Machine Learning For Particle Identification

04.Mai.2018 I Waleed Esmail

Institut für Kernphysik (IKP), Forschungszentrum Jülich

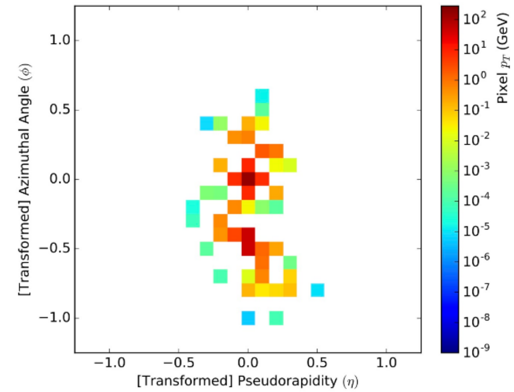
Outlines:

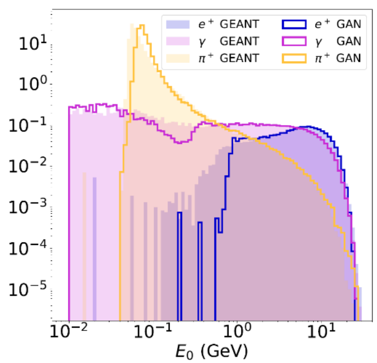
- Whetting Your Appetite.
- Introduction to Machine Learning.
- Motivating the Key Concepts.
- Boosted Decision Trees.
- Artificial Neural Networks.
- Integration to PandaRoot.
- Conclusion.

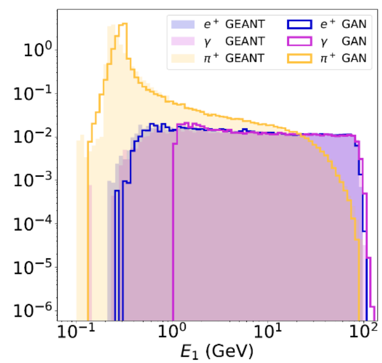
Whetting your Appetite:

- Why Multivariate Analysis (Machine Learning Techniques)?
  - Classification (Higgs discovery).
  - Clustering (Tracking).
  - Pattern Recognition (Tracking and Jet Images).
  - Generative Learning GAN (Simulation “GeantV”).

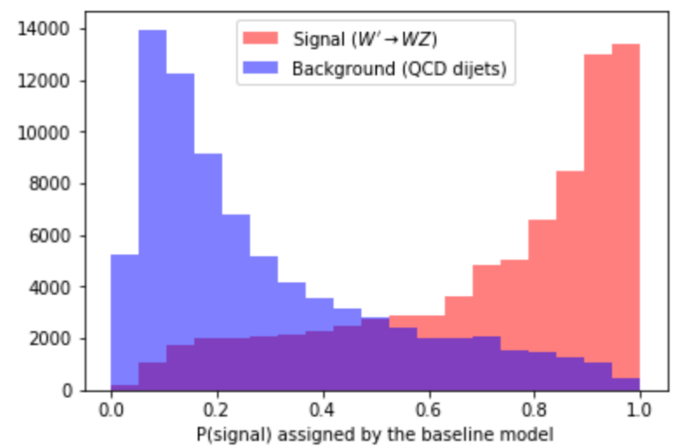
speed-up factors of up to 100,000x

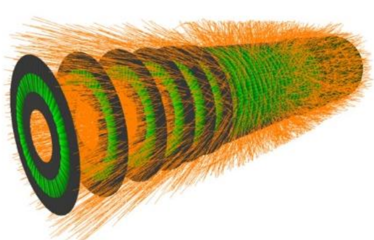






Jet Image using CNN.





Total prizes:
$25,000

Next important deadline:
August 6, 2018 - Entry deadline. You must accept the competition rules before this date in order to compete.

Join the competition

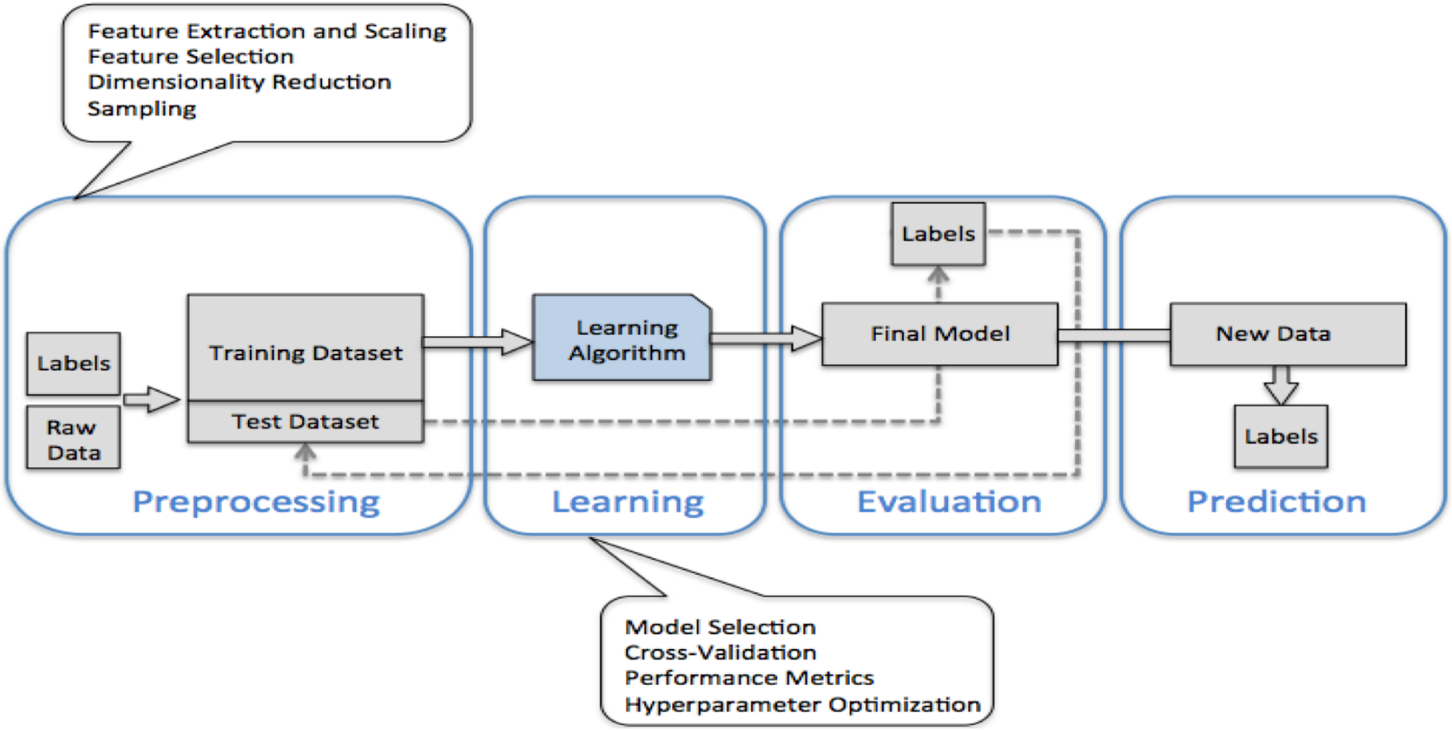
JÜLICH Forschungszentrum

Introduction to Machine Learning:

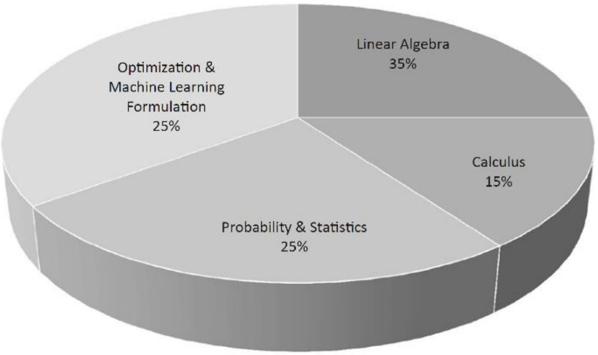
- Machine Learning (ML) is about modeling your data.
- Developing ***self-learning*** algorithms to gain knowledge from data in order to make ***predictions***.
- There are ***three*** types of ML ***algorithms***:

1. **Supervised Learning:**
   - Learn a model from labeled training data.
   - Classification (discrete labels), and Regression (continuous response variable).
2. **Unsupervised Learning:**
   - We deal with unlabeled data to explore its structure.
   - Clustering (subgrouping), and Dimensionality Reduction (feature preprocessing.)
3. **Reinforcement learning:**
   - Develop a system (agent) to improve performance via reward maximization.

