Multi-target Compiler for the Deployment of Machine Learning Models

Oscar J. Castro-Lopez

July 11, 2023

University of Luxembourg

Content

Introduction

Introduction: Building Machine Learning Models

Building Machine Learning Models

- There is a great interest in building applications infused with Machine Learning (ML) models.

Building Machine Learning Models

- There is a great interest in building applications infused with Machine Learning (ML) models.
- Modeling is done by data scientists:
  - A role closely related to: *math*, *statistics*, *domain knowledge of the data*.

Introduction: Building Machine Learning Models

Building Machine Learning Models

- There is a great interest in building applications infused with Machine Learning (ML) models.
- Modeling is done by data scientists:
  - A role closely related to: *math, statistics, domain knowledge of the data*.
  - Use tools/languages for a rapid model building/prototyping (R, Python (libraries), Weka, Knime, SAS, SPSS, etc.)

Introduction: Building Machine Learning Models

Building Machine Learning Models

- There is a great interest in building applications infused with Machine Learning (ML) models.
- Modeling is done by data scientists:
    - A role closely related to: *math, statistics, domain knowledge of the data*.
    - Use tools/languages for a rapid model building/prototyping (R, Python (libraries), Weka, Knime, SAS, SPSS, etc.)
    - Their main objective is to create the best possible model.

Introduction: Predictive Modeling Process

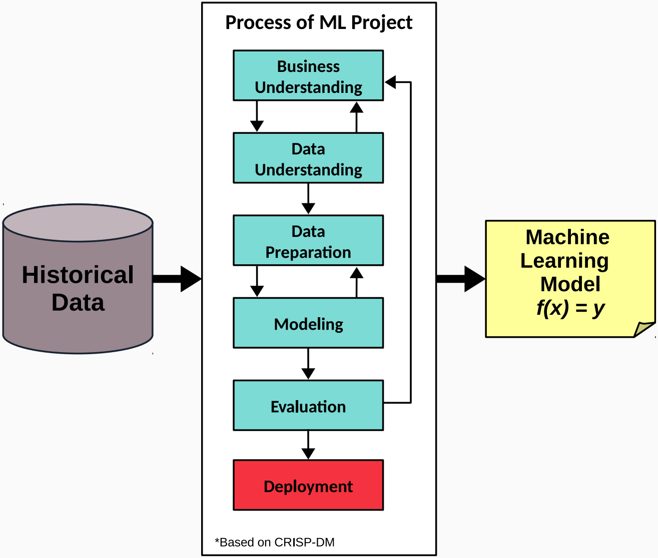


Figure 1: CRISP-DM Modeling Process

Introduction: Machine Learning Models in Production

ML Deployment

- Once a model has been built, it must be deployed to a production environment (software/application is put into operation for its intended use).

ML Deployment

- Once a model has been built, it must be deployed to a production environment (software/application is put into operation for its intended use).
- In production it is where the model generates value through the predictions made on incoming data.

ML Deployment

- Once a model has been built, it must be deployed to a production environment (software/application is put into operation for its intended use).
- In production it is where the model generates value through the predictions made on incoming data.
- The process of generating a prediction is also referred as *scoring* or *inference* (depending on the domain).

ML Deployment

Deployment
General process of taking a model (math function) to a specific operating environment where it is available for its use (software).

ML Deployment

Deployment
General process of taking a model (math function) to a specific operating environment where it is available for its use (software).

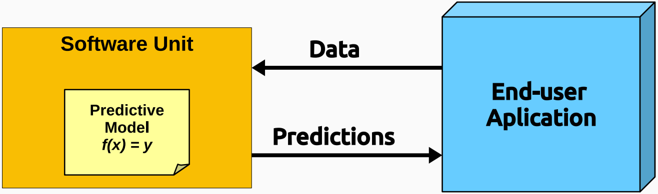


Figure 2: Predictive Model as a Software Unit

Introduction: Machine Learning Models in Production

Building Applications Integrating ML Models

- Deployment of ML models is a challenging task.

