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

- Motivate Machine Learning
- Introduce notation used throughout course
- Plan for initial lectures
    - *What*: Introduce, motivate a model
    - *How*: How to use a model: function signature, code (API)
    - *Why*: Mathematical basis -- enhance understanding and ability to improve results

- Course Overview (Course_overview_NYU.ipynb)
- Machine Learning: Overview (ML_Overview.ipynb)
- Intro to Classical ML (Intro_Classical_ML.ipynb)

Using an AI Assistant

AI Assistants are often very good at coding.

But using one to just "get the answer" deprives you of a valuable tool

- you can ask the Assistant *why* it chose to do something
- keep on asking
- treat it as a private tutor !

Learning about KNN using an Assistant as a private tutor (https://www.perplexity.ai/search/using-python-and-sklearn-pleas-407oe3uzTXu1i9xEHVR2MQ)

Week 2

Recap of last week

- Summary of Intro to Supervised Machine Learning (Intro_to_Supervised_Learning_Summary.ipynb)

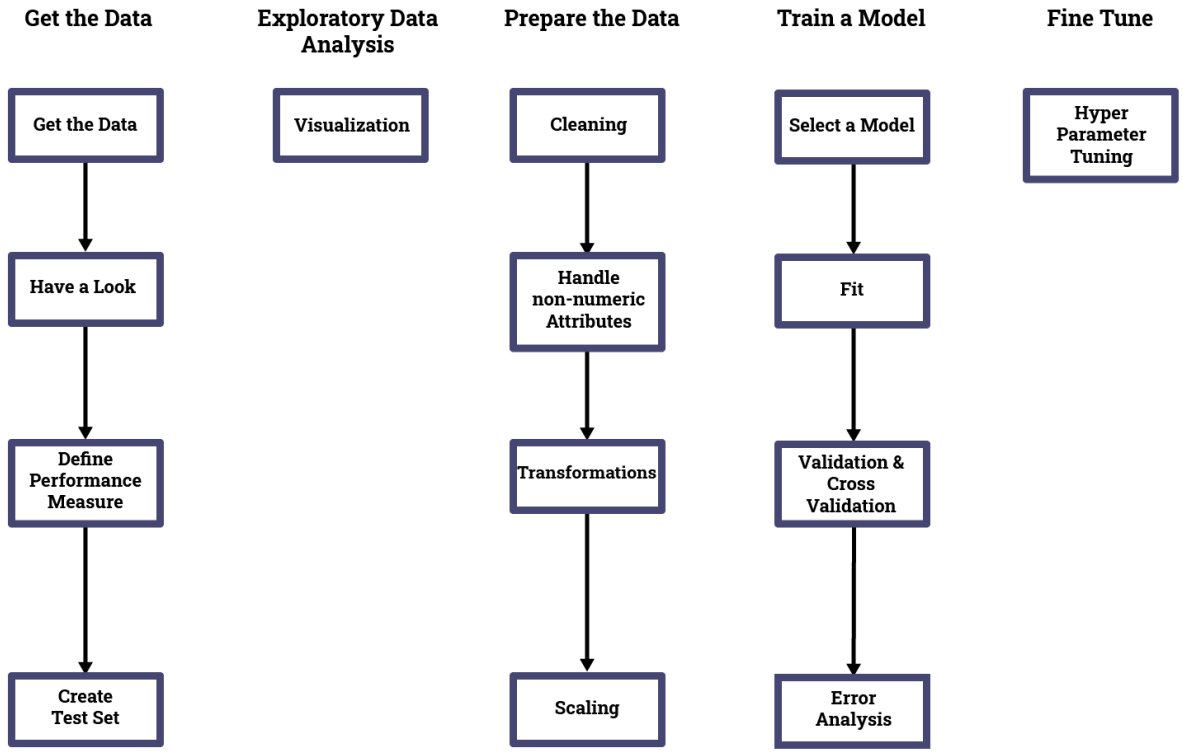
Plan

We will learn the Recipe for Machine Learning, a disciplined approach to solving problems in Machine Learning.

We will illustrate the Recipe while, at the same time, introducing a model for the Regression task: Linear Regression.

Our coverage of the Recipe will be rapid and shallow (we use an extremely simple example for illustration).