Introduction to Machine Learning:

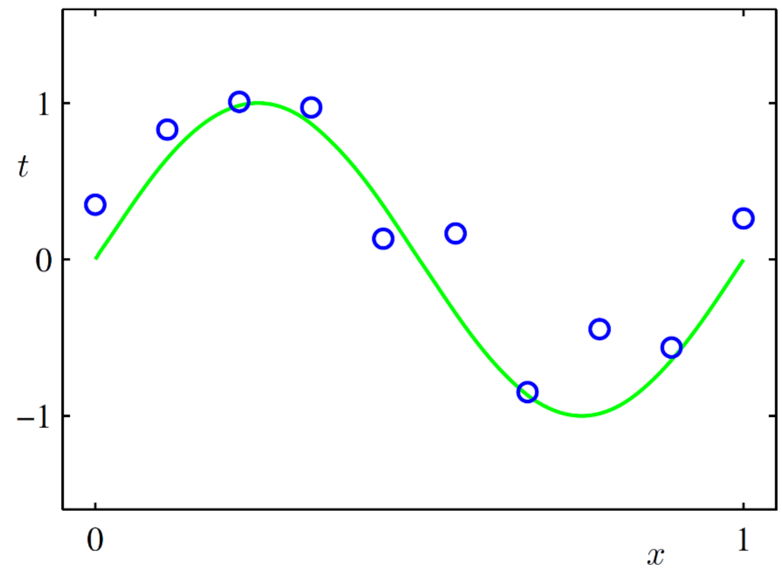


Strong grasp of the core concepts of mathematics enables one to ***select the right algorithm***, Also, it enables one to ***tune*** machine-learning/deep-learning models better.



Motivating the Key Concepts:

- A simple ***regression*** problem.
- We observe a real-valued input variable (x), and we wish to ***predict*** the value of a real-valued target variable (t).
- The ***green*** curve shows the function $\sin(2\pi x)$ used to generate the data.
- The ***blue*** points are obtained by adding Gaussian noise.
- Fit the data using a ***polynomial*** function:



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

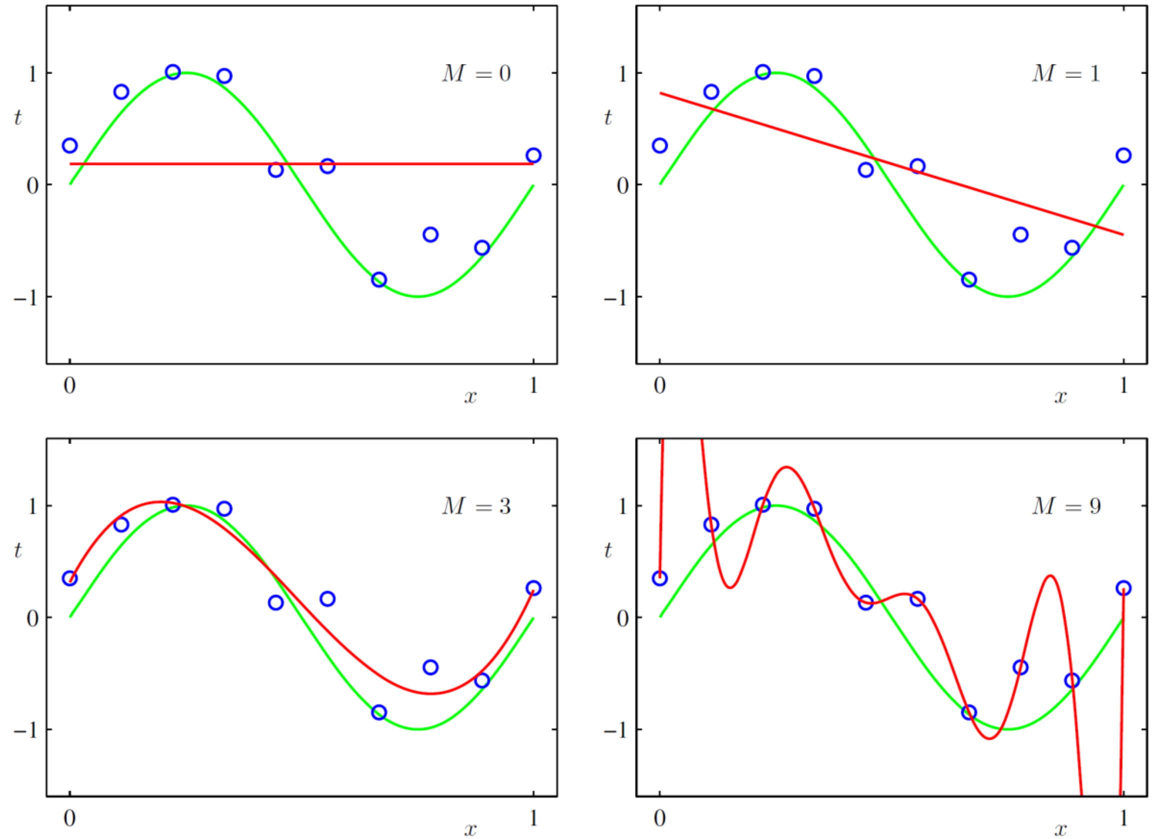
- Polynomial degree M ➡ (***model selection***).
- ***Weights*** can be determined by fitting polynomial to the training data, by ***minimizing*** an ***error function.***

JÜLICH Forschungszentrum

Motivating the Key Concepts:

- One common choice is the ***root-mean-square***.

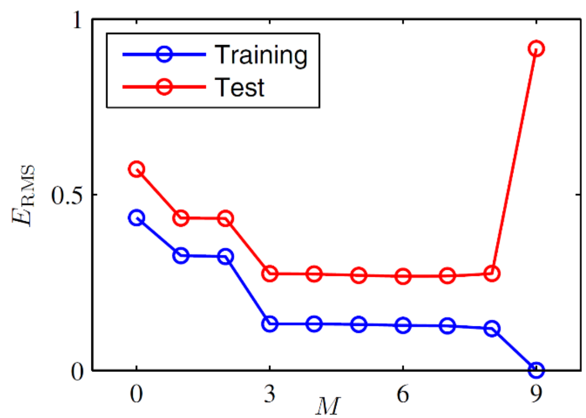
$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w})/N} \quad \Longrightarrow \quad E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2$$

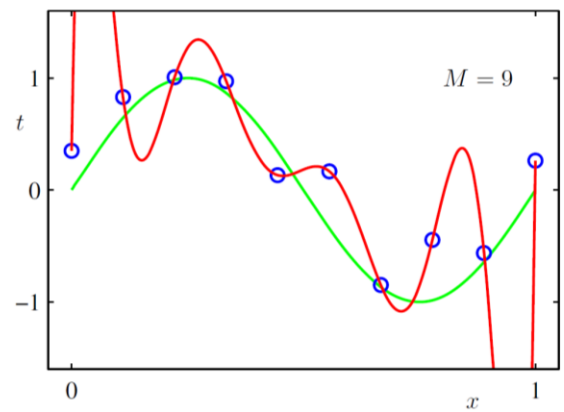


Motivating the Key Concepts:

- One common choice is the ***root-mean-square***.

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}\ )/N} \quad \Longrightarrow \quad E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2$$





Over-fitting

- But how the weights of the model can be computed. ***Probability*** theory comes to rescue.

Motivating the Key Concepts:

- Given (x), assume that the target variable (t) has a Gaussian distribution:

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right) \qquad \beta^{-1} = \sigma^2$$

- Determine the unknown parameters by ***maximum likelihood method.***

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|y(x_n, \mathbf{w}), \beta^{-1}\right)$$

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi).$$

- By maximizing with respect to (w), we obtain the desired solution (w_{ML}).

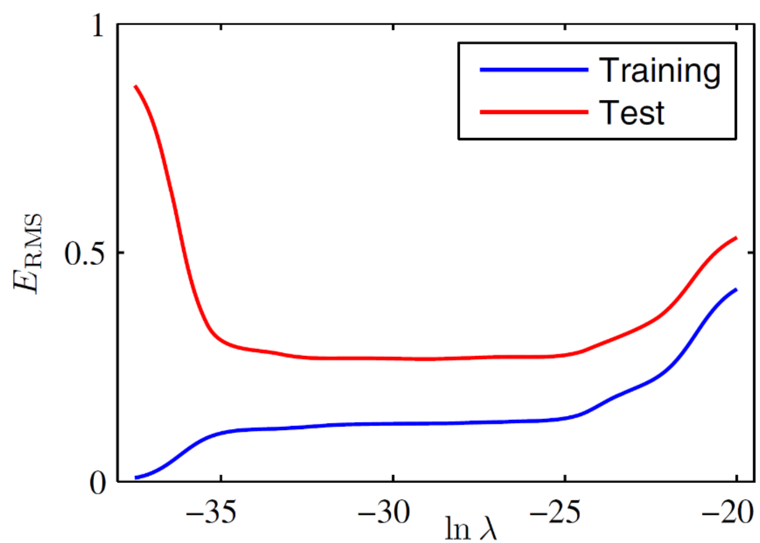
- Finally make new predictions.

