Machine Learning In High Energy Physics 2018 Tutorial

Elliot Parrish

Resources

Original Jupyter Notebooks from MLHEP2018 are here:

- https://github.com/yandexdataschool/mlhep2018

Condensed Jupyter Notebooks and lectures

- https://github.com/eparrish64/NICADD-ML-Tutorials.git

NICADD Cluster

- cms1.nicadd.niu.edu
  - Use either /xdata/USER or /bdata/USER
    - Do not use tdata
  - 1 GPU available with 4 GB of memory
  - 12 CPUs with ~66 GB
- http://nicadd.niu.edu/nhpc/index.php?n=Nhpc.Hardware

Before We Start

Log on to cms1.nicadd.niu.edu

- git clone https://github.com/eparrish64/NICADD-ML-Tutorials.git
- source setup.sh
  - Agree to conda license
  - Specify /xdata/$USER/miniconda3
  - Say yes to initialize conda

```
Do you wish the installer to initialize Miniconda3
by running conda init? [yes|no]
[no] >>> yes
```

- jupyter notebook --no-browser –port=NUMBER
  - Available ports
    - Pick a random number between 1023-65535
    - Not one someone else is using

```
[I 21:09:20.760 NotebookApp] The Jupyter Notebook is running at:
[I 21:09:20.761 NotebookApp] http://localhost:2018/?token=7fdf823347fcc683b9999bcabf9c9af2fbeaf4ae5cd0c5dd
[I 21:09:20.761 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 21:09:20.772 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///run/user/13009/jupyter/nbserver-108504-open.html
    Or copy and paste one of these URLs:
        http://localhost:2018/?token=7fdf823347fcc683b9999bcabf9c9af2fbeaf4ae5cd0c5dd
```

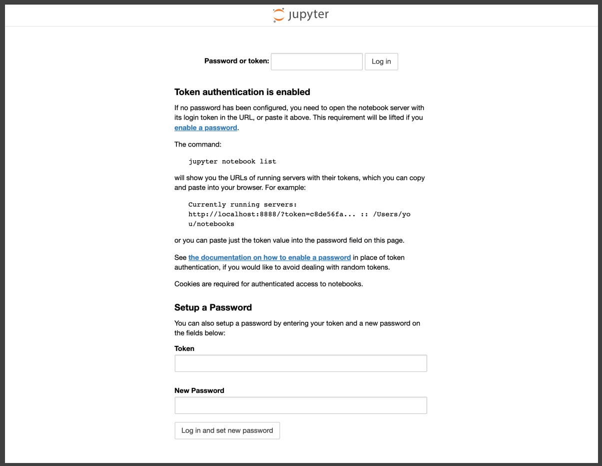
Open new terminal from your local machine

- ssh -L PORT:localhost:PORT USER@cms1.nicadd.niu.edu –N

```
ssh -L 1999:localhost:1999 eparrish@cms1.nicadd.niu.edu -N
```

Open web browser

- localhost:PORT
  - Copy token from nicadd terminal, use as password

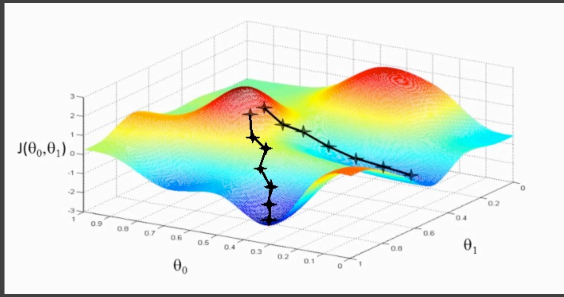


What is Machine Learning?

Function Approximation
- Trying to find coefficients for a function through “random guessing”

Loss Functions
- Defines the difference between the “guess” and the “true answer”
- Lots of ways to define this!
  - Mean square loss, root mean square loss, mean absolute error, EMD
- Typically, we minimize this

Minimization
- Gradient Descent
  - Make a guess
  - Calculate the gradient of decision and minimize
    - i.e. calculate the direction of the greatest change, then head that way



Take a subset of the data, minimize (train) using that, then apply to full dataset

Why does it work?
- Because it does

Machine Learning in HEP

Analysis
- Classification
  - Signal from background
- Novelty Detection
  - https://indico.cern.ch/event/745718/contributions/3174405/attachments/1754647/2844404/Novelty_Detection_FNALMLworkshop_2018Nov.pdf