I highly recommend reviewing and understanding this Geron notebook (external/handson-ml2/02_end_to_end_machine_learning_project.ipynb) in order to acquire a more in-depth appreciation of the Recipe.



Recipe, as illustrated by Linear Regression

[The Recipe for Machine Learning: Solving a Regression task (Recipe_via_Linear_Regression.ipynb)]

- A *process* for Machine Learning
    - Go through the methodical, multi-step process
        - Quick first pass, followed by Deeper Dives

Fitting a model: details

Recall: fitting a model (finding optimal value for the parameters) is found by minimizing a Loss function.

Let's examine a typical Loss function for Regression

- Regression: Loss Function (Linear_Regression_Loss_Function.ipynb)

Iterative training: when to stop

Increasing the number of parameters of a model improves in-sample fit (reduces Loss) but may compromise out-of-sample prediction (generalization).

We examine the issues of having too many/too few parameters.

- When to stop iterating: Bias and Variance (Bias_and_Variance.ipynb)

Get the data: Fundamental Assumption of Machine Learning

- Getting *good* training examples (Recipe_Training_data.ipynb)

Regression: final thoughts (for now)

- Regression: coda (Regression_coda.ipynb)

Deeper dives

- Fine tuning techniques (Fine_tuning.ipynb)

Using an AI Assistant

Learning about Linear Regression using an Assistant as private tutor (https://www.perplexity.ai/search/using-python-sklearn-and-matpl-vTYv7oGdRQ6upR5L5OSirg)

Week 3

Recipe "Prepare the Data" step: Transformations

We covered this at end of Week 2

We recap the importance of adding *synthetic* features to our Linear Regression example

- and *preview* the *mechanical* process of creating these features via *Transformations*

Transformations

- Prepare Data: Intro to Transformations (Prepare_data_Overview.ipynb)

Validation

We skipped over this topic in Week 2; we cover it now

Our test dataset can be used only once, yet

- we have an iterative process for developing models
- each iteration requires a proxy for out of sample data to use in the Performance Metric

The solution: create a proxy for out of sample that is a *subset* of the training data.

- Validation and Cross-Validation (Recipe_via_Linear_Regression.ipynb#Validation-and-Cross-Validation)
- Avoiding cheating in Cross-Validation (Prepare_data_Overview.ipynb#Using-pipelines-to-avoid-cheating-in-cross-validation)

Classification Task

Plan

- We introduce a model for the Classification task: Logistic Regression
- How to deal with Categorical (non-numeric) variables
    - classification target
    - features

Classification intro

- Classification: Overview (Classification_Overview.ipynb)
- Classification and Categorical Variables (Classification_Notebook_Overview.ipynb)
    - linked notebook (Classification_and_Non_Numerical_Data.ipynb)

Categorical variables (contained as subsections of Classification and Categorical Variables)

We examine the proper treatment of categorical variables (target or feature).

Along the way, we run into a subtle difficulty: the Dummy Variable Trap.

- Classification and Categorical Variables: Categorical Variables (Classification_Notebook_Overview.ipynb#Categorical-variables)

Week 4

Multinomial Classification

We generalize Binary Classification into classification into more than two classes.

- Multinomial Classification (Multinomial_Classification.ipynb)

Classification and Categorical variables wrapup

- Classification Loss Function (Classification_Loss_Function.ipynb)
- Baseline model for Classification (Classification_Baseline_Model.ipynb)
- OHE issue: Dummy variable trap (Dummy_Variable_Trap.ipynb)

Classification: final thoughts (for now)

- Classification: coda (Classification_coda.ipynb)

Plan

Good news

- You now know two main tasks in Supervised Learning
    - Regression, Classification
- You now know how to use virtually every model in `sklearn`
    - Consistent API
        - `fit`, `transform`, `predict`
- You survived the "sprint" to get you up and running with ML
- You know the *mechanical process* to implement transformations: Pipelines

Time to re-visit, in more depth, several important topics

Error Analysis

- We explain Error Analysis for the Classification Task, with a detailed example
- How Training Loss can be improved
- Error Analysis (Error_Analysis_Overview.ipynb)
    - linked notebook (Error_Analysis.ipynb)
        - Summary statistics
        - Conditional statistics
    - Worked example (Error_Analysis_MNIST.ipynb)
- Loss Analysis: Using training loss to improve models (Training_Loss.ipynb)

Imbalanced data

- Imbalanced data (Imbalanced_Data.ipynb)

Transformations: the ”why”

Part of becoming a better Data Scientist is transforming raw features into more useful synthetic features.