Motivating the Key Concepts:

- Back to over-fitting issue, one technique to control it is called ***regularization***.
- Regularization involves adding ***plenty term*** to the error function.

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

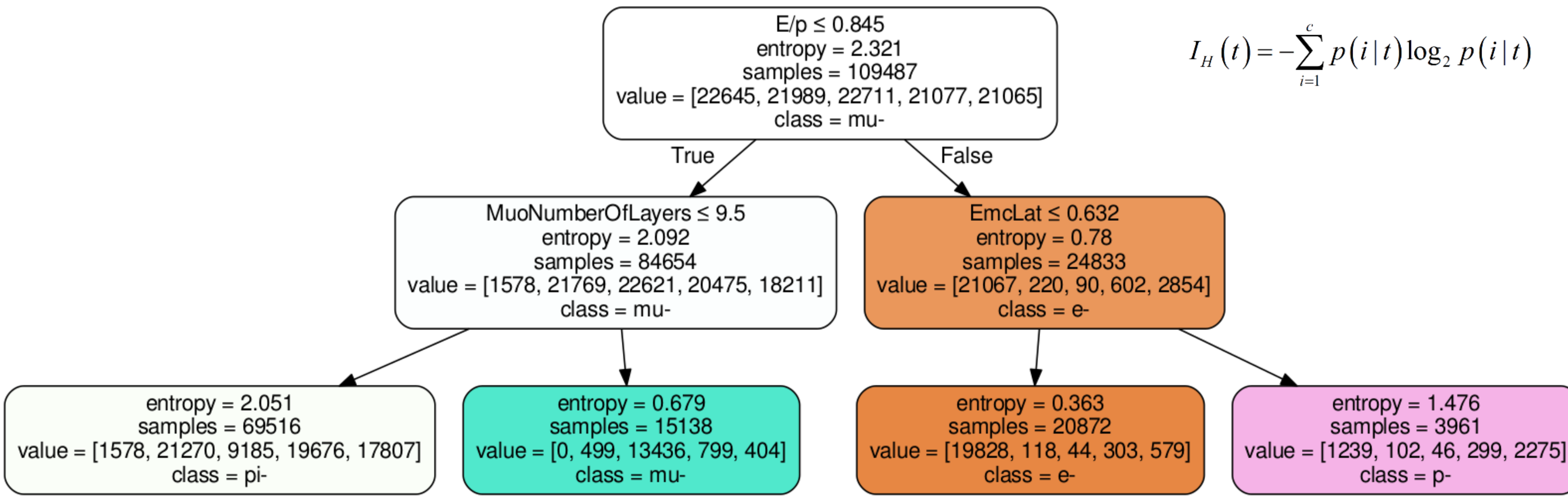
- λ governs the relative importance of the regularization term compared to the error term.



Boosted Decision Trees (BDT):

- Idea: ***break down the data*** by making decisions based on the features in the training set.
- The splitting is done by maximizing the **Information Gain** $IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$
- **BDT** is an **ensemble method.** The key concept behind **boosting** is to focus on training samples that are hard to classify.

$$I_H(t) = -\sum_{i=1}^{c} p(i|t)\log_2 p(i|t)$$



Boosted Decision Trees (BDT):

- Two event generators are used for training.

*1. **BoxGenerator:***

- **momentum range**: (0.2 - 5) GeV.
- **phi range**: 0 - 360°.
- **theta range**: 0 - 180°.
- **particle species**: [$e^{\mp}$, $\pi^{\mp}$, $\mu^{\mp}$, $k^{\mp}$, $p^{\mp}$]. One particle per event.

2. EvtGen:

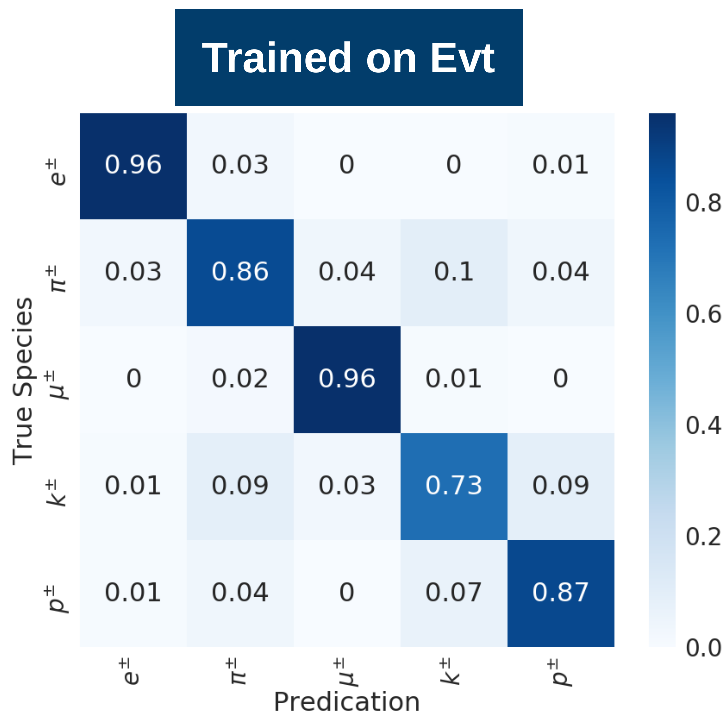
- $p\bar{p} \to X\bar{X}Y\bar{Y}$, where X, and Y = $e^{\mp}$, $\pi^{\mp}$, $\mu^{\mp}$, $k^{\mp}$, $p^{\mp}$
- **Beam momentum:** 15 GeV/c.

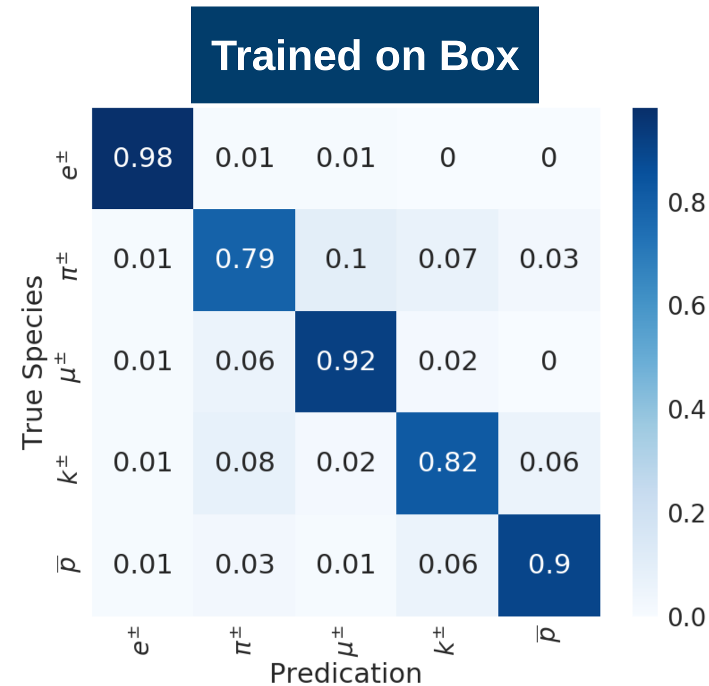
*Particles are matched to their **MC truth** information.*

Boosted Decision Trees (BDT):

- Data are organized into python ***DataFrame***. Data was splitted into **training** (**70%**) and **testing** (**30%**) sets.

energy	momentum	charge	position	MvdHits	GemHits	SttHits	TofStopTime	TofTrackLength	EmcCalEnergy
1.45992	2.1119	-1	1.53556E-4	3	0	26	0.0	0.0	0.250083
4.14557	17.1663	-1	6.65813E-6	5	0	17	3.95565	117.048	0.991413
3.51102	12.3078	-1	0.00648225	1	0	24	0.0	0.0	2.21215
3.45948	11.9486	-1	5.4671E-6	4	0	22	2.89786	86.4262	0.29421
4.78585	22.8849	-1	6.36045E-5	3	0	26	0.0	0.0	2.62603