Reconstruction
- Particle identification
- Energy calibrations
- Object reconstruction

Trigger
- Quicker convergence on complex final states

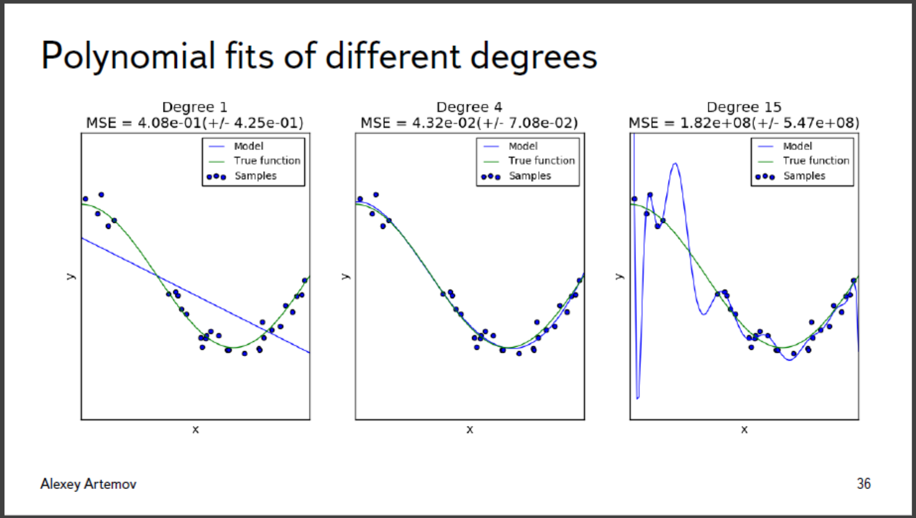
Computing
- Dataset management

A competing approach to feature engineering

Overtraining

Need to be careful about over specialization

- Want to be able to apply learned model to new datasets

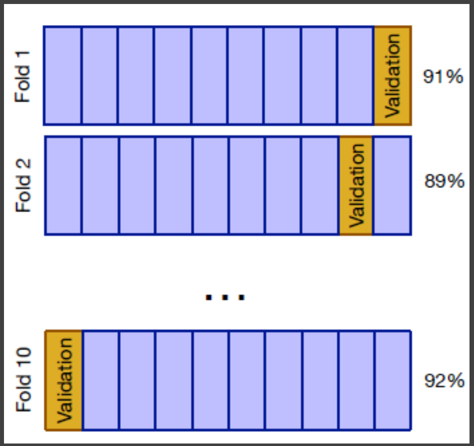


Bootstrapping

- Create new datasets by uniformly randomly sampling full dataset
- Train machine on each new dataset
- Use average of outputs

Cross-validate to prevent overtraining

- K-fold technique



Decision Trees

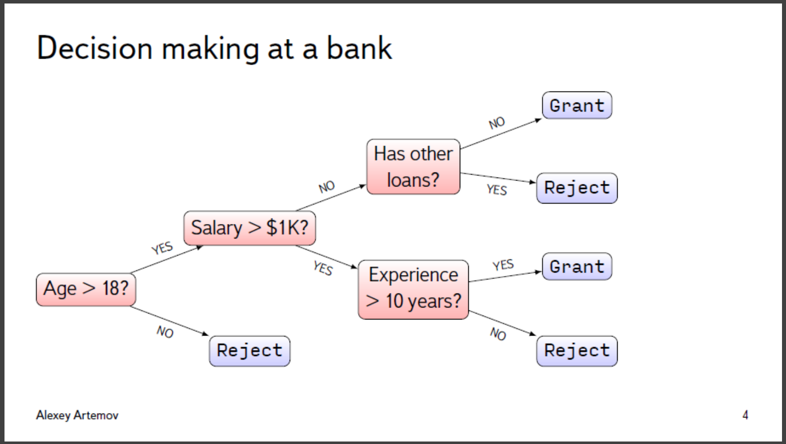
Things to tweak

- Maximum depth
- Maximum number of leaves
- Maximum number of objects in leaves
- How to stop
  - Constrained quality improvement
  - All objects fall into same leaf

Hyperparameters chosen via "guess and check"

- Can even use ML to optimize hyperparameters

Very susceptible to overfitting!!