Building Applications Integrating ML Models

- Deployment of ML models is a challenging task.
- Software in production is developed and maintained by Software Engineers who:
    - Are experts in software building tools: IDE's, SDK's, Frameworks, etc.

Building Applications Integrating ML Models

- Deployment of ML models is a challenging task.
- Software in production is developed and maintained by Software Engineers who:
  - Are experts in software building tools: IDE's, SDK's, Frameworks, etc.
  - Aim to build software following requirements and quality attributes.

Introduction: Deployment issues and approaches

Model building vs Model deployment

- Both processes are done in different languages / environments

[1] Gartner 2019: Magic quadrant for data science and machine learning platforms and Gartner 2022: AI in Organizations

Introduction: Deployment issues and approaches

Model building vs Model deployment

- Both processes are done in different languages / environments
- ML models integration must comply with Software design / architecture

[1] Gartner 2019: Magic quadrant for data science and machine learning platforms and Gartner 2022: AI in Organizations

Model building vs Model deployment

- Both processes are done in different languages / environments
- ML models integration must comply with Software design / architecture
- 60%(2019) and 53%(2022) of ML models are actually never deployed to production [1]

[1] Gartner 2019: Magic quadrant for data science and machine learning platforms and Gartner 2022: AI in Organizations

Common deployment approaches are:

- **Pipeline**, connect software in production with modeling tools.

Common deployment approaches are:

- **Pipeline**, connect software in production with modeling tools.
- **In-database**, models deployed inside a DBMS invoked by SQL query (Examples: Teradata, Oracle).

Introduction: Deployment issues and approaches

Common deployment approaches are:

- **Pipeline**, connect software in production with modeling tools.
- **In-database**, models deployed inside a DBMS invoked by SQL query (Examples: Teradata, Oracle).
- **Manual coding**, manually translate the model to a programming language.

Common deployment approaches are:

- **Pipeline**, connect software in production with modeling tools.
- **In-database**, models deployed inside a DBMS invoked by SQL query (Examples: Teradata, Oracle).
- **Manual coding**, manually translate the model to a programming language.
- **IoT/Edge-Computing**, one-to-one compilation (commonly Deep learning models).

- Using a pipeline follows a client-server approach, it can become a bottleneck for the whole system.

Introduction: Drawbacks of current approaches

- Using a pipeline follows a client-server approach, it can become a bottleneck for the whole system.
- Scaling a pipeline approach can become a pipeline jungle, hard to maintain over time [Sculley et al., 2015].

Introduction: Drawbacks of current approaches

- Using a pipeline follows a client-server approach, it can become a bottleneck for the whole system.
- Scaling a pipeline approach can become a pipeline jungle, hard to maintain over time [Sculley et al., 2015].
- In-database and IoT-Edge are limited by specific cases.

Introduction: Drawbacks of current approaches

- Using a pipeline follows a client-server approach, it can become a bottleneck for the whole system.
- Scaling a pipeline approach can become a pipeline jungle, hard to maintain over time [Sculley et al., 2015].
- In-database and IoT-Edge are limited by specific cases.
- Manual coding is a labor-intensive task, prone to errors.

Introduction: Formal description of proposed deployment

Deployment task:

Introduction: Formal description of proposed deployment

Deployment task:

Given a predictive model A, which is a computer-generated mathematical function expressed in language L, we want to generate A' expressed in a computer programming language L', where A' is semantically equivalent to A.

Introduction: Formal description of proposed deployment

Deployment task:
Given a predictive model A, which is a computer-generated mathematical function expressed in language L, we want to generate A' expressed in a computer programming language L', where A' is semantically equivalent to A.

Given that ML models can be formally defined, and that we use modeling tools to built them, we can algorithmically transform it by using a compiler.