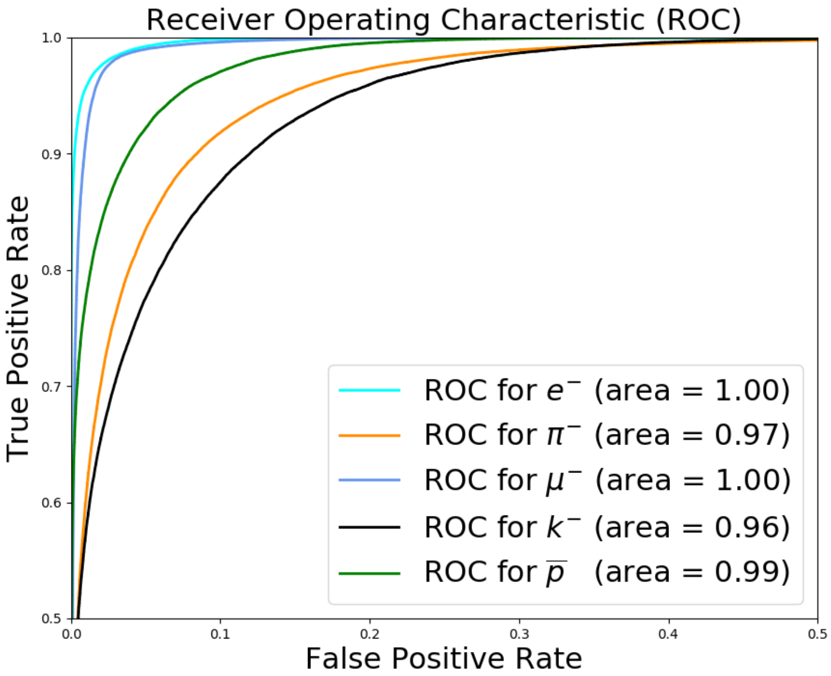




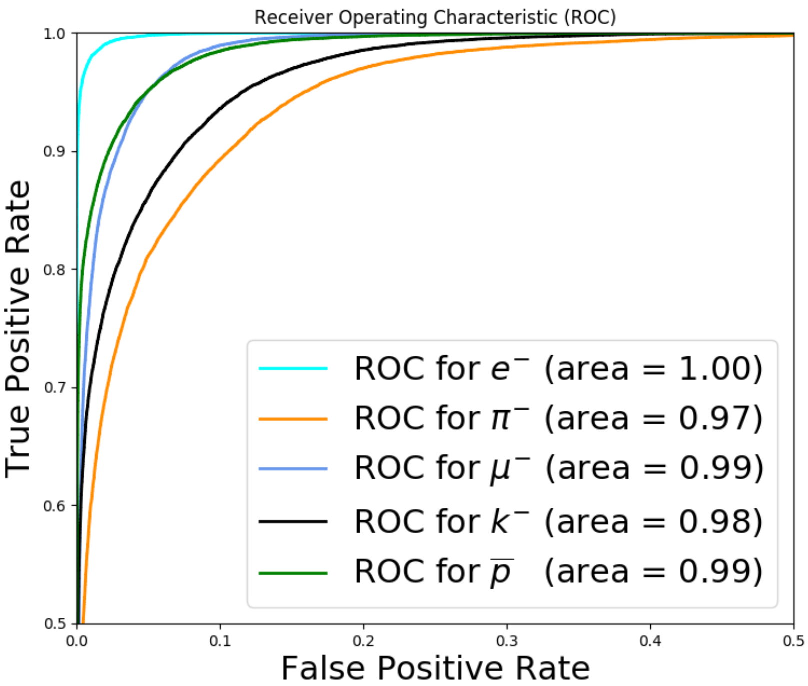
Boosted Decision Trees (BDT):

1. Receiver Operating Characteristic (ROC).

Trained on Evt



Trained on Box

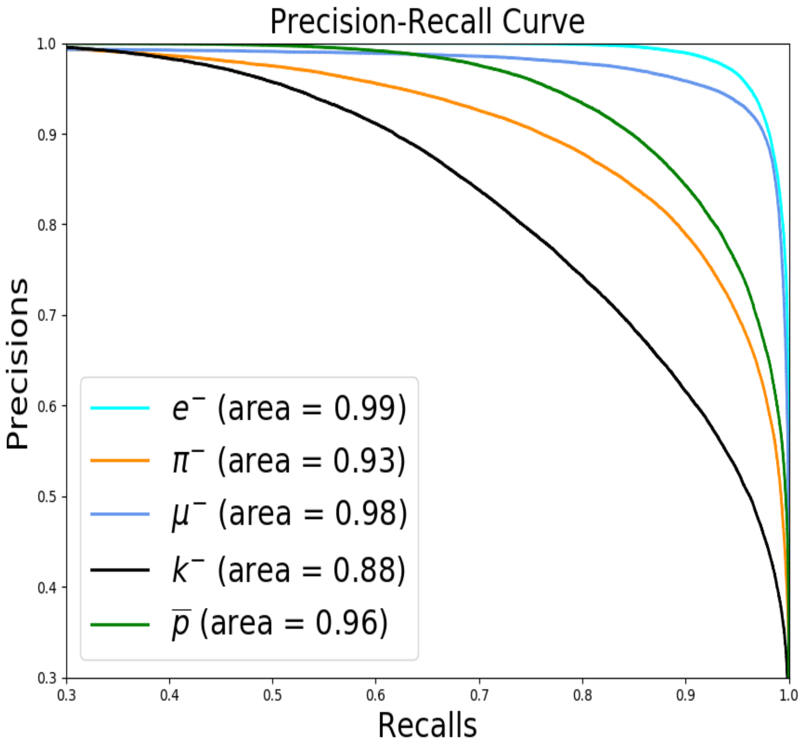


True Positives (TP) & True Negatives (TN).
False Positives (FP) & False Negatives (FN).

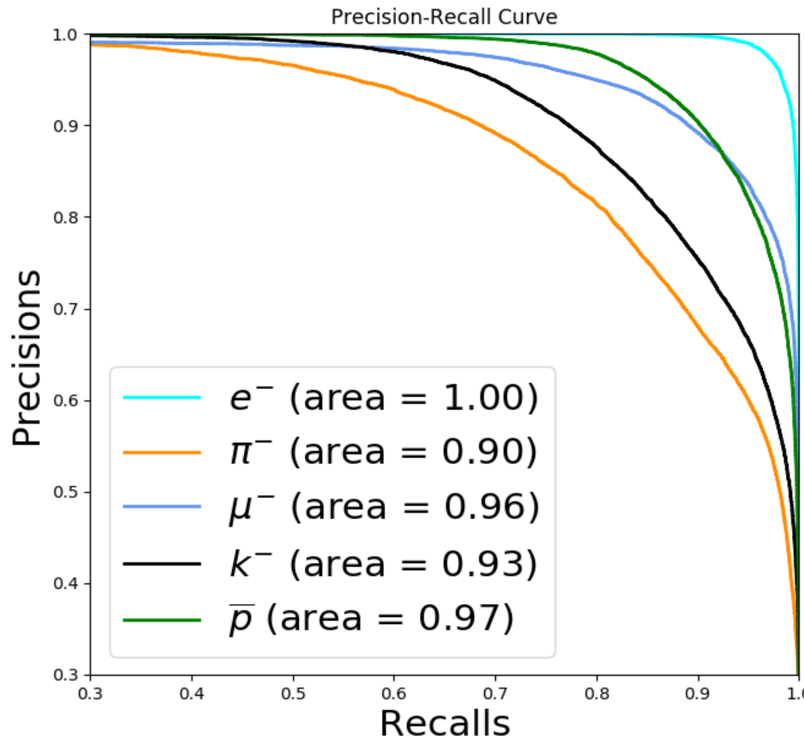
Boosted Decision Trees (BDT):

2. Precision, and Recall.

Trained on Evt



Trained on Box



$$\text{recall} = \frac{TP}{TP + FN} \quad \text{precision} = \frac{TP}{TP + FP}$$

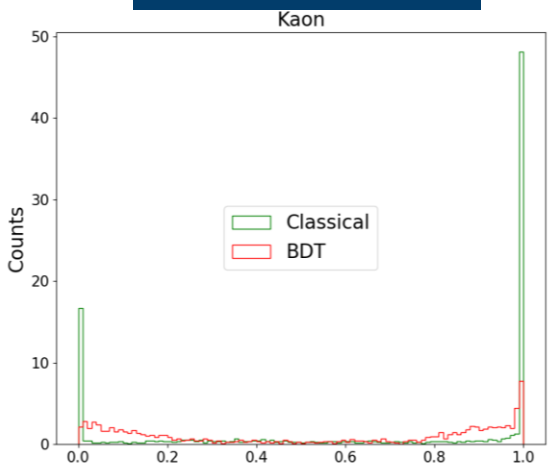
JÜLICH Forschungszentrum

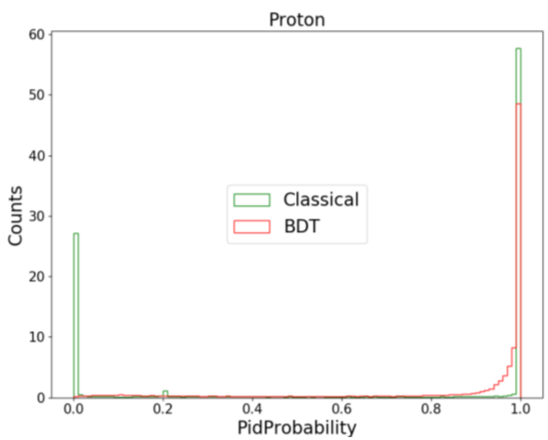
Boosted Decision Trees (BDT):

Particle Identification (PID) Probabilities.

