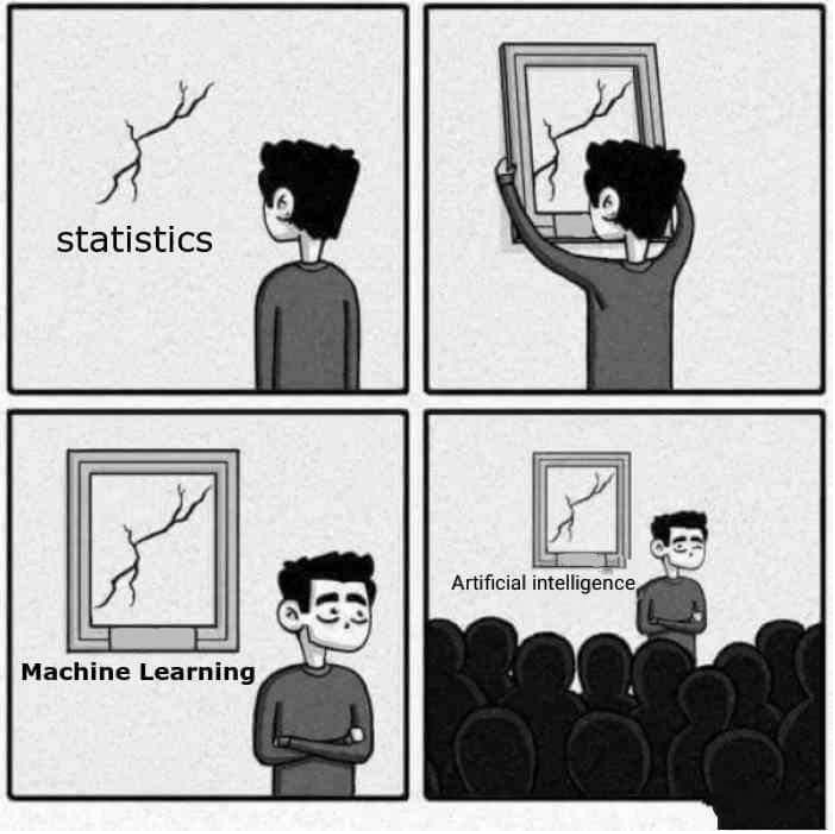
Machine learning

Sergey Korpachev (LPI / MIPT / MEPhI)

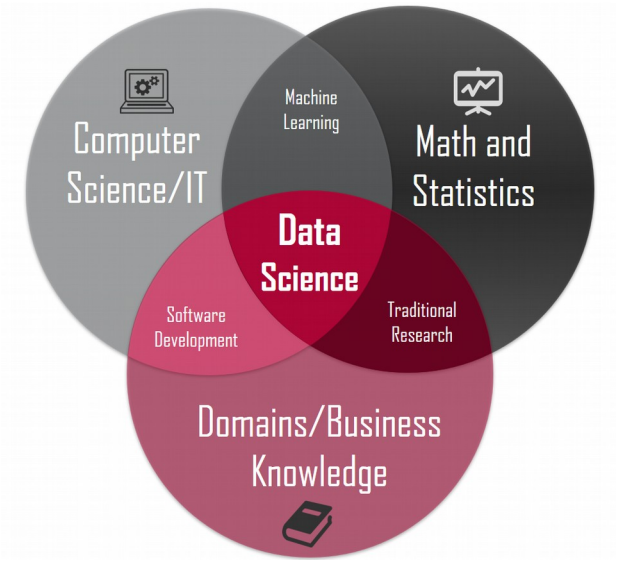
1 April 2020

ML – what is it?



Approximation

$$Y = F(X)$$



ML branches

Supervised learning

Unsupervised learning

Reinforcement learning

Semi-supervised learning

Data in ML

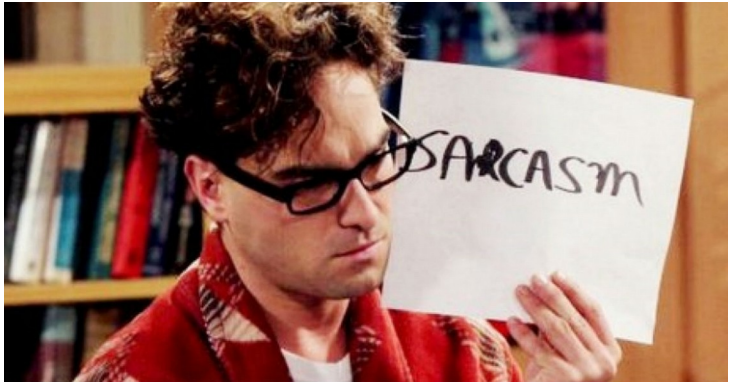
Numbers, matrices, ...

Images

Videos

Text => problems

…..



