AI Tools & Projects

Contact: Stefan Pinkernell Giorgio Busatto

Machine learning (ML) methods allow to use a computer system for perform a task by means of a model that has been trained with example data from the domain of interest.

The goal of our work is to identify ML methods and applications that are relevant for the scientific work carried out at the AWI, and to understand how they can be implemented using available software libraries within the AWI (storage and computing) infrastructure.

Methods and applications

Typical ML applications include:

- **Classification**: mapping of data samples to a predefined finite set of labels. In this case, we want a model to automatically suggest a classification for an input data sample. The model is first trained and validated on existing, already classified sample data.
- **Regression**: modelling the relation (function) between independent variables (features) and a dependent variable. A model is used to predict a function for which we have sampled a number of point. A typical application is to forecast future values (e.g. houses prices in a certain area) from past data.
- **Clustering**: grouping elements of a dataset into clusters so that elements in the same group are similar to each other according to some criteria. This is useful to discover similarity between data samples when the user cannot or does not want to classify the data themself.
- **Dimensionality reduction**: reducing the number of variables (features) of a model. This can be used as a preliminary step for other applications, e.g. one can perform classification using a smaller number of features after applying dimensionality reduction.

The following table shows a few popular ML methods (or method families) that can be used for the above-mentioned applications:

	Classification	Regression	Clustering	Dimensionality reduction
Linear models	X	X		
Random forests	X	X		
Principal component analysis (PCA)				X
Neural networks and deep learning	X	X		
k-Means			X	

The term **deep learning** refers to neural networks using larger and more complex architectures than traditional ones. Deep learning has been successfully applied in areas like image and sound classification.

It is outside the scope of this document to describe these methods in detail. In the following sections we describe the ML technologies that we have considered so far, and concrete applications / case studies.

Supported Technologies

Nowadays there is a wide choice of technologies for machine learning and therefore we restrict our attention to a subset that we consider relevant for users working at the AWI. Our supported technologies are chosen according to the following criteria:

- A technology should be wide spread and have an active community around it.
- Ideally, a technology should be based on Python or, alternatively, on R or Java. The choice of the programming language is based on its popularity in the data science community and among scientists.
- Scalability: it should be possible to process data in parallel on a cluster if the ML model or the data become too large.
- Multiplatform: in a typical scenario a user will develop a model on their own laptop and later they would like to training or run their model on resources of the AWI, e.g. on the scientific cluster or on HPC-resources.

Here is an overview of the frameworks and libraries that we have considered:

- Python data science stack: numpy (https://numpy.org/), pandas (https://pandas.pydata.org/), scipy (https://www.scipy.org/)
  These are standard tools which most data scientists are familiar with.

Contact

Questions, interested in using our AI infrastructure and tools?: o2a-support (at) awi.de

- Scikit-Learn (https://scikit-learn.org): Python toolkit for machine learning, very wide spread with a large and active and good documentation.
- TensorFlow (TF, https://www.tensorflow.org): framework for machine learning with neural networks and for deep learning
- pyTorch (https://pytorch.org): a good alternative to TensorFlow even though not as popular.
- Keras (https://keras.io): unified API for different ML back ends such as Tensorflow, Theano and CNTK
- Dask (https://dask.org): general-purpose framework for running Python programs in a parallel distributed fashion.
- Spark (https://spark.apache.org): general-purpose analytics engine with support for the Hadoop filesystem and cluster computing. It has a rich choice of machine learning libraries (e.g. MLlib). It is appropriate for processing very large, possibly distributed data sets.
- Elephas (http://maxpumperla.com/elephas): a Keras extension for training and executing Keras ML models on a Spark cluster.

According to their popularity and relevant for potential users at AWI, we are currently focused mainly on the following frameworks:

- Maschine learning in general: Scikit-Learn
- Deep learning: Tensorflow, possibly in combination with Keras

We are evaluating other frameworks like Dask and Spark for applications that involve large models and / or data sets, and therefore require more powerful computing resources.

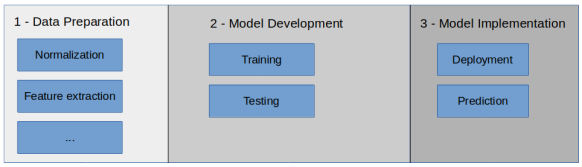
Generic Workflow for Data Analysis and Machine Learning

Drawing from our experience with the processing of LOKI-images and whale acoustic data (see the Projects Section below), we are developing a generic workflow for data classification, which is integrated in the AWI computing infrastructure.

We can identify three phases within data analysis based on ML-methods:

1. Data preparation: the raw data is converted to some standard format, cleaned and normalized. The preprocessed data is used to extract features to be used for machine learning.
2. Model development: different ML models are trained and tested with the extracted features. At the end of this phase, a model is chosen for use in production.
3. Implementation: the trained model is used to classify real data for a concrete application.