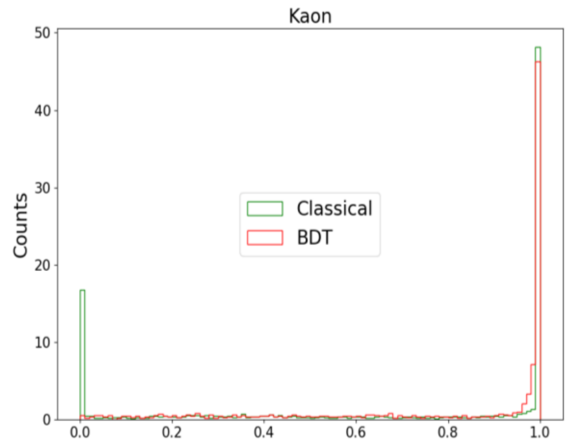
We tested the trained algorithm on data generated by **DPM generator** (elastic + inelastic).

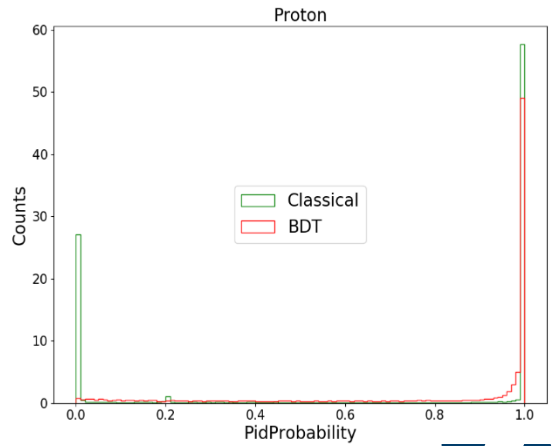
Trained on Evt



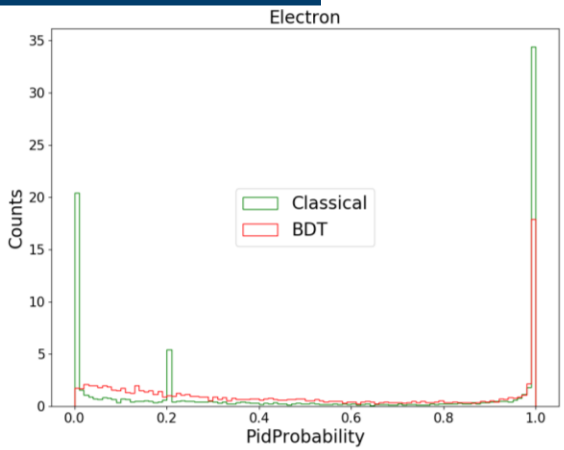


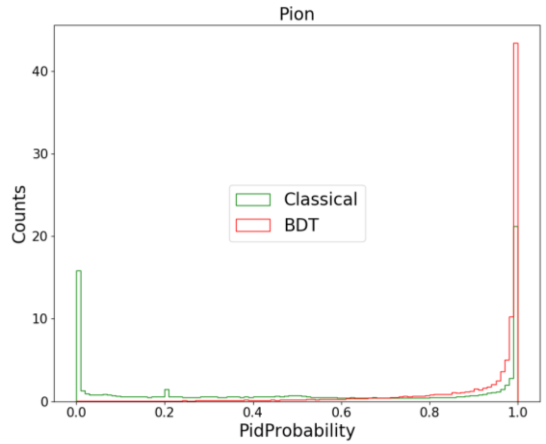
Trained on Box

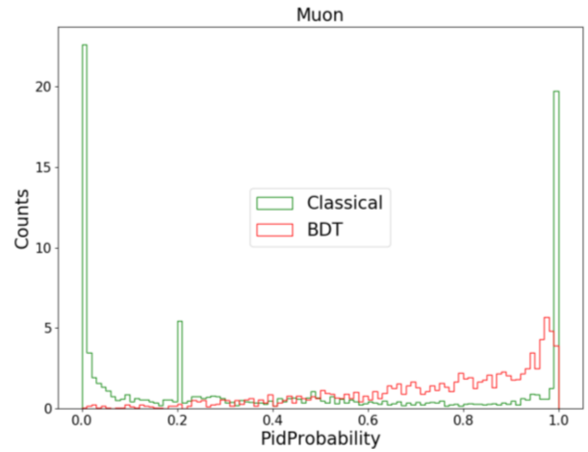




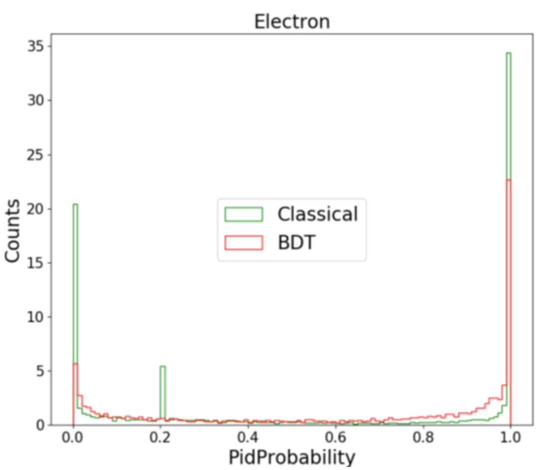
Trained on Evt

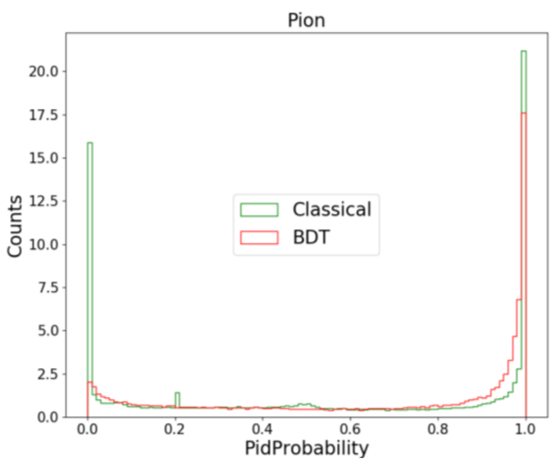


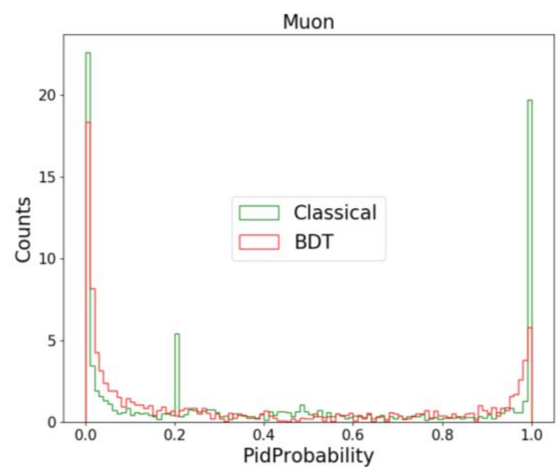




Trained on Box



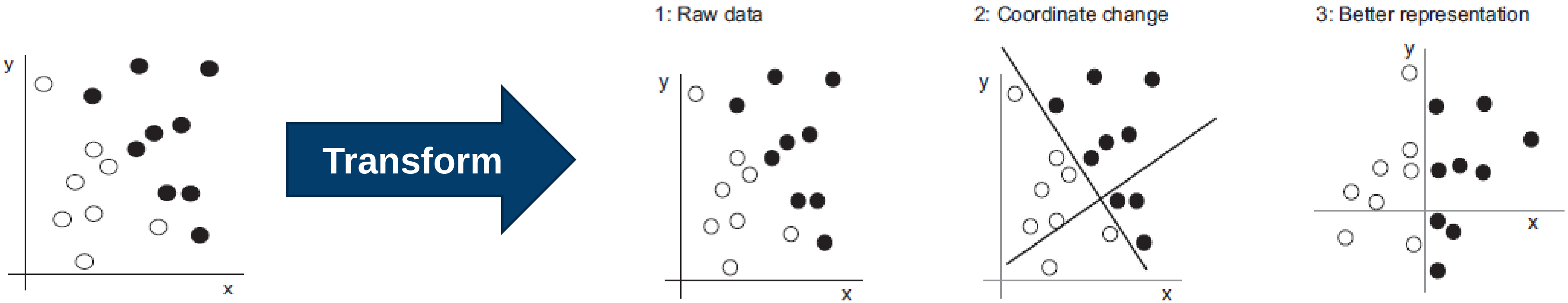


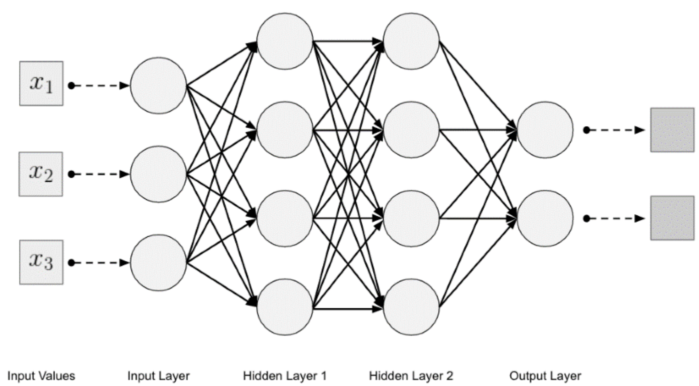


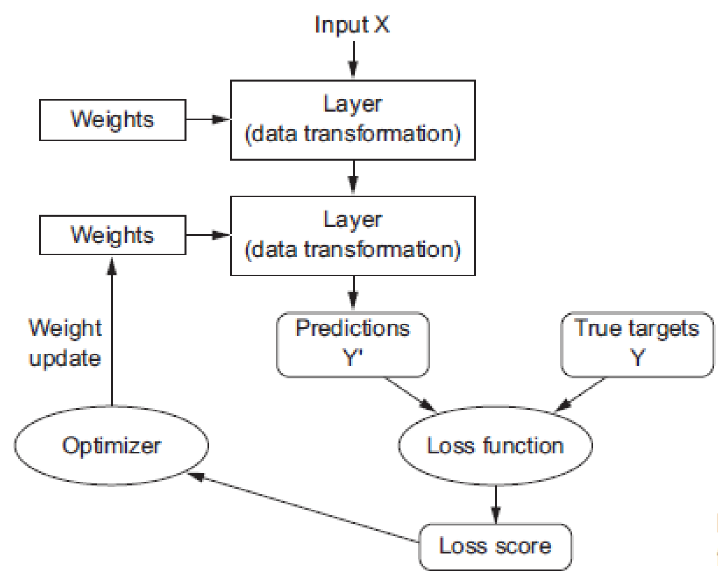
Deep Learning:

Artificial Neural Networks (ANN):

- Neural Networks is about meaningfully ***transform the data.***



