Machine Learning Engineer

A complete ML study path, focused on TensorFlow and Scikit-Learn

I **strongly recommend** you to buy **this** phenomenal book: "Hands-On Machine Learning with Scikit-Learn and TensorFlow" by OreillY, which inspired me and has driven most of the organization and hierarchy of the content listed below.

Apart from this, **everything** listed here is open source and free, most of it coming from the world-renowned universities and open source associations.

When we learn something new, especially if wide and complex, it is necessary to avoid confusion, so I tried to create the next steps of the path preferring contents from the same context and authors, when possible. When not possible, I collected both theory and examples, as well as some pointers to resources like "best practices for ___".

I organized the Path in 4 sections:

Prerequisites

- Python
- Jupyter Notebook
- The Math you need
- The Machine Learning landscape

Machine learning with Scikit-Learn

- Why Scikit-Learn?
- End-to-End Machine Learning project
- Linear Regression
- Classification
- Training models
- Support Vector Machines
- Decision Trees
- Ensemble Learning and Random Forest
- Unsupervised Learning --- **new**
- Wrapping up and looking forward

Neural Networks with TensorFlow

- Why TensorFlow?
- Up and Running with TensorFlow
- ANN - Artificial Neural Networks
- CNN - Convolutional Neural Networks
- RNN - Recurrent Neural Networks
- Training Networks: Best practices
- AutoEncoders

- Reinforcement Learning
- Next steps

Utilities

- Machine Learning Projects
- Data Science Tools
- Blogs / Youtube Channels / Websites worth taking a look!

So let's get started!

Prerequisites

Python

According to Sun Tzu:

> If you don't know Python, learn it yesterday!

Python is one of the most used and loved programming languages, and it's necessary to get things done in the Machine Learning field. Like most of the frameworks of the bigger Data Science field, TensorFlow is married with Python and Scikit-Learn is written in Python.

First, let's install Python 3 on your machine!

We are ready to start our journey!

If you don't know the basics of Python, just start from here.\ Else if you know the syntax and you want to have a more solid Python background (recommended) take this Intermediate Python Course from here.\ If you are looking for tons of exercises to get your hands dirty and get experience with Python, check here and here.

Once you're familiar with Python, take a look at Numpy, an important module for math operations, that allows you to import in Python the Tensor data type, which is the most used in ML (especially when dealing with Neural Nets). A tensor is not a matrix! This is an awesome Numpy Tutorial.

I also recommend you to install the Pycharm Community Edition, a complete IDE for Python development, and set a new Python virtual environment for your experiments.

Jupyter Notebook

Directly from here: "The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more." Working with data means -> a lot of expriments. And to document experiments, and organize them in a valuable way to get insights, you definitely need to use Jupyter Notebook during your journey. Why?

The math you need

Whoever tells you that the math behind Machine Learning is hard... is not so wrong! But you have to consider that every time you're going to use it, it will be handled by the machine for you! So, the important is to grasp the main concepts and recognize limits and applications of those. No one is going to ask you to calculate a gradient by hand! So, if you are not familiar with these concepts, check them, because they are the reason behind everything.

With these three resources, you'll get out the most of what you really need to understand deeply.

A top course about linear algebra is here.\ Integrate with basic probabilities and statistic concepts here.\ The most of the remaining math you need here.

The machine learning Landscape

Directly from the book cited earlier, this is the most concise and illuminating overview of **what is** and **when you need** machine learning. Let's stop using buzzwords! Check it here.

Machine Learning with Scikit-Learn

To install Scikit-Learn

```
python pip install -U scikit-learn
```

If you encounter some problems, it may be because you don't have the latest version of pip. So in the same folder run:

```
 python -m pip install --upgrade pip
```

Why Scikit-Learn

Scikit-Learn is one of the most complete, mature and well-documented libraries for Machine Learning tasks. It comes out-of-the-box with powerful and advanced models and offers facility functions for the data science process.