Tree Ensembles

Random Forest
- Many weak trees over bootstrapped samples, average outputs

Stacking
- Blend output of weak learners with raw features

Boosting
- Start with one learner, improve on original learner with more learners, compute loss function, adjust new learner based off output from previous learner

Lots of hyperparameter optimization

Great at linear feature extraction
- What about nonlinear?
  - Neural Nets!

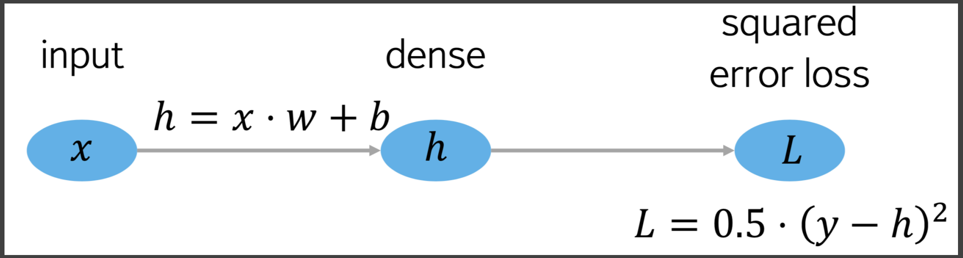
Tree Tutorial

https://github.com/eparrish64/NICADD-ML-Tutorials/blob/master/Tutorials/DecisionTrees.ipynb

Neural Networks

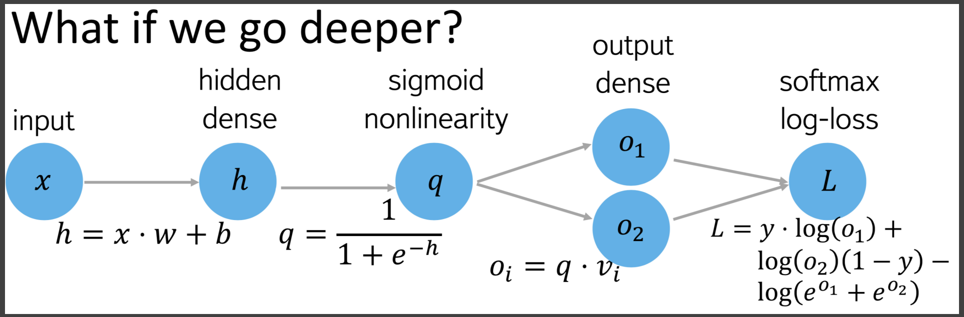
Basic structure
- Input layer
- Hidden layers
  - Dense layer
    - i.e. Linear fitting
  - Nonlinearity layer
    - i.e. Adjust linear fitting to nonlinear model (lots of options)
  - More
    - Can make this as complex as you like
      - Must be able to compute gradient
- Output Layer
  - Activation
    - Function to map output
      - [-1,1], [0,1], etc.

Backpropagation
- Start from last layer
- Calculate gradient of loss function w.r.t. each weight
  - Chain Rule
- Minimize gradient
  - Parameter optimization