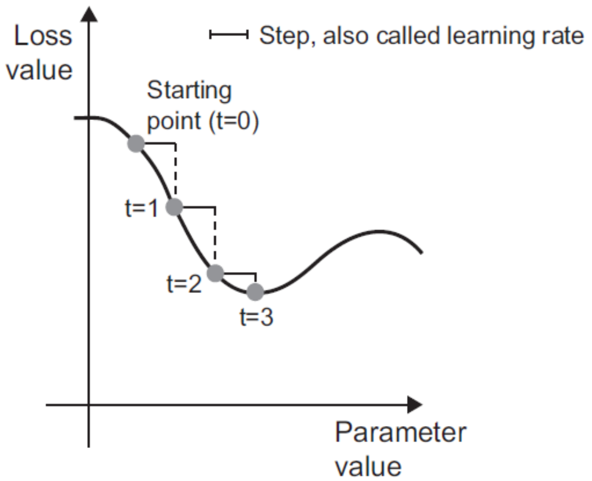


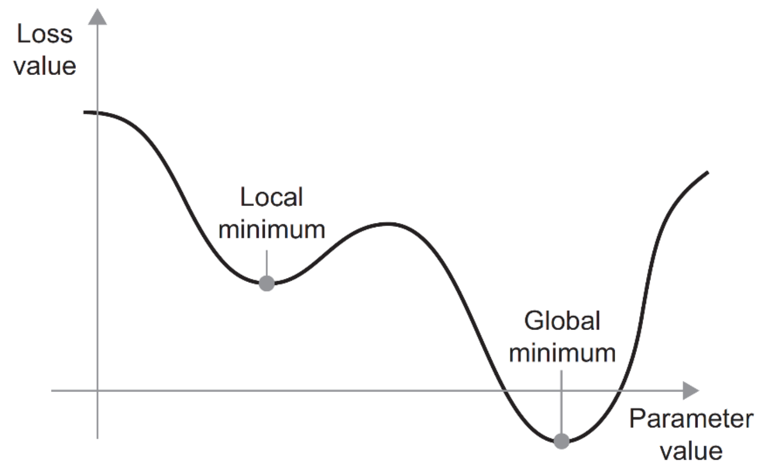


- *Training loop of DNN:*
  - ***Input data***, parameterize by ***weights*** (transform the data), predict and compute the ***loss score***, and update (***epochs***).

JÜLICH Forschungszentrum

Artificial Neural Networks (ANN):



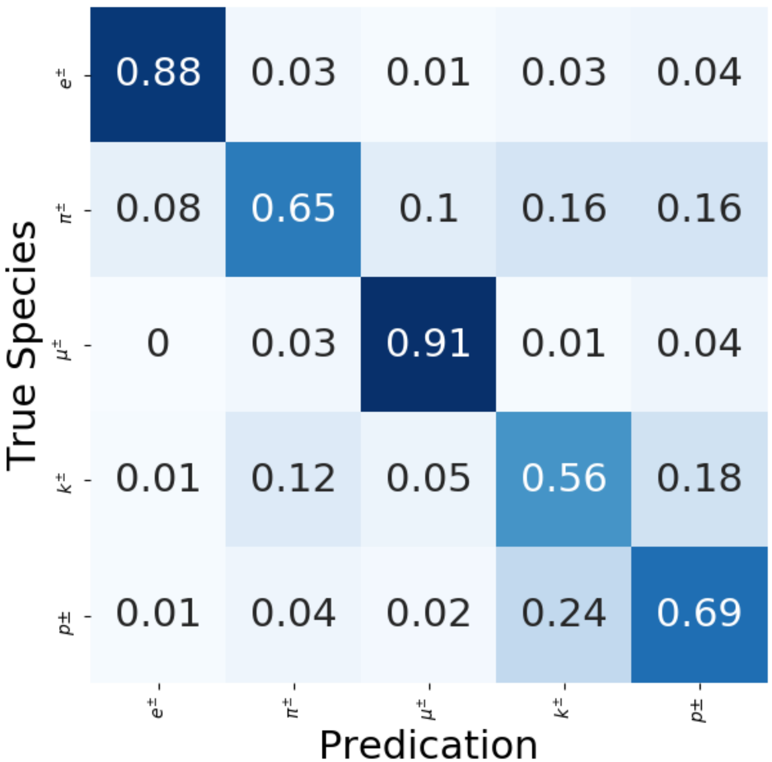


- The update step is guided by ***minimizing the loss function*** by calculating its ***gradient***.
- Model parameters (weights) are updated through back-propagation algorithm.

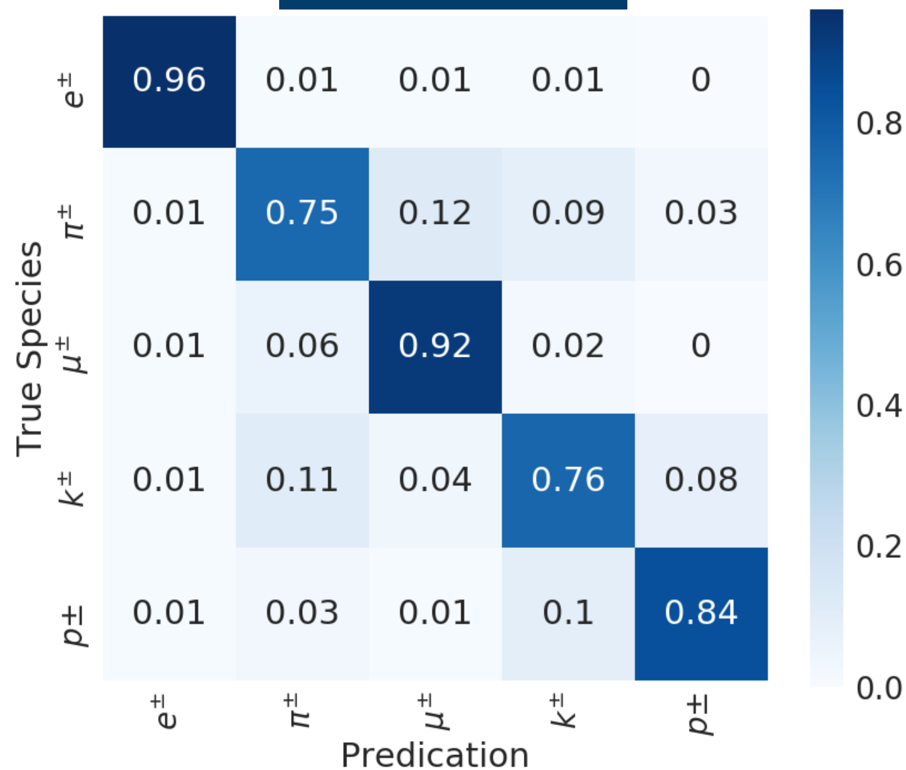
- ***Keras*** Python Package was used.
- Keras is built on top of ***tensor-flow***, and can run on GPUs.