Introduction: Formal description of proposed deployment

Deployment task:
Given a predictive model A, which is a computer-generated mathematical function expressed in language L, we want to generate A' expressed in a computer programming language L', where A' is semantically equivalent to A.

Given that ML models can be formally defined, and that we use modeling tools to built them, we can algorithmically transform it by using a compiler.

Declarative $\rightarrow$ Procedural

Compiler

Compiler

We developed a multi-target compiler to translate Machine Learning models into source code to automate the deployment to production environments

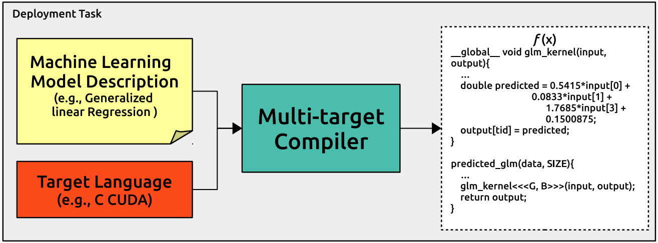


Figure 3: Multi-target compiler that translates ML models to source code

We can effectively automate the deployment task with a compiler

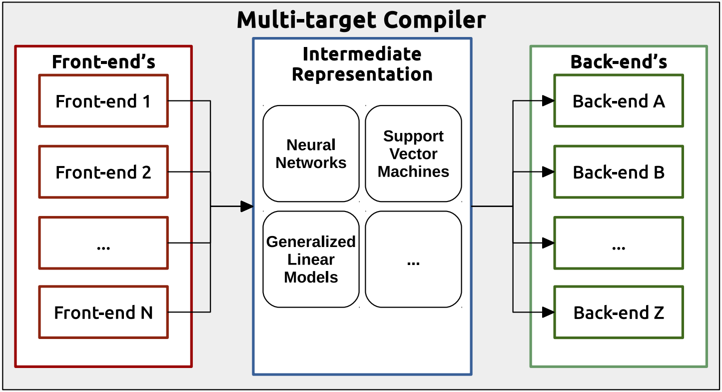


Figure 4: General design of the proposed multi-target compiler.

Intermediate Representation Template pt. 1

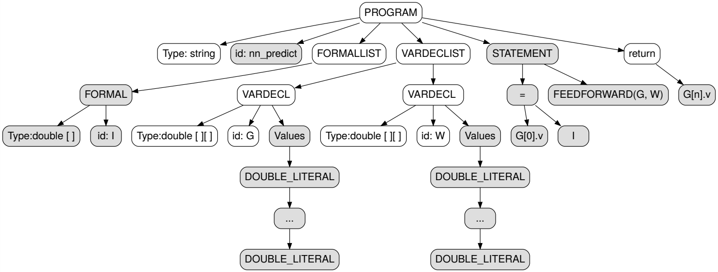


Figure 5: Example of a IR template for a neural network.

Intermediate Representation Template pt. 2

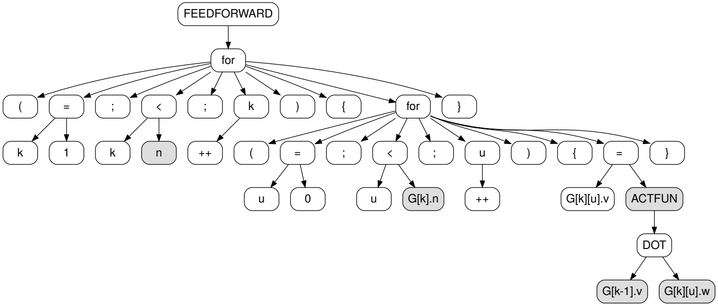


Figure 6: Example of feed forward computation of a neural network.