Pros:
- Allows many inputs
- Fantastic at non-linearity

Cons:
- Very large number of hyperparameters
- Black box



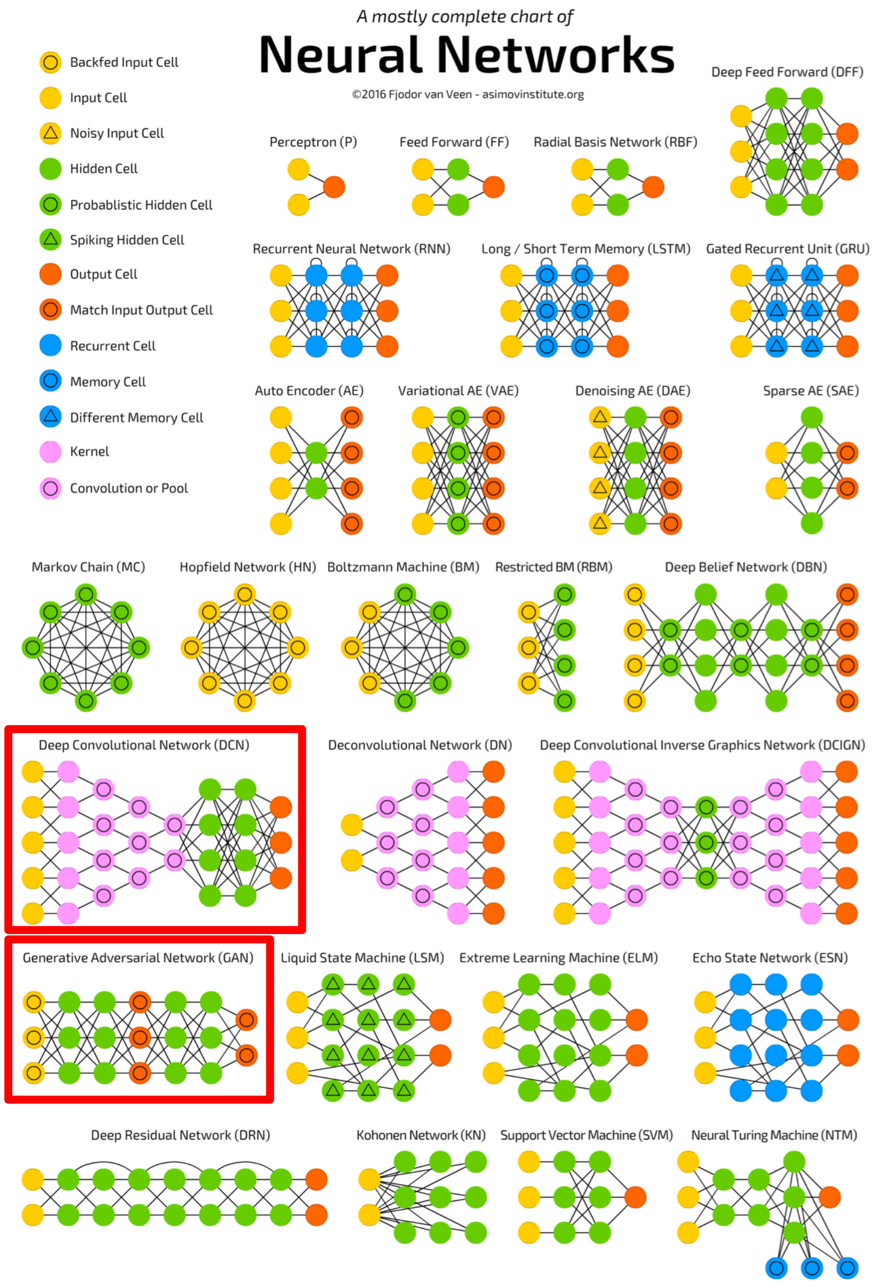
DIY Neural Network

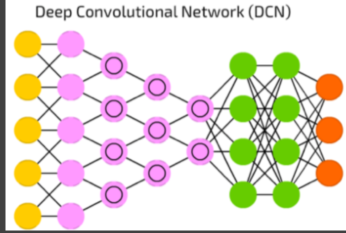
https://github.com/eparrish64/NICADD-ML-Tutorials/blob/master/Tutorials/DIY_NeuralNetwork.ipynb

Many different types of Neural Networks

Going to take a deeper look at

- Convolutional Neural Networks
- Generative Adversarial Networks (GANs)