In an earlier week, we presented the mechanics (Prepare_data_Overview.ipynb) (how to use `sklearn` to implement transformation Pipelines) of Transformations. This week, we focus on the necessity (the "why"): transforming raw data into something that tells a story

- Becoming a successful Data Scientist (Becoming_a_successful_Data_Scientist.ipynb)
- Transformations: overview (Transformations_Overview.ipynb)
    - linked notebooks:

Week 5

Transformations: the "why" (continued)

- Becoming a successful Data Scientist (Becoming_a_successful_Data_Scientist.ipynb)
- Transformations: overview (Transformations_Overview.ipynb#Scaling)
    - linked notebooks:
        - Transformations: scaling (Transformations_Scaling.ipynb)
        - Transformations: normalization (Transformations_Normalization.ipynb)
        - Other Transformations (Transformations_Other.ipynb)

Loss functions: mathematical basis

Where do the Loss functions of Classical Machine Learning come from ? We take a brief mathematical detour into Loss functions.

- Entropy, Cross Entropy, and KL Divergence (Entropy_Cross_Entropy_KL_Divergence.ipynb)
- Loss functions: the math (Loss_functions.ipynb)
    - Maximum likelihood
    - Preview: custom loss functions and Deep Learning

More models for classification

Plan

- More models: Decision Trees, Naive Bayes
    - Different flavor: more procedural, less mathematical
    - Decision Trees: a model with *non-linear* boundaries
- Ensembles
    - Bagging and Boosting
    - Random Forests

Decision Trees, Ensembles

- Decision Trees: Overview (Decision_Trees_Overview.ipynb)
- Decision Trees (Decision_Trees_Notebook_Overview.ipynb)
    - linked notebook (Decision_Trees.ipynb)
- Trees, Forests, Ensembles (Ensembles.ipynb)

Naive Bayes

- Naive Bayes (Naive_Bayes.ipynb)

Week 6

More models for classification

Plan

Continue with more models for classification.

We continue with the *ensemble* technique that *combines* the prediction of multiple models.

Combining multiple models: Ensembles (continued)

- Trees, Forests, Ensembles (Ensembles.ipynb#Boosting)

Support Vector Classifiers

- Support Vector Machines: Overview (SVM_Overview.ipynb)
- SVC Loss function (SVM_Hinge_Loss.ipynb)
- SVC: Large Margin Classification (SVM_Large_Margin.ipynb)
- SVM: Kernel Transformations (SVM_Kernel_Functions.ipynb)
- SVM Wrapup (SVM_Coda.ipynb)

Classification: final thoughts

- Classification: coda (Classification_coda.ipynb)

Loss functions: mathematical basis (deferred from previous week)

Where do the Loss functions of Classical Machine Learning come from ? We take a brief mathematical detour into Loss functions.

- Entropy, Cross Entropy, and KL Divergence (Entropy_Cross_Entropy_KL_Divergence.ipynb)
- Loss functions: the math (Loss_functions.ipynb)
    - Maximum likelihood
    - Preview: custom loss functions and Deep Learning

Unsupervised Learning

Unsupervised Learning: PCA

- Unsupervised Learning: Overview (Unsupervised_Overview.ipynb)
- PCA Notebook Overview (Unsupervised_Notebook_Overview.ipynb)
    - linked notebook (Unsupervised.ipynb)
- PCA in Finance (PCA_Yield_Curve_Intro.ipynb)

Week 7

We continue with the Unsupervised Learning topic.

Unsupervised Learning

Unsupervised Learning: PCA (continued)

- Importance of number of components: visualization (Unsupervised.ipynb#Visualizing-the-fidelity-of-the-reduced-dimension-representation)
- Interpreting the components (Unsupervised.ipynb#Can-we-interpret-the-components-?)

Classical ML: deeper dives

Loss functions: mathematical basis (deferred)

Where do the Loss functions of Classical Machine Learning come from ? We take a brief mathematical detour into Loss functions.