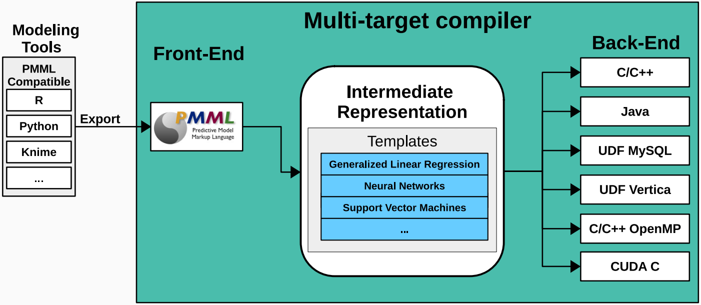


Figure 7: Implementation of the multi-target compiler.

Empirical evaluation

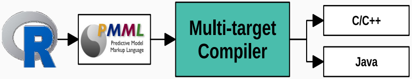
Validation and Evaluation

- The first validation is the correctness of the generated code. Predictions must be equal in the original model creation tool and generated code.

Validation and Evaluation

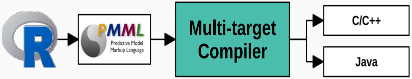
- The first validation is the correctness of the generated code. Predictions must be equal in the original model creation tool and generated code.
- Evaluation, testing the efficiency of the execution of the predictive models by using the generated code by the compiler.