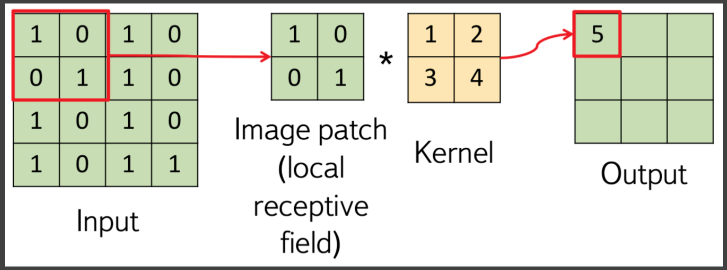


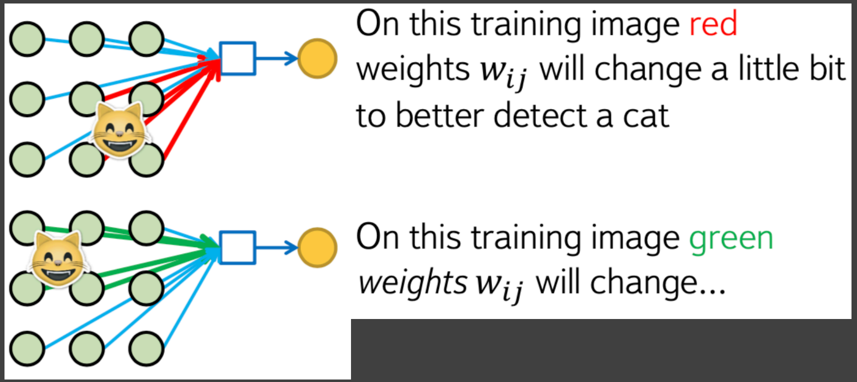


Convolutional Neural Networks

Convolutions

- Apply a kernel (filter) to image

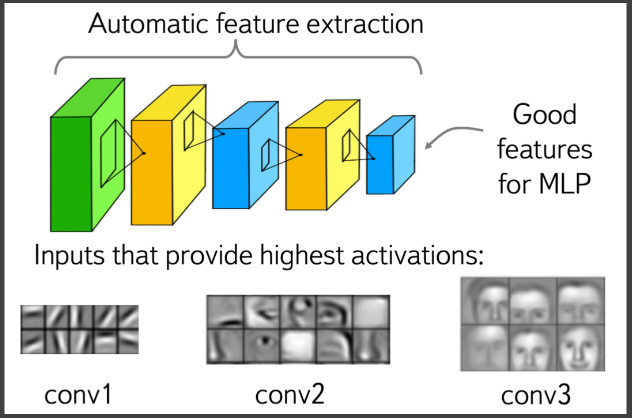




- Gradients of shared weights are summed

Can adjust size of kernel to find larger patterns

- Or use multiple kernels!



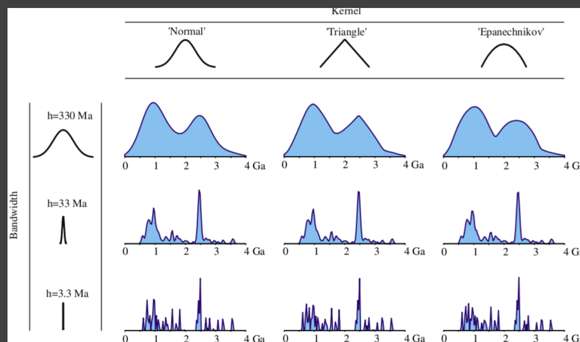
CNN Tutorial

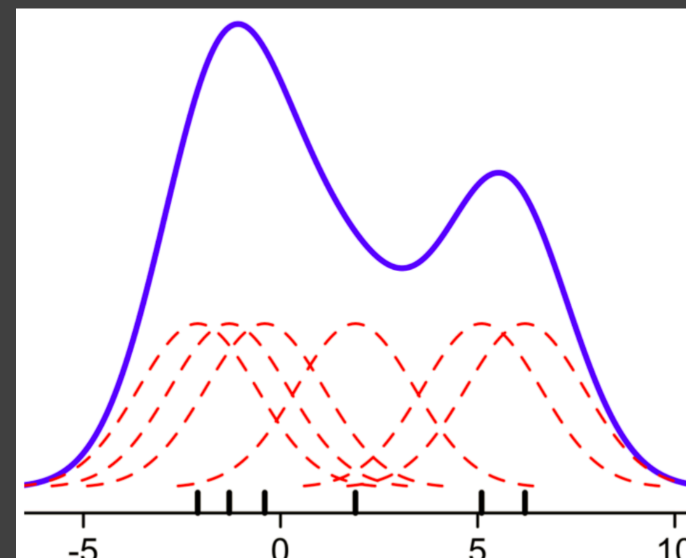
https://github.com/eparrish64/NICADD-ML-Tutorials/blob/master/Tutorials/CNN_Keras.ipynb

Generative Models

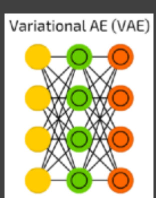
Trying to find underlying Probability Density Function from distribution

- Specialize kernel to distribution shape

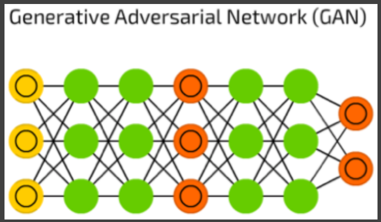




Variational Autoencoder



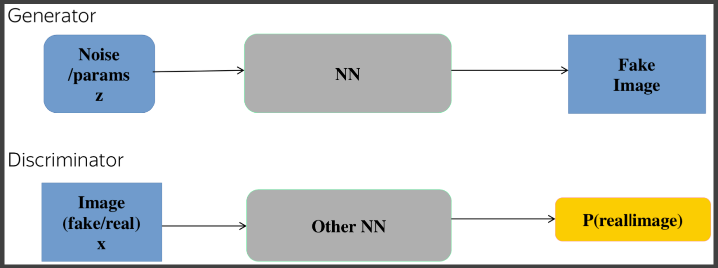
- Given
  - Data
  - Distribution
  - Parameterized model
- Maximizes likelihood of parameters