- Entropy, Cross Entropy, and KL Divergence (Entropy_Cross_Entropy_KL_Divergence.ipynb)
- Loss functions: the math (Loss_functions.ipynb)
    - Maximum likelihood
    - Preview: custom loss functions and Deep Learning

Deeper Dives

- Linear Regression in more depth (Linear_Regression_fitting.ipynb)
- Interpretation: Linear Models (Linear_Model_Interpretation.ipynb)
- Missing data: clever ways to impute values (Missing_Data.ipynb)
- Feature importance (Feature_Importance.ipynb)
- SVC Loss function derivation (SVM_Derivation.ipynb)

Bridge between Classical ML and Deep Learning

Gradient Descent

Machine Learning is based on minimization of a Loss Function. Gradient Descent is one algorithm to achieve that.

- Gradient Descent (Gradient_Descent.ipynb)

Recommender Systems (Pseudo SVD)

How does Amazon/Netflix/etc. recommend products/films to us ? We describe a method similar to SVD but that is solved using Gradient Descent.

This theme of creating a custom Loss Functions and minimizing it via Gradient Descent is a recurring theme in the upcoming Deep Learning second half of the course.

- Recommender Systems (Recommender_Systems.ipynb)
- Preview: Some cool Loss functions (Loss_functions.ipynb#Loss-functions-for-Deep-Learning:-Preview)

Deeper Dives

Deep Learning

DL Week 1 Introduction to Neural Networks and Deep Learning

Plan

Deep Learning/Neural networks

- Set up your Tensorflow environment (Tensorflow_setup.ipynb)
- Neural Networks Overview (Neural_Networks_Overview.ipynb)

Neural network: practical

- Coding Neural Networks: Tensorflow, Keras
    - Intro to Keras (Tensorflow_Keras.ipynb)
    - **Note**
        - If you have problems using the `plot_model` function in Keras on local machine: see here (Setup_ML_Environment_NYU.ipynb#Too visualization-of-graphs-(optional)) for a fix.
    - Linked notebooks
        - DNN Tensorflow example Notebook local (DNN_TensorFlow_example.ipynb) **(local machine)**
        - DNN Tensorflow example Notebook from github (https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/DNN_TensorFlow_example (**Google Colab**)
- Practical Colab
    - **Colab**: Practical Colab Notebook from github (https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/Colab_practical.ipynb)

Practical advice

- Karpathy: Recipe for training Neural Nets

DL Week 2 Intro to NN (continued); Convolutional Neural Networks

Here is a quick review of Neural Networks

- Neural Network summary (Neural_Network_summary.ipynb)

Practical advice (continued)

- Karpathy: Recipe for training Neural Nets (Karpathy_Recipe_for_training_NN.ipynb)

Plan

The topics introduced in the Neural Networks Overview are now covered more in-depth.

- Where do Neural Networks get their power from ?
- How exactly do we compute the gradients ?
- How does a special language/library facilitate automatic computation of the gradients ?

Neural network theory

- A neural network is a Universal Function Approximator (Universal_Function_Approximator.ipynb)

Training Neural Networks (introduction)

- Intro to Training (Neural_Networks_Intro_to_Training.ipynb)
- Training Neural Networks - Back propagation (Training_Neural_Network_Backprop.ipynb)

How to compute gradients automatically

- Why TensorFlow ?: Gradients made easy (Training_Neural_Network_Operation_Forward_and_Backward_Pass.ipynb)

Deeper Dives

- Keras, from past to present (Tensorflow_Keras_Archaeology.ipynb)
- History/Computation Graphs: Tensorflow version 1 (DNN_TensorFlow_Using_TF_version_1.ipynb)
- Raw_TensorFlow example Notebook from github (https://colab.research.google.com/github/kenperry-public/ML_Spring_2024/blob/master/Raw_TensorFlow.ipynb) (**Colab**)

Plan

We introduce a new layer type.

This is motivated by inputs with dimensions in addition to the feature dimension.

The Convolutional Neural Network layer type

- Introduction to CNN (Intro_to_CNN.ipynb)
    - CNN pictorial (CNN_pictorial.ipynb)
- Notational standards, definitions (CNN_Notation.ipynb)
- CNN: Space and Time (CNN_Space_and_Time.ipynb)
    - CNN example from github (https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/CNN_demo.ipynb) (**Colab**)
    - CNN example from github (CNN_demo.ipynb) (**local machine**)

The following notebooks are an older attempt at a *visual* explanation of the CNN.

Hopefully, the "Introduction" notebook is more intuitive and may supercede these visual notebooks.