We'll learn and use other modules along the road, for a quick usage just look at their official documentation.

End-to-End Machine Learning project

For a first taste, I suggest you go through this Kaggle notebook, which has a classical example of an ML task. The goal is trying to predict if a Titanic passenger would have been most likely to survive or not. Many things will be unclear for now, but don't worry, they will all be explained comprehensively later. It is nice to get the picture of the "applied" project, going through the classical steps of the applied Machine Learning (problem framing, data exploration, question formulation...).

The notebook is on Kaggle, the go-to platform for ML and general Data Science projects, which provides a lot of free datasets and offers interesting challenges and ML model experiments.

Remember: Read it, trying to get the big picture of the process, because some details, functions and code will be clearer later.

Linear Regression

This is the simplest form of Machine Learning, and the starting point for everyone interested in predicting outcomes from a dataset. Check here the theoretical lesson from Andrew NG and then go through these examples, from the simplest to the most complex. This is the math behind Linear Regression.

- Example 1
- Example 2
- Example 3

Classification

Classification is one of the most important ML tasks, when wanting to predict an outcome out of different possibilities. For example, given handwritten digits, classify them with the lowest error possible. The simplest case is binary classification (Yes or No, Survived or Not Survived), have a look here. Check here for a brief explanation of the theory of logistic regression for classification, and check here for a deeper comprehension (using the Titanic dataset). You can use a lot of different ML models to classify things, even neural networks! For now, just take a look here, where you see an example of accuracy and recall comparison among different models. Here you have an article about the metrics used to **evaluate** your classifiers.

Training models

Here I grouped some of the techniques used in ML tasks to train the models. In this Google Crash Course you find:

- Gradient Descent
- Learning Rate
- SGD
- Regularization

Support Vector Machines

This is another classical algorithm to create ML models. Here you have the explanation of the theory, and here a more pratical approach. Check both. Here is a very good explanation + practice application in Scikit-Learn.

Decision Trees

Decision Trees are one of the most simple but effective ideas behind predicting outcomes, and they're used in many ways (i.g. Random Forest). Check here and go through the playlist to get a theoretical overview of Decision Trees (ID3). Here you have the pratical application of ID3. Here you have some end-to-end examples with Scikit-Learn:

- Example 1
- Example 2
- Example 3

Ensemble Learning and Random Forest

The idea of Ensemble Learning is to leverage all the different features, pros and cons of several ML models to obtain a group of "voters" that, for each prediction, gives you the most likely outcome, voted by different classifiers (SVM, ID3, maybe Logistic Regression). Here you get the basics of ensemble learning model, and here you find the most classic of them, the Random Forest. Altough the idea is simple, this ensemble model turned out to be really effective tackling even some "hard" classification problems, or with a lot of data.

Here you get a complete overview of the best practices for ensemble learning, and here you find an example of Random Forest with Scikit-Learn. Both links come with a bunch of useful techniques to use in practice.

Unsupervised Learning

First look (in order):

- Here you have a brief introductionary video.
- This article explains Unsupervised Learning really well.
- Here is an interesting read about the difference among Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

The two most important techniques here are Association Rules Exploration and Clustering. I provide examples and tutorials for both.

Association Rules tutorials and examples: 1, 2, 3, 4, 5.

Clustering tutorials and examples: 1, 2, 3, 4, 5, 6.

Second pass: Stanford slides. MIT slides.

Tips & Best practices: 1, 2, 3, 4, 5.

Wrapping up and looking forward

Now, if you followed all the steps and explored all the resources I posted, you're likely to be more confident with Machine Learning and have a general idea of things. Of course you need to explore and learn more, because this field is changing and enhancing techniques and approaches day-by-day! All the algorithms we've seen are widely used in the Data Science and Analytics fields, but there are some complex tasks where they fail or give really poor performances. Now we are ready to fall down in the **deep** rabbit hole, trying to understand how Neural Networks and in general Deep Learning can help tackling big problems with millions of parameters and variables. Why use Deep Learning over classical ML algorithms?