These phases are summarized in the following diagram:



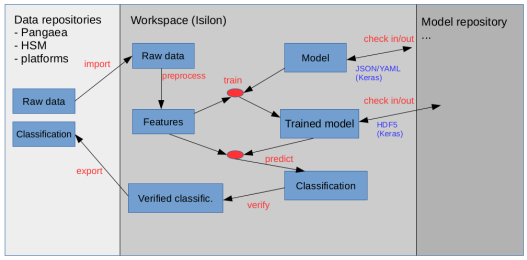
An important aspect when working with machine learning is the data flow, i.e.

1. Which data are processed and where they are stored.
2. How the data is transferred between the different storage systems.
3. How the data is processed, i.e. which implementation of which algorithms is run on which compute resources.

A typical data flow for a ML classification within the AWI infrastructure can be sketched as follows (see also the diagram):

- The raw data are stored in an archiving system, e.g. Pangaea, HSM, platforms.
- The raw data must be initially imported into a mass storage where they can be processed. This is normally a shared project folder on the Isilon system (workspace).
- The raw data are preprocessed and use to produce features. The result is again saved to the workspace.
- ML models (e.g. Keras models) are also stored in the workspace. They can be archived in a model repository and re-use at a later point in time. Realization of model management is still an open task.
- In the subsequent steps, a model is trained, verified, and finally used for data classification. The resulting classified data can be exported again to some data archive.

ML classification workflow



This workflow is described in more detail in a separate page.

Projects

Evaluation of the Dask Framework

Python and Pandas are often used for data analytics tasks. A typical scenario is that a user starts working on a small dataset on their PC and then want to move on to larger datasets when their code is working. If the dataset size exceeds the capability of a personal computer, it is necessary to move to more powerful computing resources, possibly using parallel computation on a computer cluster. Dask is a framework that provides an API very similar to Pandas' and allows to work on large datasets on a cluster.

Since both Python and Pandas are wide-spread among data scientists, we have evaluated Dask on a few example data sets of different sizes to see how well a Dask application scales. The results of this evaluation are:

1. A comparison of performance and memory consumption between Pandas and Dask on the same benchmark data sets.
2. A short HOWTO documentation on how to set up and use Dask.

These results are documented in a separate page.

Classification of Zooplankton images (LOKI)

In cooperation with workgroup Niehoff

Tools / methods: Machine learning and deep learning (convolutional neural networks), Spark, Elephas.

Deep learning approach

Stella Mahler, internship and bachelor thesis (2019): “Automatic Classification of Zooplankton Images through Convolutional Neural Networks”

Goal: Comparison of the InceptionV3 and the Resnet101 architectures, as pre-trained and self-trained models.

Image data and taxonomy information were obtained from the Ecotaxa system and provided by Nicole Hildebrandt.

The data have a strong bias in relation to class size. Therefore, classes with lots of images were reduced by means of sub-sampling and classes with few images were extended by means of augmentation. Very small classes were completely discarded or merged together into new classes at a higher taxonomy level. The preprocessed data set consisted of 11 classes, each containing 1000 images. Furthermore, all images were normalized with respect to size, brightness, and so on.

Partitioning of the data: Training set 60%, test and validation set 20% each.

Results:

- Pre-trained models have poor results (accuracy < 0.1) probably due to the fact that the models were trained with image from a completely different domain and did not contain relevant features.
- Self-trained models deliver significantly better results (ResNet 0.6; InceptionV3 0.8).

The details of this projects are described in a separate page.

Machine learning approach using image descriptors

Classification of plankton images by means of machine learning methods based on image descriptors.

Comparison of different machine learning frameworks: pyspark, Scikit-Learn, Tensorflow. In particular, evaluation of Spark for ML:

1. Is it possible to use the same ML models with Scikit-learn and Spark?
2. Do these models scale with large data sets?

Original data set: about 120000 samples, 25 classes.

Reduced data set (with the purpose of reducing bias): 7000 samples, 7 classes (1000 samples per class)

Preliminary results of image classification with descriptors show an accuracy comparable to that obtained by CNNs working directly on the image data. These result must be further evaluated.

pySpark implementation: long training times (several hours on a virtual machine with 16 cores, 32GB RAM). This could be due to some configuration error and shows that more effort is needed to tune a pySpark model / installation.

Scikit-Learn implementation: training times in the order of magnitude of a few minutes.

Tensorflow: offers other methods and is therefore difficult to compare with the previous two wrt performance.

Open tasks: code consolidation, further evaluation of results, Spark environment configuration.

The details of this project can be found in a separate page.

Classification of whale calls (acoustic data)

Cooperation with workgroup O. Böbel

Machine learning on Cray (Ollie)

Tests with Urika-CS AI and analytics applications.

Dockerized ML Workflows

A recurrent issue in machine learning is that of reproducibility: Once a model has been developed and tested, it should be possible to run the same model with the same data on another system with little or no effort at all. There are two main scenarios in which this is important:

- The developer of the model may want to run the model on a different computer, possibly with more computing power.
- The developer would like to make their model to accessible to others.

In both cases, the model, its runtime environment (libraries, frameworks, and son on), and possibly some example test data should be easy to install and get to work.

For these reasons, we started to experiment with Docker. The goal is to package a machine learning model together with its example data sets and required libraries in a docker container that can be imported and run out of the box.