- CNN: explained in pictures (CNN_Overview.ipynb)

Deeper dives

- Convolution as Matrix Multiplication (CNN_Convolution_as_Matrix_Multiplication.ipynb)

DL Week 3 Training Neural Networks: details

Convolutional Neural Networks (CNN) wrap-up

- Remaining details (CNN_Space_and_Time.ipynb#Kernel-size-1)

Plan

- Why training a Neural Network can be difficult: fine-details of training
- Introduce a new layer type: Recurrent layers
    - Part of our "sprint": final layer type
    - Will revisit more theoretical issues in subsequent lectures

Training Neural Networks: the fine details

- The dynamics of training (Training_Neural_Networks_Overview.ipynb)
    - Effects of changing: activation functions; weight initialization
    - initialization and scaling
    - dropout
    - learning rate schedules
    - vanishing/exploding gradients

Recurrent Neural Networks (RNN)

- Introduction to Recurrent Neural Network (RNN) (Intro_to_RNN.ipynb)

DL Week 4 Recurrent Neural Networks

Recurrent Neural Networks (RNN) Continued

- Recurrent Neural Network Overview (RNN_Overview.ipynb)
    - linked notebook: RNN in code -- Imdb sentiment classification (Keras_examples_imdb_cnn.ipynb#Try-an-LSTM-as-a-means-of-obtaining-a-finite-length-representation-of-the-sequence)

RNN: Issues

- Gradients of an RNN (RNN_Gradients.ipynb)
- RNN: Gradients that Vanish/Explode (RNN_Vanishing_and_exploding_gradients.ipynb)
- RNN: Visualization (RNN_Visualization.ipynb)

Sprint is over ! We have covered the basic layer types; time for you to learn by experimenting.

Review of layer types

- What layer type to choose (Neural_Network_Layer_Review.ipynb)

Deeper dives

- RNN: How to deal with long sequences (RNN_Long_Sequences.ipynb)
- Using an RNN for Generative AI (RNN_generative.ipynb)
    - LSTM_text_generation from github (https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/Keras_examples_LSTM_text_generat (**Colab**)
    - LSTM_text_generation from github (Keras_examples_LSTM_text_generatio (**local machine**)

DL Week 5 Transfer Learning; Natural Language Processing

Advanced Recurrent Architectures: LSTM

Plan

The "vanilla" Recurrent Neural Network (RNN) layer we learned is very much exposed to the problem of vanishing/exploding gradients.

We will review the issue and demonstrate a related layer type (the LSTM) designed to mitigate the problem.

We present an extremely useful trick (Transfer Learning) for leveraging the hard work that others have done.

Concepts

There are a number of pieces of the LSTM which can appear overwhelming when seen together for the first time. We will explore these concepts separately before seeing how they are integrated into the LSTM.

- Residual connections (RNN_Residual_Networks.ipynb)
- Neural Programming (Neural_Programming.ipynb)

LSTM: An improved RNN

- Introduction to the LSTM (Intro_to_LSTM.ipynb)
- LSTM Overview (LSTM_Overview.ipynb)

Transfer Learning

Transfer learning allows us to adapt a model trained for one task to be able to solve a new task with a small amount of work. As models get bigger and bigger, the future of Deep Learning may be one where you use Transfer Learning more than developing your own models from scratch.

- Transfer Learning (Continued) (Transfer_Learning.ipynb)
    - Transfer Learning example from github (https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/TransferLearning_demo.ipynb) (**Colab**)
    - Transfer Learning example from github (TransferLearning_demo.ipynb) (**local machine**)
    - Utility notebook (Dogs_and_Cats_reformat.ipynb)
        - Takes the *very large* raw data (from Kaggle) used in the Transfer Learning example
        - Creates a much smaller subset, using a different directory structure
        - The above notebook uses this reorganized, smaller subset

NLP

Plan

We will make an initial pass on the topic of learning from text: Natural Language Processing.

The first pass will use well-established techniques that are relatively easy to follow.

We then explore some recent advances that have greatly increased the power of NLP.

The Transformer architecture is a key contributor.

