Attention

Our ultimate goal is to introduce the Transformer (which uses Attention heavily) in theory, and demonstrate its use in Large Language Models.

Review continued

Transformers: Review (Review_Transformer.ipynb)

Natural Language Processing: Review (Review_NLP.ipynb)

Language Models, the future (present ?) of NLP: Review (Review_LLM.ipynb)

Week 3: Technical

Plan (part 1)

We finish up the Language Model module with some surprising abilities that seem to "emerge" when LLM's become very large.

[In context learning (Review_LLM.ipynb#In-context-Learning)](Review_LLM.ipynb#In-context-Learning)

Beyond Transfer Learning: Fine-tuning a pre-trained model

Plan

We introduce "Modern Transfer Learning": using model hubs.

The hub we will use for the final project: HuggingFace

- illustrate how to fine-tune a pre-trained model
- quick Intro to HF
    - best way to learn: through the course !
    - uses Datasets
        - will introduce later
    - PyTorch version (uses Trainer); we will focus on Tensorflow/Keras version

HuggingFace Transformers course

The best way to understand and use modern Transfer Learning is via the HuggingFace course (https://huggingface.co/course).

You will learn

- about the Transformer
- how to use HuggingFace's tools for NLP (e.g., Tokenizers)
- how to perform common NLP tasks
    - especially with Transformers
- how to fine-tune a pre-trained model
- how to use the HuggingFace dataset API

All of this will be invaluable for the Course Project.

- does not have to be done using HuggingFace
- but using at least parts of it will make your task easier

- HuggingFace intro (Transfer_Learning_HF.ipynb)
    - linked notebook: Using a pretrained Sequence Classifier (HF_quick_intro_to_models.ipynb)

Suggested reading

HuggingFace course (https://huggingface.co/course)

Functional Models

Plan (part 2)

Enough theory (for the moment) !

The Transformer (whose theory we have presented) is built from plain Keras.

Our goal is to dig into the **code** for the Transformer so that you too will learn how to build advanced models.

Before we can do this, we must

- go beyond the Sequential model of Keras: introduction to the Functional model
- understand more "advanced" features of Keras: customomizing layers, training loops, loss functions
- The Datasets API

Basics

We start with the basics of Functional models, and will give a coding example of such a model in Finance.

- Functional API (Functional_Models.ipynb)

Functional Model Code: A Functional model in Finance: ”Factor model”

We illustrate the basic features of Functional models with an example

- does not use the additional techniques of the next section (Advanced Keras)

Autoencoders for Conditional Risk Factors (Autoencoder_for_conditional_risk_factors.ipynb)

- code (https://github.com/stefan-jansen/machine-learning-for-trading/blob/main/20_autoencoders_for_conditional_risk_factors/06_conditional_a

Week 4

Plan

We continue our exploration of the Functional API in Keras.

We will spend some time examining the code for the Transformer.

We will also introduce the TensorFlow Dataset (TFDS) API, a way to consume large datasets using a limited amount of memory.

THIS WILL BE A VERY CODE-INTENSIVE WEEK

Functional Model Code: A Functional model in Finance: ”Factor model”

We finish up from last week by looking at the actual code

Autoencoders for Conditional Risk Factors (Autoencoder_for_conditional_risk_factors.ipynb)

- code (https://github.com/stefan-jansen/machine-learning-for-trading/blob/main/20_autoencoders_for_conditional_risk_factors/06_conditional_a

Datasets: Big data in small memory

Plan

Last piece of technical info to enable the project

- TensorFlow Dataset (TF_Data_API.ipynb)

Background

- Python generators (Generators.ipynb)

Notebooks

- Dataset API: play around (TFDatasets_play_v1.ipynb)

Advanced Keras

Keras provides many features that make it easier to write complex models

- Custom layer types
- Custom training loops
- Custom Loss functions

We will illustrate these techniques with a coding example.

- Multiple models combined: Transformer (Keras_Advanced.ipynb#Functional-model:-the-basics,-illustrated-by-the-Transformer)
- Custom layers (Keras_Advanced.ipynb#Custom-layers:-subtle-point)
- Custom loss, Custom training loop (Keras_Advanced.ipynb#Model-specialization)

Advanced Keras code: Neural Style Transfer: Non-trivial Loss function

- Neural Style Transfer (Neural_Style_Transfer.ipynb)

Putting it all together: Code: the Transformer

We now have enough background to understand the code for the Transformer.

We will examine the code in the excellent TensorFlow tutorial on the Transformer (https://www.tensorflow.org/text/tutorials/transformer)

The Transformer: Code (Transformer_code.ipynb)

Suggest reading

Tensorflow tutorial: Neural machine translation with a Transformer and Keras (https://www.tensorflow.org/text/tutorials/transformer)

Attention in detail

In our review, we had deferred a detailed view of implementing Attention. Now we will explore it at a very hight level.

We *will not* spend time on the actual code. If you're interested there are several web articles that do so, for example, here (https://machinelearningmastery.com/how-to-implement-multi-head-attention-from-scratch-in-tensorflow-and-keras/)

- Implementing Attention (Attention_Lookup.ipynb)

Functional models: subtleties

Week 5/6

Putting it all together: Code: the Transformer (continued)

We almost finished the module last week. Let's wrap up with some details related to Training.