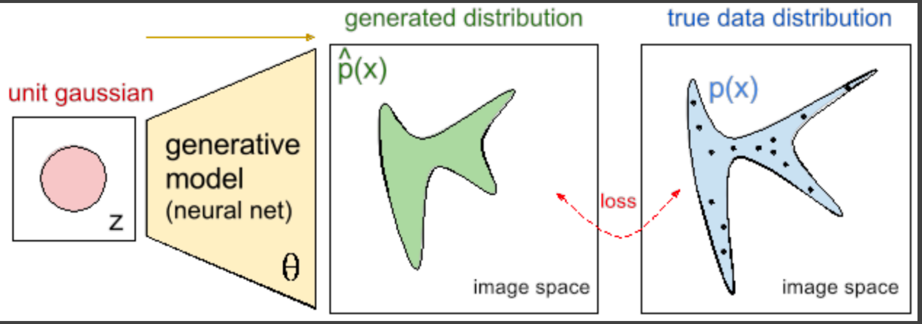


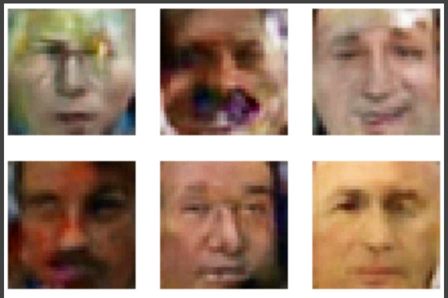
Generative Adversarial Networks

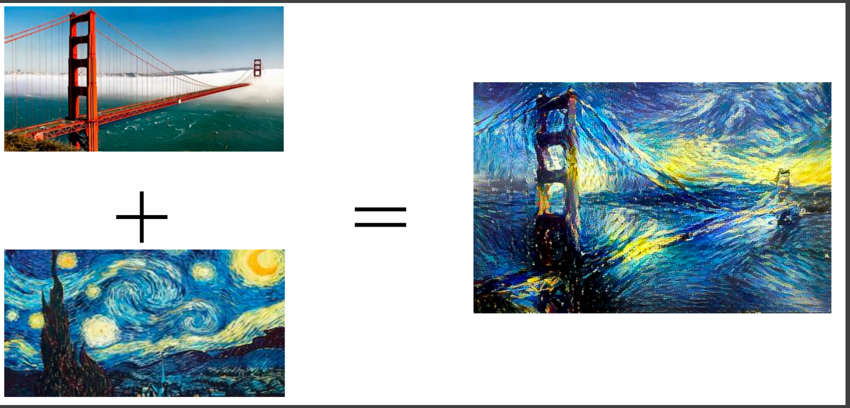
Use one network to constrain another

- Typically 1:1 generator to discriminator
  - Can use multiple discriminators for 1 generator









Pythia GAN Tutorial

https://github.com/eparrish64/NICADD-ML-Tutorials/blob/master/Tutorials/Pythia-Tune-AVO.ipynb