Learning from text: Deep Learning for Natural Language Processing (NLP)

- Natural Language Processing Overview (NLP_Overview.ipynb)

We revisit some code we had previously studied

- in the RNN module: to illustrate various ways to eliminate the time dimension
- but this time with an emphasis on the NLP aspects
    - NLP from github (Colab) (https://colab.research.google.com/github/kenperry-public/ML_Spring_2025/blob/master/Keras_examples_imdb_cnn.ipynb)
    - NLP from github (local machine) (Keras_examples_imdb_cnn.ipynb)

Evolution of Word representations

- How to represent a word: syntax (NLP_Tokenization.ipynb)
- How to represent a word: meaning (NLP_Word_Representations.ipynb)

DL Week 6 NLP Continued/Transformers

NLP (continued)

- Natural Language Processing Overview (continued) (NLP_Overview.ipynb#Embeddings)

Evolution of Word representations

- How to represent a word: syntax (NLP_Tokenization.ipynb)
- How to represent a word: meaning (NLP_Word_Representations.ipynb)

Transformers

Plan

We present Attention, a way to enhance the power of RNN's, which is heavily used in a new layer type for sequence processing: the Transformer.

The Transformer layer type is now predominant in the area of Natural Language Processing (NLP). We give a quick introduction but we will revisit it in the module on advanced NLP.

Attention

- Attention: motivation (Intro_to_Attention.ipynb)

We summarize (with illustrations) the key points regarding attention.

- The evolution of the Encoder/Decoder RNN to
    - a "loop-free" architecture via Self-Attention
    - combined with Cross attention from Encoder to Decoder
- Transformer motivation: Illustrated (Attention_motivation_illustrated.ipynb)

These ideas are combined into a new architecture called the **Transformer**

Transformers

- Transformer (Transformer.ipynb)

Attention: in depth

- Implementing Attention (Attention_Lookup.ipynb)

Deeper dives

- Transformer
    - Keras example: pre-defined Attention layers (https://keras.io/examples/nlp
        - notebook (https://colab.research.google.com/github/keras-team/
          io/blob/master/examples/nlp/ipynb/text_classification_with_trans
    - TensorFlow tutorial: implements Attention, positional encoding (https://www.tensorflow.org/text/tutorials/transformer)
        - notebook (https://colab.research.google.com/github/tensorflow/text/blob/r

Further reading

- Attention
    - Neural Machine Translation by Jointly Learning To Align and Translate (https://arxiv.org/pdf/1409.0473.pdf)
    - Attention is all you need (https://arxiv.org/pdf/1706.03762.pdf)

Language Models

Language Models: the future (present ?) of NLP ?

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

DL Week 7 Large Language Models/Interpretation

Language Models (continued)

Having introduced the Language Model objective, we now explore the growth of Large Language Models and their power.

- Large Language Models and Beyond (Review_LLM.ipynb#Large-Language-Models-(LLM)-and-Beyond)

Scaling LLM's

We now have the capabilities to build models with extremely large number of weights. Is it possible to have too many weights ?

Yes: weights, number of training examples and compute capacity combine to determine the performance of a model.

How large should my Transformer be ? (Transformers_Scaling.ipynb)

Suggested reading

- Scaling laws (https://arxiv.org/pdf/2001.08361.pdf)

Further reading

- Inference budget
    - Transformer Inference Arithmetic (https://kipp.ly/transformer-inference-arithmetic/)
    - LLaMA: Open and Efficient Foundation Language Models (https://arxiv.org/pdf/2302.13971.pdf)
    - Large language models aren't trained enough (https://finbarr.ca/llms-not-trained-enough/)

Interpretation

As we've hinted at before: the art of Deep Learning is writing a Loss function that captures the semantics of the problem you want to solve.

- We've spent several weeks learning about Neural Networks. At best, we suggested theories for how they are able to achieve what they do.

This week: we will explore methods for testing theories as to how Neural Networks work.

What is a Neural Network really doing ? Interpretation

- Introduction to Interpretation of Deep Learning (Intro_to_Interpretation_of_DL.ipynb)
- Interpretation: Simple Methods (Interpretation_of_DL_Simple.ipynb)
- interpretation: Gradient Ascent (Gradient_ascent.ipynb)
- Adversarial Examples (Adversarial_Examples.ipynb)

Advanced Keras (preview of the Advanced Course on Deep Learning)

It turns out that we can't implement these models using the Sequential model of Keras that we have been using all semester:

- it's time to give a peek at the Functional model.

We look at more advanced Keras techniques that allow us to write complex Loss functions.