Sequential experiments GLM models

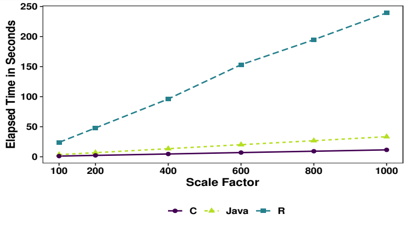


Workflow

Sequential experiments GLM models

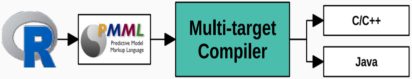


Workflow

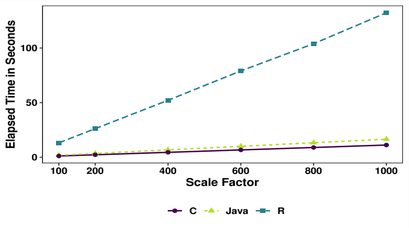


DCCC

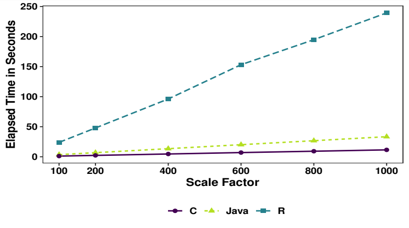
Sequential experiments GLM models



Workflow

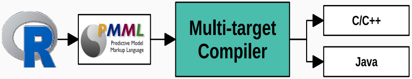


EGSS

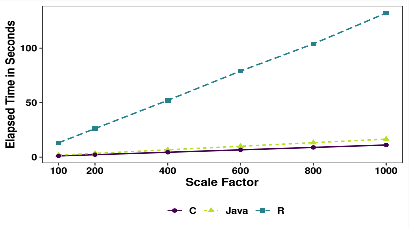


DCCC

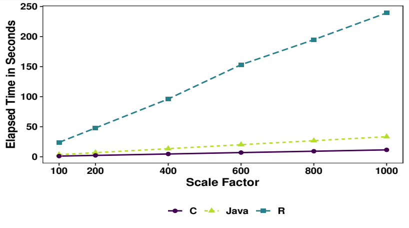
Sequential experiments GLM models



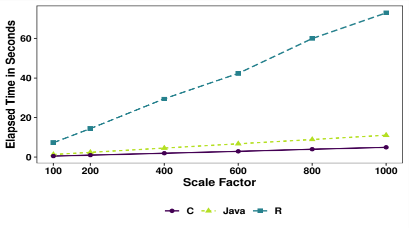
Workflow



EGSS



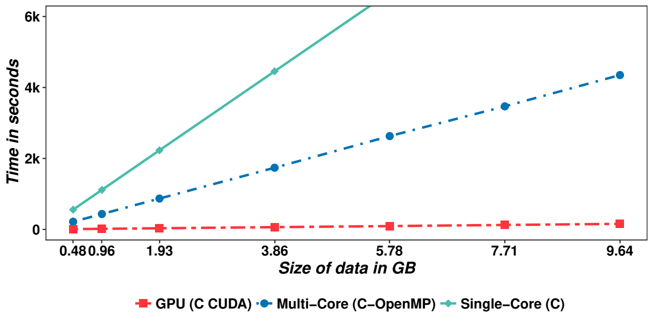
DCCC



OSPI

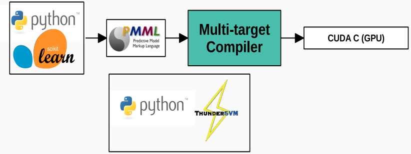
Sequential and parallel experiments with a SVM binary class model

Workflow



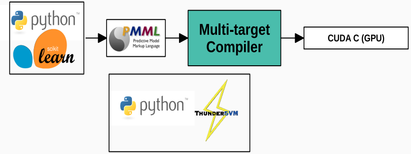
DCCC dataset

GPU Experiments with SVM multi-class model

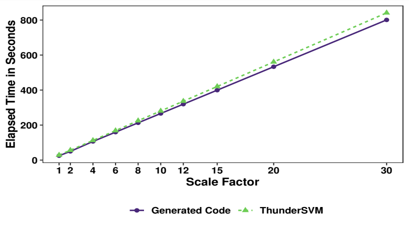


Workflow

GPU Experiments with SVM multi-class model

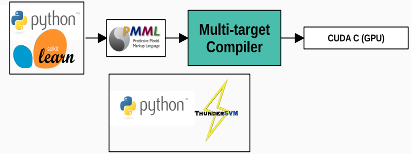


Workflow

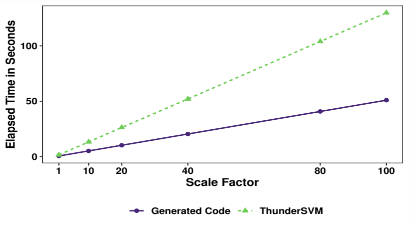


MNIST

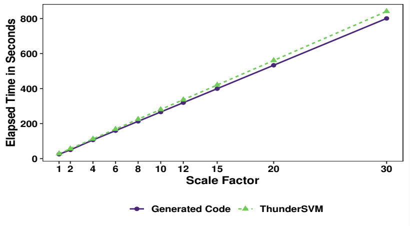
GPU Experiments with SVM multi-class model



Workflow

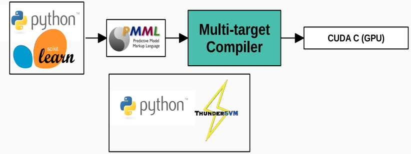


Poker

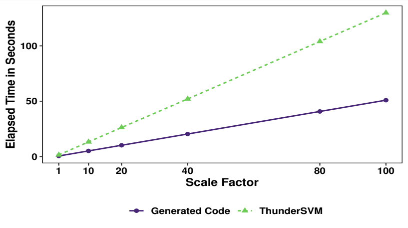


MNIST

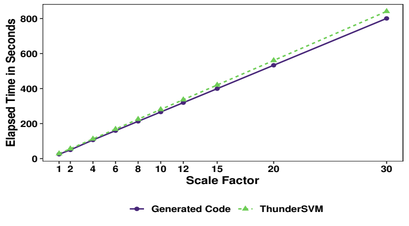
GPU Experiments with SVM multi-class model



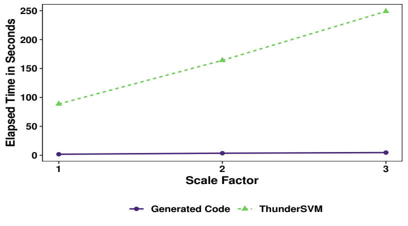
Workflow



Poker



MNIST



RCV1

Conclusions

Conclusions

- We presented a compiler to translate ML models to source code.