Artificial Neural Networks (ANN):

Trained on Evt



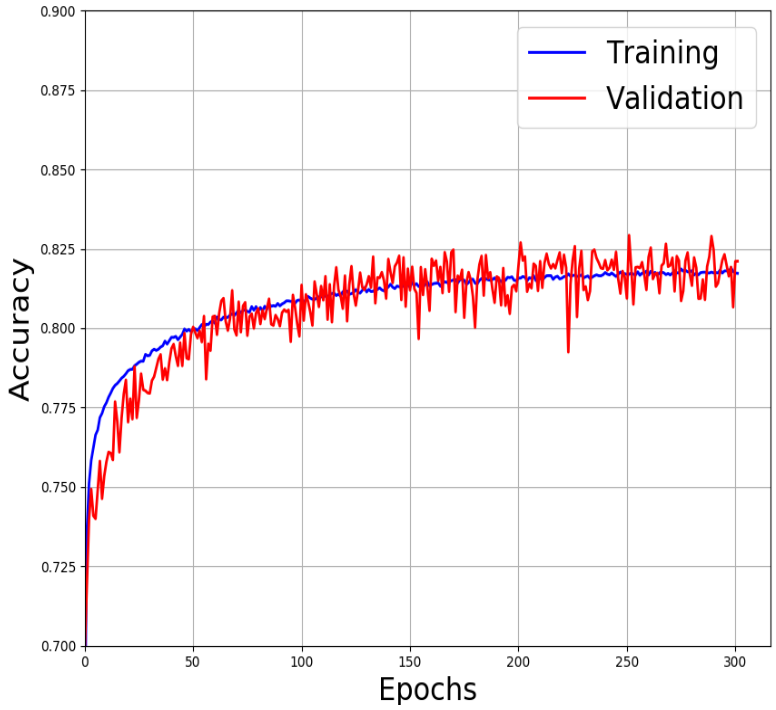
Trained on Box



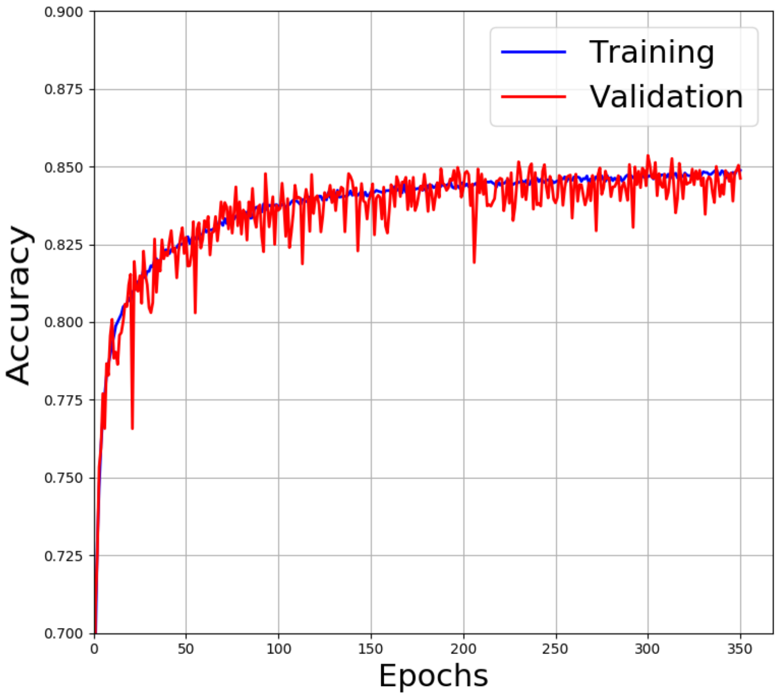
JÜLICH
Forschungszentrum

Artificial Neural Networks (ANN):

Trained on Evt



Trained on Box

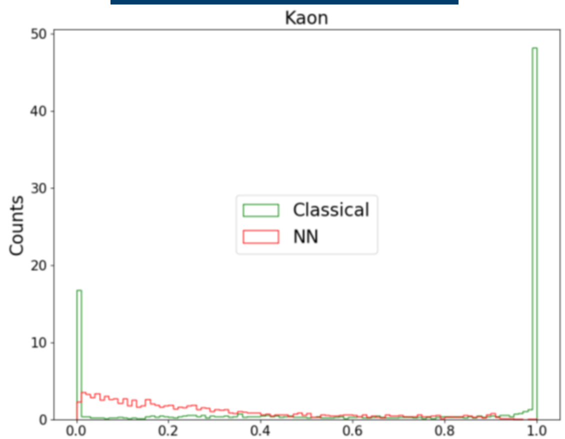


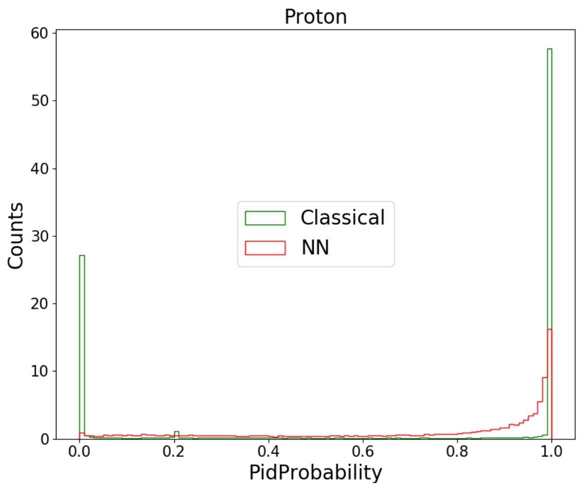
Artificial Neural Networks (ANN):

Particle Identification (PID) Probabilities.

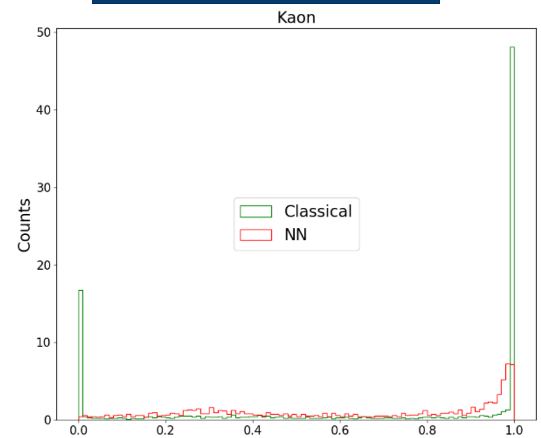
We tested the trained algorithm on data generated by **DPM generator** (elastic + inelastic).