- Additionally, we will look at other advanced features such as
    - Custom training loop
    - Cool things you can do with Gradients
    - Defining your own layer type
- Advanced Keras (Keras_Advanced.ipynb)

Wrapping up

- Final thoughts (Deep_Learning_Coda.ipynb)

Additional Deep Learning resources

Here are some resources that I have found very useful.

Some of them are very nitty-gritty, deep-in-the-weeds (even the "introductory" courses)

- For example: let's make believe PyTorch (or Keras/TensorFlow) didn't exists; let's invent Deep Learning without it !
    - You will gain a deeper appreciation and understanding by re-inventing that which you take for granted

Andrej Karpathy course: Neural Networks, Zero to Hero (https://karpathy.ai/zero-to-hero.html)

- PyTorch
- Introductory, but at a very deep level of understanding
    - you will get very deep into the weeds (hand-coding gradients !) but develop a deeper appreciation

fast.ai

`fast.ai` is a web-site with free courses from Jeremy Howard.

- PyTorch
- Introductory and courses "for coders"
- Same courses offered every few years, but sufficiently different so as to make it worthwhile to repeat the course !
    - Practical Deep Learning (https://course.fast.ai/)
    - Stable diffusion (https://course.fast.ai/Lessons/part2.html)
        - Very detailed, nitty-gritty details (like Karpathy) that will give you a deeper appreciation

Stefan Jansen: Machine Learning for Trading (https://github.com/stefan-jansen/machine-learning-for-trading)

An excellent github repo with notebooks

- using Deep Learning for trading
- Keras
- many notebooks are cleaner implementations of published models

Assignments

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

Regression

- Assignment notebook: Using Machine Learning for Hedging (assignments/Regression%20task/Using_Machine_Learning_for_Hedging.ipynb)
- Data
    - There is an archive file containing the data
    - You can find it
        - Under the course page: Content --> Data --> Assignments --> Regression task
        - You won't be able to view the file in the browser, but you **will** be able to Download it
    - You should unzip this archive into the *the same directory* as the assignment notebook
    - The end result is that the directory should contain
        - The assignment notebook and a helper file
        - A directory named `Data`

Classification

- Assignment notebook: Ships in satellite images (assignments/Classification%20task/Ships_in_satellite_images.ipynb#)
- Data
    - There is an archive file containing the data
    - You can find it
        - Under the course page: Content --> Data --> Assignments --> Classification task
        - You won't be able to view the file in the browser, but you **will** be able to Download it
    - You should unzip this archive into the *the same directory* as the assignment notebook
    - The end result is that the directory should contain
        - The assignment notebook and a helper file

Midterm Project: Bankruptcy One Year Ahead

- Assignment notebook Bankruptcy One Year Ahead (assignments/bankruptcy_one_yr/Bankruptcy_oya.ipynb)
- Data
    - There is an archive file containing the data
    - You can find it
        - Under the course page: Content --> Data --> Assignments --> Bankruptcy One Year Ahead
        - You won't be able to view the file in the browser, but you **will** be able to Download it
    - You should unzip this archive into the *the same directory* as the assignment notebook
    - The end result is that the directory should contain
        - The assignment notebook and a helper file
        - A directory named `Data`

Keras practice

- Assignment notebook Ships in satellite images: Neural Network (assignments/keras_intro/Ships_in_satellite_images_P1.ipynb)
- Data (same as for the Classification assignment)

Convolutional Neural Networks (CNN)

- Assignment notebook Ships in satellite images: Neural Network (assignments/CNN_intro/Ships_in_satellite_images_P2.ipynb)
- Data (same as for the Classification assignment)
    - please repeat the directions given in that assignment for obtaining the data

Final project; Stock prediction

- Assignment notebooks:
    - Stock prediction (assignments/stock_prediction/Final_project_StockPrediction.ipynb)
    - Submission guidelines (assignments/stock_prediction/Final_project.ipynb)
- Data
    - There is an archive file containing the data
    - You can find it
        - Under the course page: Content --> Data --> Assignments --> Stock Prediction
        - You won't be able to view the file in the browser, but you **will** be able to Download it
    - You should unzip this archive into the *the same directory* as the assignment notebook
    - The end result is that the directory should contain
        - The assignment notebook, submission guidelines notebook
        - A directory named `Data/train`

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

```
Done
```