[The Transformer: Code (continued) (Transformer_code.ipynb#Custom-Learning-Rate-Schedule)](Transformer_code.ipynb#Custom-Learning-Rate-Schedule)

Synthetic Data: Autoencoders

New major topic: Synthetic data.

After last week's "code-heavy" modules, we are back to "theory" !

We will address several ways to create new examples, staring with the simplest model and moving on to models that are more complex.

Generating synthetic data using Autoencoders and its variants.

[Autoencoder (Autoencoders_Generative.ipynb)](Autoencoders_Generative.ipynb)

Suggested Reading

[TensorFlow Tutorial on Autoencoders (https://www.tensorflow.org/tutorials/generative/autoencoder)](https://www.tensorflow.org/tutorials/generative/autoencoder)

We now study a different type of Autoencoder

- that learns a *distribution* over the training examples
- by sampling from this distribution: we can create synthetic examples

[Variational Autoeconder (VAE) (VAE_Generative.ipynb)](VAE_Generative.ipynb)

Suggested Reading

[TensorFlow tutorial on VAE (https://www.tensorflow.org/tutorials/generative/cvae)](https://www.tensorflow.org/tutorials/generative/cvae)

Further reading

[Tutorial on VAE (https://arxiv.org/pdf/1606.05908.pdf)](https://arxiv.org/pdf/1606.05908.pdf)

We finish the module on Autoencoders with the Vector Quantized Autoencoder]

- encoding produced is *discrete* rather than continuous
- facilitates sequences of mixed data types: numbers, images, speech
- interesting new Deep Learning operator: stop gradient

[Vector Quantized Autoencoder (VQ_VAE_Generative.ipynb)](VQ_VAE_Generative.ipynb)

Suggested Reading

vanilla VQ-VAE (https://arxiv.org/pdf/1711.00937.pdf)

VQ-VAE-2 paper (https://arxiv.org/pdf/1906.00446.pdf)

Synthetic Data: GANs

- GAN: basic (GAN_Generative.ipynb)
- GAN loss (GAN_Loss_Generative.ipynb)
- Wasserstein GAN (Wasserstein_GAN_Generative.ipynb)

Notebooks

- GAN to Generate Faces (CelebA_01_deep_convolutional_generative_adversarial_network.ipynb)

Suggested reading

- Generative Adversarial Nets (https://arxiv.org/pdf/1406.2661.pdf)
- TensorFlw Tutorial DCGAN (https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tut
    - this is a tutorial from which our code notebook was derived

Synthetic Data: GANs for Timeseries

Timeseries data

- Time GAN (TimeGAN_Generative.ipynb)

Further reading

Quant GAN (https://arxiv.org/pdf/1907.06673.pdf)

- Similar goal as TimeGAN
    - Uses *Temporal Convolution Networks (TCN)* for Generator and Discriminator
        - Is Dilated Convolution
        - effectively creates a longer window
    - Quant GAN used for Risk Management: lecture slides (https://cfe.columbia.edu/sites/default/files/content/slides/Modelling%20A

Synthetic Data: Evaluating the quality

Synthetic data: Evaluation (SyntheticData_Evaluation.ipynb)

- case study: Evaluation of Time GAN (TimeGAN_evaluation.ipynb)

Further reading

Frechet Inception Distance (FID) (https://arxiv.org/pdf/1706.08500.pdf#page=39)

Week 6/7 Advanced topics

Synthetic Data: Self-improvement by generating examples

LLM Self Improvement (LLM_Self_Instruction.ipynb)

- Self improvement (https://arxiv.org/pdf/2210.11610.pdf)
    - goal is to fine-tune a LLM for question answering
        - without an **a priori** fine-tuning dataset
            - use a LLM to **generate** a fine-tuning dataset
            - Use few-shot, CoT prompts:
                - Input=question; Output=answer + rationale
                - Input=question, LLM generates output
                    - multiple outputs
                    - extract answer from output
                        - - use majority voting on answer to filter responses - hopefully: majority is accurate: "high confidence" == fraction of responses that agree ?
            - The high confidence (large fraction of generated responses to a question agree in answer) generated examples become the fine-tuning dataset

- Self-instruct (https://arxiv.org/pdf/2212.10560.pdf)

Combining a LLM with external capabilities

- Extra parametric capabilities (LLM_plus_Extra_Parametric.ipynb)

WebGPT: Non-parametric knowledge: future of search ? Relate to NLP

- Non-parametric knowledge (Retriever_plus_LLM.ipynb)

DALL-E: Mixing Text and Image

Social concerns

- Model bias
    - show model cards
- Alignment
    - Alignment (Alignment.ipynb)
    - Alignment Anthropic (Alignment_Anthropics.ipynb)
    - **Deeper Dive** Reinforcement Learning (Reinforcement%20Learning.ipynb)
- Environmental
    - ML Carbon impact calculator (https://mlco2.github.io/impact/#compute)
    - Total compute: detailed calc for many models (https://arxiv.org/pdf/2005.14165.pdf#page=46)
        - Energy: train vs inference (https://arxiv.org/pdf/2005.14165.pdf#page=39)

Assignments

Your assignments should follow the Assignment Guidelines (assignments/Assignment_Guidelines.ipynb)

Final Project

Assignment notebook (assignments/FineTuning_HF/FineTune_FinancialPhraseBank.ipynb)

```
In [1]: print("Done")
```

```
Done
```