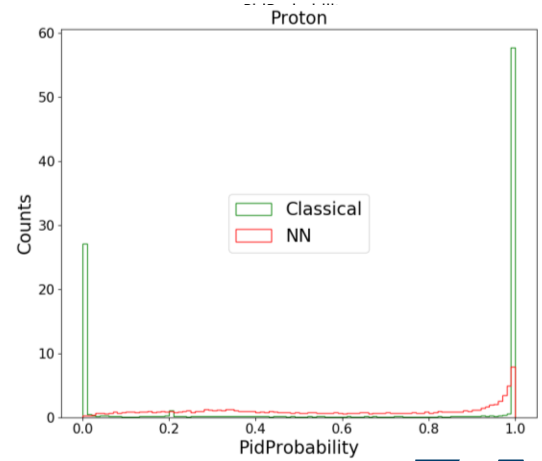
Trained on Evt



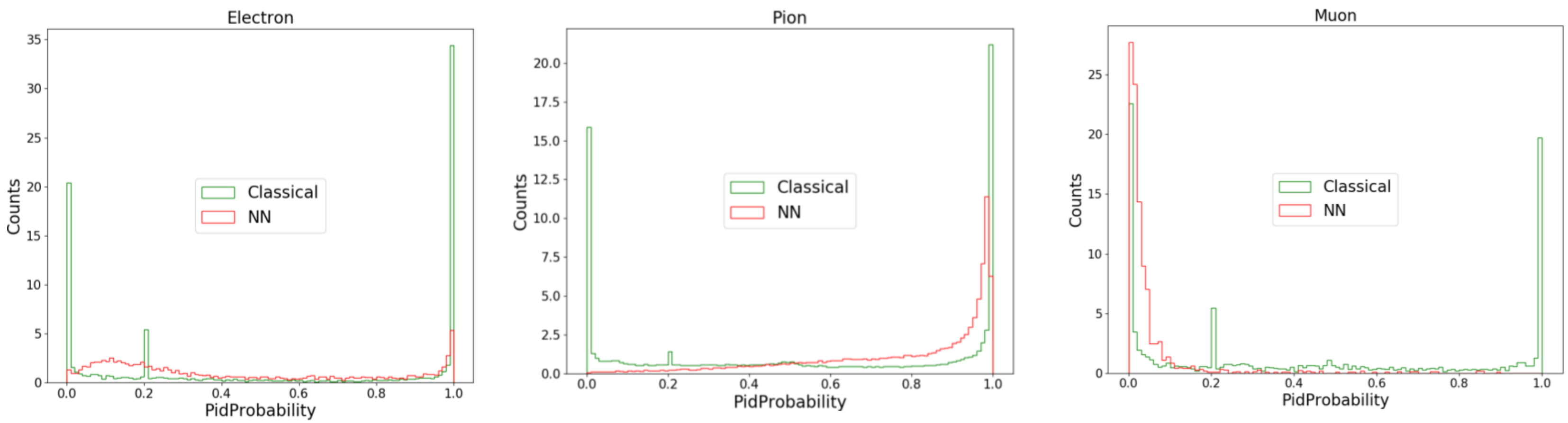


Trained on Box

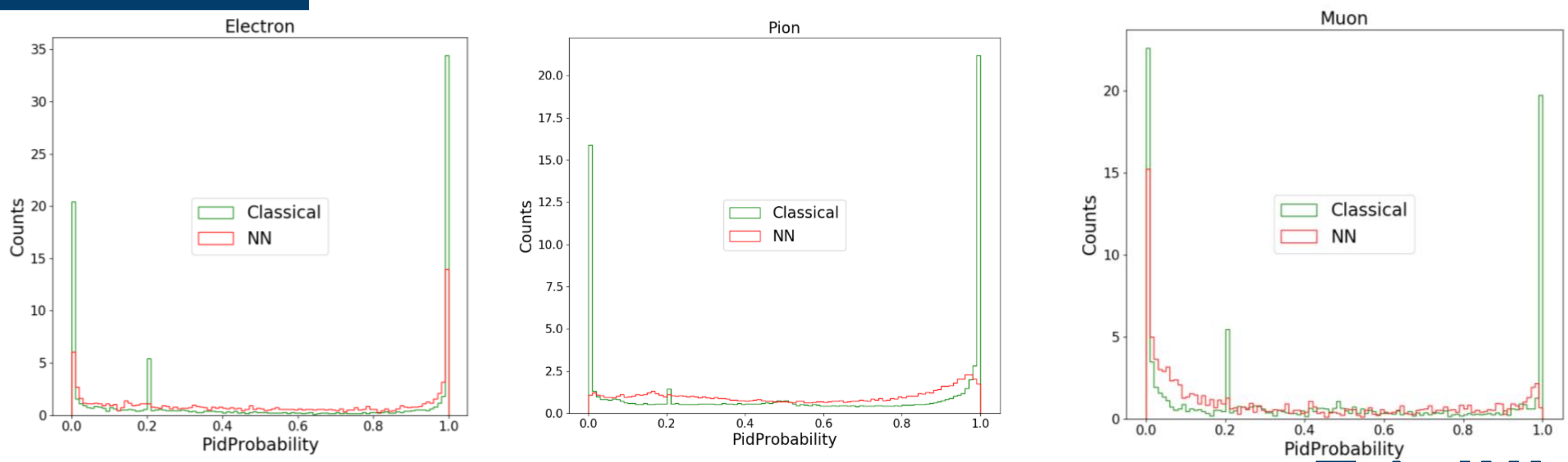




Trained on Evt



Trained on Box



Integration to PandaRoot:

.

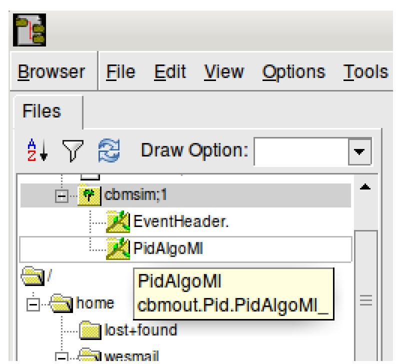
```
In [1]: from sklearn.externals import joblib
        clf = joblib.load('savedRF.pkl')
```

```
In [5]: entry_to_predict_on = s_df.iloc[:,0:-1]
        particles = clf.predict_proba(entry_to_predict_on)
        List = ['e-','pi-','mu-','k-','p-']
        particles = pd.DataFrame(particles, columns=List)
        particles['event'] = s_df.loc[:,'events']
```

Associator Task

output

PndPidMlAssociatorTask



Integration to PandaRoot:

```
FairRunAna* fRun = new FairRunAna();
FairRuntimeDb* rtdb = fRun->GetRuntimeDb();
fRun->SetInputFile(inPidFile);
fRun->AddFriend("signal_ml.root");
```

```
theAnalysis->FillList(muplus,  "MuonTightPlus",  "PidAlgoMl");
theAnalysis->FillList(muminus, "MuonTightMinus", "PidAlgoMl");
theAnalysis->FillList(piplus,  "PionLoosePlus",  "PidAlgoMl");
theAnalysis->FillList(piminus, "PionLooseMinus", "PidAlgoMl");
```

Integration to PandaRoot:

- **https://github.com/wesmail/MLPID_For_PANDARoot**

wesmail Update README.md		Latest commit fa3848a 6 hours ago
.gitattributes	RF classifier	a month ago
Classifier.py	Update Classifier.py	a month ago
PndPidMlAssociatorTask.cxx	Create PndPidMlAssociatorTask.cxx	a month ago
PndPidMlAssociatorTask.h	Create PndPidMlAssociatorTask.h	a month ago
README.md	Update README.md	6 hours ago
savedRF.pkl	RF classifier	a month ago
tut_ana.C	Update tut_ana.C	a month ago

To be able to use this code you have to be sure that Python interpreter is installed on your machine. If you do not have Python, Please navigate to Python website https://www.python.org/. It is preferable to install Python2 since we are using ROOT, and usually Python3 with ROOT have a lot of problems.

1. Having Python2 installed we can now install the necessary Python modules. The required packages are:
   - ROOT (Python bindings of ROOT https://root.cern.ch/building-root).
   - ipython
   - jupyter (optional)
   - scipy
   - numpy
   - root_numpy
   - pandas
   - matplotlib
   - seaborn
   - tensorflow (for deep learning)
   - tensorboard (for deep learning)
   - keras (for deep learning)

to install any of these modules use pip, e.g `pip install ipython`. If you do not know the exact name of the module use `pip search` and to narrow the search use `grep` with `pip`, e.g `pip search numpy | grep numpy`.

2. The second step is to copy these two files `PndPidMlAssociatorTask.cxx` and `PndPidMlAssociatorTask.h` to: `yourPandaRootDirectory/source/pid/PidClassifier/`. Then open `CMakeLists.txt` and add the following lines: `PidClassifier/PndPidMlAssociatorTask.cxx` and `PidClassifier/PndPidMlAssociatorTask.h`.

You can find easily where to paste these lines. Then open `PidLinkDef.h` and add the following line:
`#pragma link C++ class PndPidMlAssociatorTask+;`

Finally compile PANDARoot again from the build directory.

Available Multivariate packages:

- TMVA (ROOT).
- MLlib Apache Spark.
- Sci-Kit Learn.
- TensorFlow (Deep Learning).
- Keras (Deep Learning).
- PyTorch (Deep Learning).
- DL4J (Deep Learning, Java).
- R Implementations.
- ...

Conclusion and outlook:

- *Boosted Decision Trees (BDT)* showed good performance in classifying charged particles and outperformed the classical PID methods over the specified momentum range.

- *Neural Networks* also showed poor performance in the classifying task (needs more deep investigations).

- Interface between Python and PandaRoot by ZeroMQ.
- Write a release note of what have been done.

THANK YOU

BACK UP

Boosting:

1. Draw a random subset of training samples d_1 without replacement from the training set D to train a weak learner C_1.
2. Draw second random training subset d_2 without replacement from the training set and add 50 percent of the samples that were previously misclassified to train a weak learner C_2.
3. Find the training samples d_3 in the training set D on which C_1 and C_2 disagree to train a third weak learner C_3.
4. Combine the weak learners C_1, C_2, and C_3 via majority voting.