The Best Machine Learning Libraries in Python

By Scott Robinson • November 10, 2015 • 4 Comments

Introduction

There is no doubt that neural networks, and machine learning in general, has been one of the hottest topics in tech the past few years or so. It's easy to see why with all of the really interesting use-cases they solve, like voice recognition, image recognition, or even music composition. So, for this article I decided to compile a list of some of the best Python machine learning libraries and posted them below.

In my opinion, Python is one of the best languages you can use to learn (and implement) machine learning techniques for a few reasons:

- **It's simple**: Python is now becoming the language of choice among new programmers thanks to its simple syntax and huge community
- **It's powerful**: Just because something is simple doesn't mean it isn't capable. Python is also one of the most popular languages among data scientists and web programmers. Its community has created libraries to do just about anything you want, including machine learning
- **Lots of ML libraries**: There are tons of machine learning libraries already written for Python. You can choose one of the hundreds of libraries based on your use-case, skill, and need for customization.

The last point here is arguably the most important. The algorithms that power machine learning are pretty complex and include a lot of math, so writing them yourself (and getting it right) would be the most difficult task. Lucky for us, there are plenty of smart and dedicated people out there that have done this hard work for us so we can focus on the application at hand.

By no means is this an exhaustive list. There is lots of code out there and I'm only posting some of the more relevant or well-known libraries here. Now, on to the list.
The Most Popular Libraries

I've included a short description of some of the more popular libraries and what they're good for, with a more complete list of notable projects in the next section.

Tensorflow

This is the newest neural network library on the list. Just having been released in the past few days, Tensorflow is a high-level neural network library that helps you program your network architectures while avoiding the low-level details. The focus is more on allowing you to express your computation as a data flow graph, which is much more suited to solving complex problems.

It is mostly written in C++, which includes the Python bindings, so you don't have to worry about sacrificing performance. One of my favorite features is the flexible architecture, which allows you to deploy it to one or more CPUs or GPUs in a desktop, server, or mobile device all with the same API. Not many, if any, libraries can make that claim.

It was developed for the Google Brain project and is now used by hundreds of engineers throughout the company, so there's no question whether it's capable of creating interesting solutions.

Like any library though, you'll probably have to dedicate some time to learn its API, but the time spent should be well worth it. I spent only a few minutes playing around with the core features and could already tell Tensorflow would allow me to spend more time implementing my network designs and not fighting through the API.

- Good for: Neural networks
- Website
- Github

scikit-learn

The scikit-learn library is definitely one of, if not the most, popular ML libraries out there among all languages. It has a huge number of features for data mining and data analysis, making it a top choice for researches and developers alike.
Its built on top of the popular NumPy, SciPy, and matplotlib libraries, so it'll have a familiar feel to it for the many people that already use these libraries. Although, compared to many of the other libraries listed below, this one is a bit more lower level and tends to act as the foundation for many other ML implementations.

- Good for: Pretty much everything
- Website
- Github

Theano

Theano is a machine learning library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays, which can be a point of frustration for some developers in other libraries. Like scikit-learn, Theano also tightly integrates with NumPy. The transparent use of the GPU makes Theano fast and painless to set up, which is pretty crucial for those just starting out. Although some have described it as more of a research tool than production use, so use it accordingly.

One of its best features is great documentation and tons of tutorials. Thanks to the library's popularity you won't have much trouble finding resources to show you how to get your models up and running.

- Good for: Neural networks and deep learning
- Website
- Github

Pylearn2

Most of Pylearn2's functionality is actually built on top of Theano, so it has a pretty solid base.

According to Pylearn2's website:

> Pylearn2 differs from scikit-learn in that Pylearn2 aims to provide great flexibility and make it possible for a researcher to do almost anything, while scikit-learn aims to work as a “black box” that can produce good results even if the user does not understand the implementation.
Keep in mind that Pylearn2 may sometimes wrap other libraries such as scikit-learn when it makes sense to do so, so you're not getting 100% custom-written code here. This is great, however, since most of the bugs have already been worked out. Wrappers like Pylearn2 have a very important place in this list.