Neural Networks with TensorFlow

In this section we'll follow a track that will take us from zero knowledge about neural networks to fully understanding them, thanks to the Stanford University Deep Learning course and some tutorials I've searched over the internet. Some of them come from Google, others from Stanford or Cambridge universities, and you will learn to leverage neural networks (ANN, CNN, RNN) for several kinds of ML tasks. These are some use cases of using TensorFlow for ML tasks.

The theory and the applications of the Neural Networks is not too easy to get at a first look. Because of that, you'll need to pass again through tutorials and videos, to ensure a fully comprehension of the coming topics. Because of that, I spent a decent amount of time trying to understand (reading paths like this, articles, offical forums, related subreddits) which was **the most effective way** to deeply learn the concepts, formulas, tradeoffs... I came up with this approach, but you can tweak it as you prefer, because every brain is different.

> After taking the TensorFlow section, apply this 3-step iterative cycle:
>
> - 1 Get an idea of the main concepts through an **entire pass of this** Stanford course, don't worry too much about the math explanations, focus on the **what and why**.
> - 2 Deeply explore **one topic at time**, with theory + tutorials + examples (e.g. RNN theory + RNN tutorials + RNN examples)
> with the links and resources of the topic section of the guide.
> - 3 After iterating the 2nd step for each topic, walk again through the entire Stanford course. This time you can fully understand all the formulas, connecting them and catching also the "math flow" of the course.

This iterative process (1-2-2-2-2.....-3) can be repeated as many times as you want, and will probably construct in your mind a nice **general schema** of the things. In each complete iteration you can drop one or more topics, and focus on the ones that are more interesting to you or not so clear.

In each section I've put content for the first time you arrive there (during the first complete iteration), and some content for next time you arrive there (after the 3rd step).

The structure follows the track proposed by the awesome Stanford course. You can find the slides here.

This is an alternative course from MIT, more or less the same contents. It's worth watching it to compare and have a different point of view on the things you are learning, besides listening 2X to the best professors of the world exploring each topic.

This is the **Book** I refer to in each section.

Why TensorFlow?

Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++. TensorFlow is the de-facto standard for the major industry-sized companies that need to implement Machine Learning algorithms. It is built for scaling, with really cool features to parallelize training over multiple GPU's or devices.

Up and Running with TensorFlow

Assuming you have Python stored in the variable PATH, to install the Tensorflow library you just need to open a terminal inside you Python installation folder and run this command.

```
python pip install tensorflow
```

The first read I recommend you is this. The second thing to do is to follow this Introduction to TensorFlow directly from the **awesome** Google Education page. Again, some theoretical concepts might be unclear, but focus on how the TensorFlow library and processes are conceived. This is a good resume of the latter. Another beginner tutorial from google. This is about the TensorFlow 2.0 update. These 1 and 2 explain the "hard" things to grasp of TensorFlow. Higly recommended.

Now you're most likely familiar with **TensorFlow as a tool**, and it's time to understand **how to use** it to build large scale Neural Networks.

ANN - Artificial Neural Networks

First look (in order):

- This video.
- This is your bible, understand it totally.
- This is a gem and read this from the authors.
- This is a really fast-talking guy implementing a Neural Network library from scratch, super useful to understand how the core of an NN is implemented in Python. You can imagine that each existing framework is just an enormous expansion of this concept-library.
- This is a step-by-step backpropagation example with calculus.

Second pass:

- ANN Chapter.

Tips & Best practices: 1, 2, 3, 4, 5, 6, 7, 8.

CNN - Convolutional Neural Networks

First look (in order):

- Here is an awesome deep explanation.
- Here another super good one.
- Here is a serious CNN tutorial with TensorFlow.

Second pass:

- CNN Chapter.