Conclusions

- We presented a compiler to translate ML models to source code.
- The code can be embedded inside operational environments (i.e. automate deployment) and is self-contained.

Conclusions

- We presented a compiler to translate ML models to source code.
- The code can be embedded inside operational environments (i.e. automate deployment) and is self-contained.
- We can effectively reduce the time-to-deploy.

Conclusions

- We presented a compiler to translate ML models to source code.
- The code can be embedded inside operational environments (i.e. automate deployment) and is self-contained.
- We can effectively reduce the time-to-deploy.
- We can leverage sequential and parallel architectures for efficient scoring in production environments.

Conclusions

- We presented a compiler to translate ML models to source code.
- The code can be embedded inside operational environments (i.e. automate deployment) and is self-contained.
- We can effectively reduce the time-to-deploy.
- We can leverage sequential and parallel architectures for efficient scoring in production environments.
- The modularity in our compiler allows for an easy extension of both new types of models and new target languages.

References

References i

Lopez-Rojas, E., Elmir, A., and Axelsson, S. (2016).
PaySim: A financial mobile money simulator for fraud detection.
In *Proceedings of the European Modeling and Simulation Symposium*, pages 249–255, Larnaca, Cyprus. Dime University of Genoa.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015).
Hidden technical debt in machine learning systems.
In *Advances in Neural Information Processing Systems*, pages 2503–2511.

Yeh, I.-C. and hui Lien, C. (2009).
The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients.
Expert Systems with Applications, 36(2, Part 1):2473 – 2480.

Current work/interests:

- **Speeding up programs.**
  - Special focus on Python.
  - Improving performance of DS/ML pipelines.
  - Code compilation/optimization.
  - Developing libraries: HPC, CUDA, Big Data.
- **Applied DS/ML.**

Thank you!!

Any questions?

Email: `oscar.castro@uni.lu`

Example of PMML

```xml
<PMML version="4.2">
 <Header copyright="Copyright (c) 2016 lcid" description="Neural Network PMML Model">
  <Extension name="user" value="lcid" extender="Rattle/PMML"/>
  <Application name="Rattle/PMML" version="1.4"/>
  <Timestamp>2016-12-09 18:56:44</Timestamp>
 </Header>
 <DataDictionary numberOfFields="9">
  <DataField name="MITBEAT_NN_FFT" optype="categorical" dataType="string">
   <Value value="A"/>
   <Value value="N"/>
  </DataField>
  <DataField name="V1" optype="continuous" dataType="double"/>
...
 </DataDictionary>
 <NeuralNetwork modelName="NeuralNet_model" functionName="classification" numberOfLayers="3"
      activationFunction="logistic">
...
   <OutputField name="Predicted_MITBEAT_NN_FFT" feature="predictedValue"/>
   <OutputField name="Probability_A" optype="continuous" dataType="double" feature="probability"
          value="A"/>
   <OutputField name="Probability_N" optype="continuous" dataType="double" feature="probability"
          value="N"/>
  </Output>
  <NeuralInputs numberOfInputs="8">
...
  </NeuralInputs>
  <NeuralLayer numberOfNeurons="28">
   <Neuron id="9" bias="0.83109384784387">
    <Con from="1" weight="-1.58888618835219"/>
...
   </Neuron>
...
   </Neuron>
  </NeuralLayer>
  <NeuralLayer numberOfNeurons="2" activationFunction="threshold" threshold="0.5">
   <Neuron id="38" bias="1.0">
    <Con from="37" weight="-1.0"/>
   </Neuron>
...
  </NeuralLayer>
  <NeuralOutputs numberOfOutputs="2">
   <NeuralOutput outputNeuron="38">
    <DerivedField name="derivedNO_MITBEAT_NN_FFT" optype="continuous" dataType="double">
     <NormDiscrete field="MITBEAT_NN_FFT" value="A"/>
    </DerivedField>
   </NeuralOutput>
   <NeuralOutput outputNeuron="39">
    <DerivedField name="derivedNO_MITBEAT_NN_FFT" optype="continuous" dataType="double">
     <NormDiscrete field="MITBEAT_NN_FFT" value="N"/>
    </DerivedField>
   </NeuralOutput>
  </NeuralOutputs>
 </NeuralNetwork>
</PMML>
```