- Good for: Neural networks
- Website
- Github

Pyevolve

One of the more exciting and different areas of neural network research is in the space of genetic algorithms. A genetic algorithm is basically just a search heuristic that mimics the process of natural selection. It essentially tests a neural network on some data and gets feedback on the network's performance from a fitness function. Then it iteratively makes small, random changes to the network and proceeds to test it again using the same data. Networks with higher fitness scores win out and are then used as the parent to new generations.

Pyevolve provides a great framework to build and execute this kind of algorithm. Although the author has stated that as of v0.6 the framework is also supporting genetic programming, so in the near future the framework will lean more towards being an Evolutionary Computation framework than a just simple GA framework.

- Good for: Neural networks with genetic algorithms
- Github

NuPIC

NuPIC is another library that provides to you some different functionality than just your standard ML algorithms. It is based on a theory of the neocortex called Hierarchical Temporal Memory (HTM). HTMs can be viewed as a type of neural network, but some of the theory is a bit different.

Fundamentally, HTMs are a hierarchical, time-based memory system that can be trained on various data. It is meant to be a new computational framework that mimics how memory and computation are intertwined within our brains. For a full explanation of the theory and its applications, check out the whitepaper.

- Good for: HTMs
Pattern

This is more of a 'full suite' library as it provides not only some ML algorithms but also tools to help you collect and analyze data. The data mining portion helps you collect data from web services like Google, Twitter, and Wikipedia. It also has a web crawler and HTML DOM parser. The nice thing about including these tools is how easy it makes it to both collect and train on data in the same program.

Here is a great example from the documentation that uses a bunch of tweets to train a classifier on whether a tweet is a 'win' or 'fail':

```python
from pattern.web import Twitter
from pattern.en import tag
from pattern.vector import KNN, count

twitter, knn = Twitter(), KNN()

for i in range(1, 3):
    for tweet in twitter.search('#win OR #fail', start=i, count=100):
        s = tweet.text.lower()
        p = '#win' in s and 'WIN' or 'FAIL'
        v = tag(s)
        v = [word for word, pos in v if pos == 'JJ'] # JJ = adjective
        v = count(v) # {'sweet': 1}
        if v:
            knn.train(v, type=p)

print knn.classify('sweet potato burger')
print knn.classify('stupid autocorrect')
```

The tweets are first collected using `twitter.search()` via the hashtags '#win' and '#fail'. Then a k-nearest neighbor (KNN) is trained using adjectives extracted from the tweets. After enough training, you have a classifier. Not bad for only 15 lines of code.

- Good for: NLP, clustering, and classification
- Github
Caffe

Caffe is a library for machine learning in vision applications. You might use it to create deep neural networks that recognize objects in images or even to recognize a visual style.

Seamless integration with GPU training is offered, which is highly recommended for when you're training on images. Although this library seems to be mostly for academics and research, it should have plenty of uses for training models for production use as well.

- Good for: Neural networks/deep learning for vision
- Website
- Github

Other Notable Libraries

And here is a list of quite a few other Python ML libraries out there. Some of them provide the same functionality as those above, and others have more narrow targets or are more meant to be used as learning tools.

Nilearn

- Built on top of scikit-learn
- Github

Statsmodels

- Github

PyBrain (inactive)

- Github

Fuel

- Github

Bob
- skdata
  - Github

MILK
- Github

IEPY
- Github

Quepy
- Github

Hebel
- Github

mlxtend
- Github

nolearn
- Github

Ramp
- Github

Feature Forge
- Github

REP
- Github
Python-ELM
  • Github

PythonXY
  • Website

XCS
  • Website

PyML
  • SourceForge

MLPY (inactive)
  • Website

Orange
  • Website

Monte
  • Github

PYMVPA
  • Website

MDP (inactive)
  • Github

Shogun
  • Website

PyMC
• Website

Gensim
• Website

Neurolab
• Github

FFnet (inactive)
• SourceForge

LibSVM
• Github

Spearmint
• Github

Chainer
• Github

topik
• Github

Crab
• Github

CoverTree
• Github

breze
• Based on Theano
• Github
deap
  • Github
neurolab
  • Github
Spearmint
  • Github
yahmm
  • Github
pydeep
  • Github
Annoy
  • Github
neon
  • Github
sentiment
  • Github

* artificial intelligence, python