Tips & Best practices: 1, 2, 3, 4, 5, 6, 7, 8.

RNN - Recurrent Neural Networks

First look (in order):

- Here, a gentle but detailed explanation.
- Here is another interesting explanation.
- Here is a video with a more pratical approach.
- Here you have an amazing interactive demo.
- Here, a guide to implement RNN in TensorFlow.
- Here, a 7-page long blog post regarding the TensorFlow implementation.

Second pass:

- RNN Chapter

Tips & Best practices: 1, 2, 3, 4, 5, 6, 7.

Training Networks: Best practices

First look (in order): I **strongly recommend** you to refer to this page from Stanford and go through all the Module 1 and 2. I also put here a list of the various topics to explore when talking about *how to train NN in real life applications*.

- Overfitting vs Underfitting: 1, 2, 3, 4, 5.
- Vanishing/Exploding Gradient: 1, 2, 3, 4, 5.
- Transfer Learning: 1, 2, 3, 4, 5.
- Faster Optimizers: 1, 2, 3, 4.
- Avoiding Overfitting through Regularization: 1, 2, 3, 4.

Second pass:

- Google best practices.
- Regularizaton Chapter.
- Optimization Chapter.
- Pratical Methodology Chapter.

AutoEncoders

First look (in order):

- Here you find a first read.
- This is your second recommended read.
- This is a lecture from Andrew NG.
- I also give you some examples: 1, 2, 3, 4.

Second pass: AutoEncoders Chapter.

Tips & Best practices: 1, 2, 3, 4, 5.

Reinforcement Learning

First look (in order):

- Here you have an explanation video.
- This article explains RL really well.
- Here is an interesting read.
- Some examples: 1, 2, 3, 4.

Second pass: The go-to guide. Paper with state-of-the-art RL architecture. Complete free book on RL.

Tips & Best practices: 1, 2.

Utilities

Hey You. During the last few years I collected tons of articles, web apps, reddit threads, best practices, projects and repositories, and I want to share with you each single bit of information, organizing them by type of resource (blogs or projects ideas, and so on).

Machine Learning Projects

- Enormous and awesome collection of Data Science Cheat Sheets
- Infinite collection of actual Data Science / ML projects
- Infinite collection of tutorials and ML projects in TensorFlow
- Other TensorFlow examples

Tools

- Google Data Visualization Facets
- Interactive Neural Network

Youtube Channels

- Enthought
- Siraj Raval
- 3Blue1Brown
- Microsoft
- TensorFlow Official channel
- Engineering Man
- The Tech Lead

Blogs

- How to build a data science portfolio
- Distill blog
- Keras
- Paolo Galeone blog
- TensorFlow official blog
- KD Nuggets
- Incredible Graphic explanations

Websites worth taking a look!

- The best machine learning **short** book i've ever read
- A monster collection of Data related free course
- Machine Learning Map
- Data modeling for Business Intelligence
- Statistics explained
- Visualizing data - Turorials
- Fast.ai
- Open.ai
- Explained.ai
- Machine Learning Glossary
- Python ML Tutorials
- For Italian Learners!
- Immersive math
- DeepLizard
- Common Statistical Fallacies

- Scikit-Learn pratical recipes

Subreddits you want to follow!

- 10 awesomee subreddits related to data sciece
- An incredible tool to discover trends and subs

Next Steps Roadmap

Thanks to the great success of this guide, i've decided to expand it a lot and make more similar for different topics. Some of the extensions i'm adding here during the next weeks are:

- Unsupervised Learning : **done**
- Machine Learning mindset framework (how to think like a data scientist)
- Data processing and preparation with Pandas
- Feature Selection
- Features Engineering
- Extending the parameters optimization section
- Keras Library
- TensorFlow 2.0
- How to deploy my model on AWS
- How to deploy my model on Azure

Later on, i'll do an entire guide on Cloud Computing with AWS and Azure!

... Coming Very Soon! Stay tuned :)