Example of generated C code

```c
double *predicted_mitbeat_nn_fft(double in_v1, double in_v2, double in_v3, double in_v4, double
       in_v5, double in_v6, double in_v7, double in_v8){
        //Variable transformation
        double new_v1 = in_v1;
        ...
        // layer
        static double layer0[28];
        layer0[0] = 1*0.83109384784387+new_v1*-1.58888618835219+new_v2*-1.82672308266525+new_v3
              *3.33746196373367+new_v4*-0.386660430766735+new_v5*2.37454410956807+new_v6
              *-1.30898075446596+new_v7*-1.43872713454692+new_v8*2.4529341743543;
        layer0[0] = 1 / (1+exp(-layer0[0]));
        ...
        // layer
        static double layer1[1];
        layer1[0] = 1*-4.62592500800819+layer0[0]*-1.03197631554537+layer0[1]*6.43276062612687+
              layer0[2]*3.06106357914918+layer0[3]*4.25114324158677+layer0[4]*-1.81616145986666+
              layer0[5]*-2.64384641103975+layer0[6]*-1.5951340774553+layer0[7]*1.27863903623786+
              layer0[8]*0.707355744671864+layer0[9]*-1.76172788928607+layer0
              [10]*0.53212821423806+layer0[11]*-1.96944871187955+layer0[12]*-1.7584593904505 6+
              layer0[13]*-1.08114467050309+layer0[14]*1.75717862584521+layer0
              [15]*-0.364837336276863+layer0[16]*0.991809219615911+layer0
              [17]*-0.194322645909799+layer0[18]*-1.16222444790653+layer0[19]*-2.48554250501996+
              layer0[20]*-0.225800791983646+layer0[21]*-0.957736167039977+layer0
              [22]*-2.89636184187767+layer0[23]*5.69469441754116+layer0[24]*0.730216654347973+
              layer0[25]*-2.22926009625776+layer0[26]*-3.14190898011756+layer0
              [27]*2.20678305117676;
        layer1[0] = 1 / (1+exp(-layer1[0]));

        // layer
        static double layer2[2];
        layer2[0] = 1*1.0+layer1[0]*-1.0;
        layer2[0] = (layer2[0]>0.5)? 1.0 : 0.0;
        layer2[1] = 1*0.0+layer1[0]*1.0;
        layer2[1] = (layer2[1]>0.5)? 1.0 : 0.0;

        return layer2;
}
char const * predicted_mitbeat_nn_fft_response(double probabilities[]){
        char const *  labels[] = {"A", "N"};
        int max = 0;
        int i;
        for (i = 1; i < 2; i++) {
                if(probabilities[i]>probabilities[max])
                        max =i;
        }
        return labels[max];
}
```

Datasets

- DCCC. Default of Credit Card (Kaggle).
- EGSS. Electrical Grid Stability Simulated (UCI).
- OSGI. Online Shoppers' Purchasing Intention (UCI).
- MNIST. It is a database of images of handwritten digits (LIBSVM Data).
- Poker. This is the poker hand dataset (LIBSVM Data).
- RCV1. This is the Reuters Corpus Volume 1 dataset (LIBSVM Data).

- Kaggle: `https://www.kaggle.com/datasets`
- UCI: `https://archive.ics.uci.edu/ml/datasets.php`
- LIBSVM: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

Computer equipment

- Workstation Intel Xeon W-2133 CPU 6 cores 3.60 GHz, 64 GB RAM.
- GPU GeForce GTX 1080 8GB RAM, 2,560 CUDA cores.