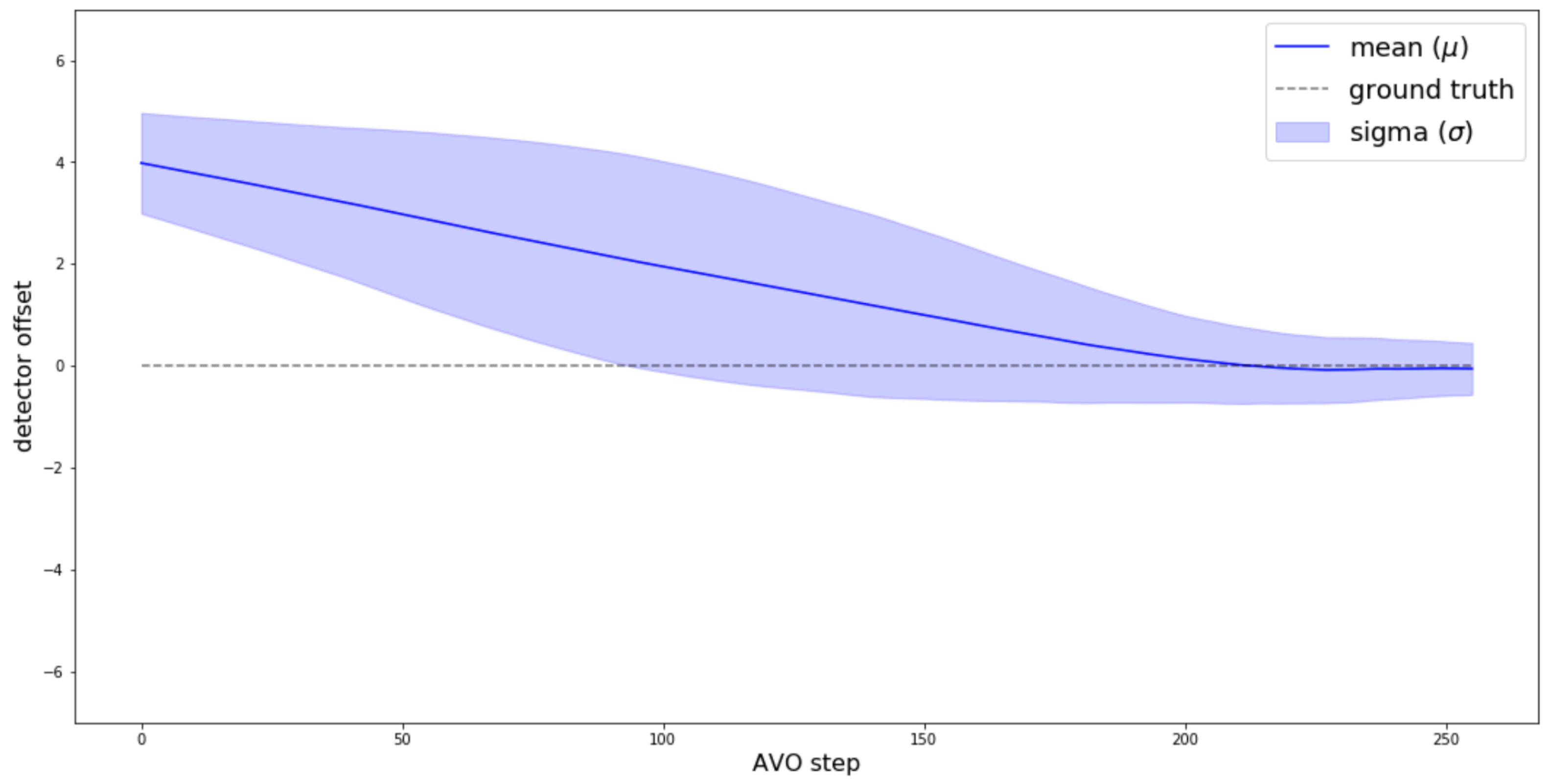
Perhaps a more straight forward example

- http://www.rricard.me/machine/learning/generative/adversarial/networks/keras/tensorflow/2017/04/05/gans-part2.html

Generate your own human faces!

- https://github.com/eparrish64/NICADD-ML-Tutorials/blob/master/Tutorials/mlhep2018/Generating%20Faces%20with%20GANs%20-%20Solution%20-%20Wasserstein.ipynb

Pythia GAN Tutorial



Resetting Up Environment

To setup the environment again (without installing conda and packages)

- source setup_environment.sh

Then start jupyter notebook and access via port forwarding as before

Useful Links

Indico

- https://indico.cern.ch/event/687473/timetable/#20180806.detailed

GitHub

- https://github.com/yandexdataschool/mlhep2018

Jupyter Notebooks

- https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html

Machine Learning Cheat Sheets

- https://ml-cheatsheet.readthedocs.io/en/latest/index.html
- https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning
- https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-science-pdf-f22dc900d2d7

Acknowledgements

All material taken from the Fourth Machine Learning in High Energy Physics Summer School 2018

National Research University Higher School of Economics

University of Oxford

Yandex

Dr. Alexey Artemov (Skoltech, HSE)

Dr. Ekaterina Chernyak (HSE)

Maxim Borisyak (YSDA)

Nikita Kazev (YSDA, HSE)

Dr. Andrey Ustyuzhanin (YSDA, HSE, ICL)

Sergey Uzunyan and the NICADD cluster

Thank you

Backup

Outline

What is Machine Learning?

Machine Learning in HEP

Overtraining

Decision Trees

Tree Ensembles (Tutorial)

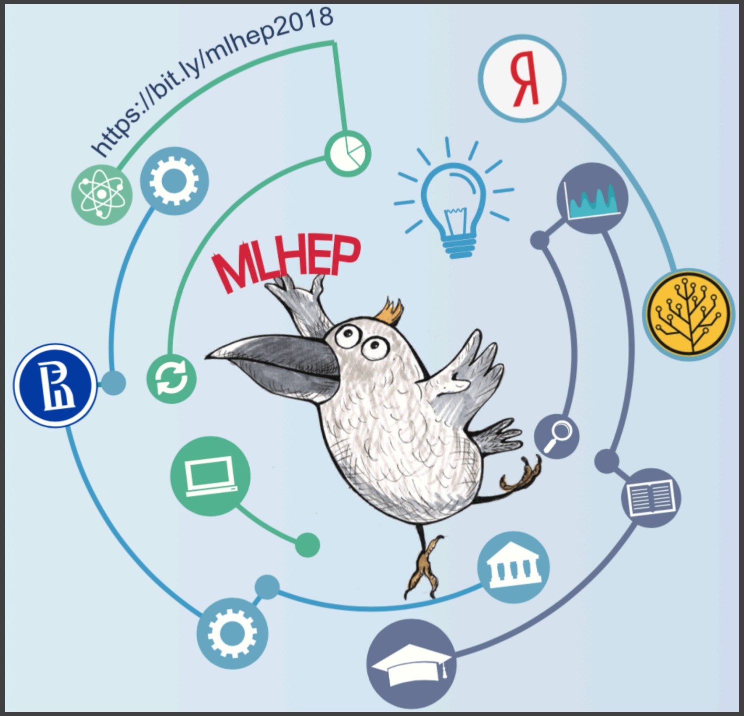
Neural Networks (Tutorial)

Types of Neural Nets

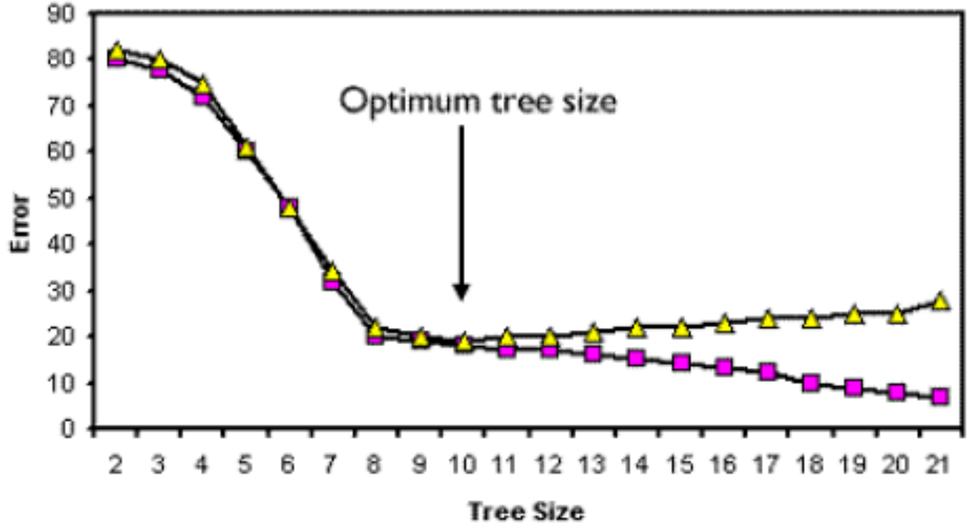
Convolutional Neural Networks (Tutorial)
- Tensorboard

Generative Models

Generative Adversarial Networks (Tutorial)



Decision tree pruning



- Learn a large tree (effectively overfit the training set)
- Detect overfitting via K-fold cross-validation
- Optimize structure by removing least important nodes

Proposed Outline

Machine Learning Theory
- Function approximation
- Loss functions
- Linear regression
- Gradient Descent
- Classification

Machine Learning Machines
- Decision Trees
  - Boosted Decision Trees
- Neural Nets
  - Backpropagation
  - Convolution
  - Adversarial
  - Generative Adversarial
  - Recurrent

Machine Learning Tools
- TMVA
- XGBoost
- AdaBoost
- Scikit Learn
- Tensorflow
- Keras
- PyTorch

Tutorials
- BDT
  - TMVA
    - Developed by Rafael Oreamuno Madriz and Elliot Parrish