“Я очень рад тебя видеть”

Spaces in ML

Feature space $\mathbb{X}$
- real
- categorical

Target space $\mathbb{Y}$
- $\mathbb{Y} = \overline{(0,C)} \rightarrow$ classification
- $\mathbb{Y} = \mathbb{R}^d \rightarrow$ regression

Sample $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_\ell)$

Training set $X,Y = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{\ell}$

Model $a(\mathbf{x}) \leftarrow$ predicts target

Loss function $L(a,Y) \leftarrow$ evaluates model's quality

Training

$a(\mathbf{x}) = \operatorname{argmin}_{\mathbf{a} \in \mathbb{A}} L(a,Y)$

Types of the Supervised ML Problems

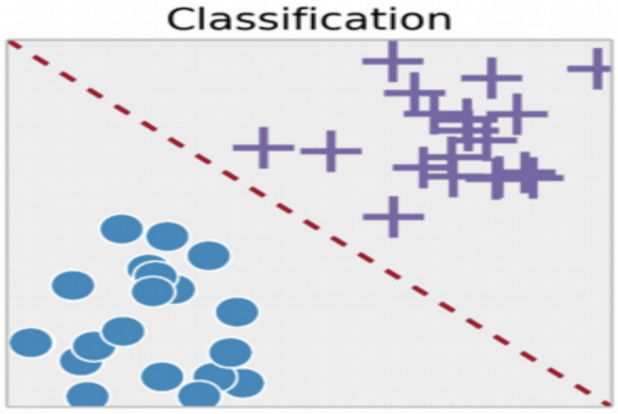
Classification:

$Y = \{-1, +1\}$ - two-class classification

e.g. “signal”, “background”

$Y = \{1, 2, \ldots, M\}$ - multi-class classification

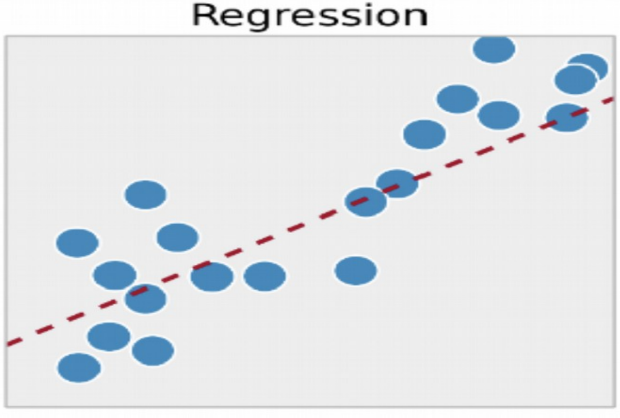
e.g. particle ID



Regression:

$Y = \mathbb{R}$ or $Y = \mathbb{R}^m$

e.g. reconstructed subatomic particle kinematics



grey area:

classification may be in terms of probabilities

regression may be in terms of e.g. integers

Supervised

C vs R

? cat, dog or muffin

? relevant or spam

? disease or not

? good or bad

- rent price
- temperature
- annual profit
- driving time

Supervised

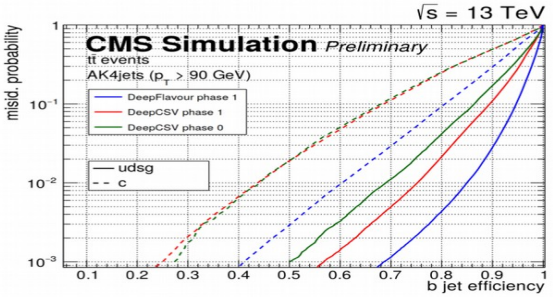
Classification in HEP

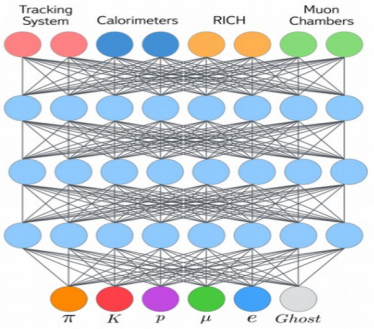
? b, c or ud jet

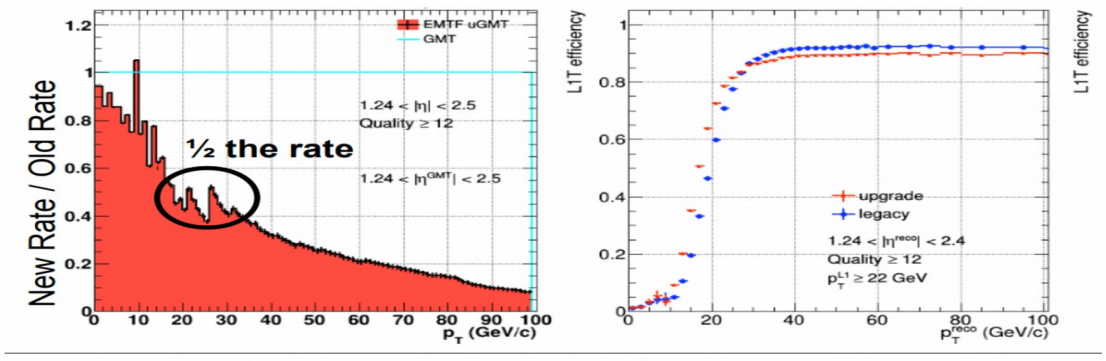
? π, K, or μ particle

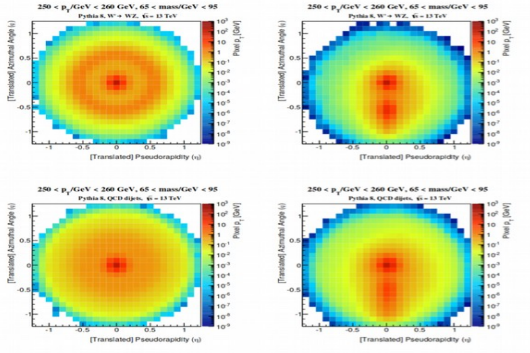
? $t\bar{t}$ or QCD event

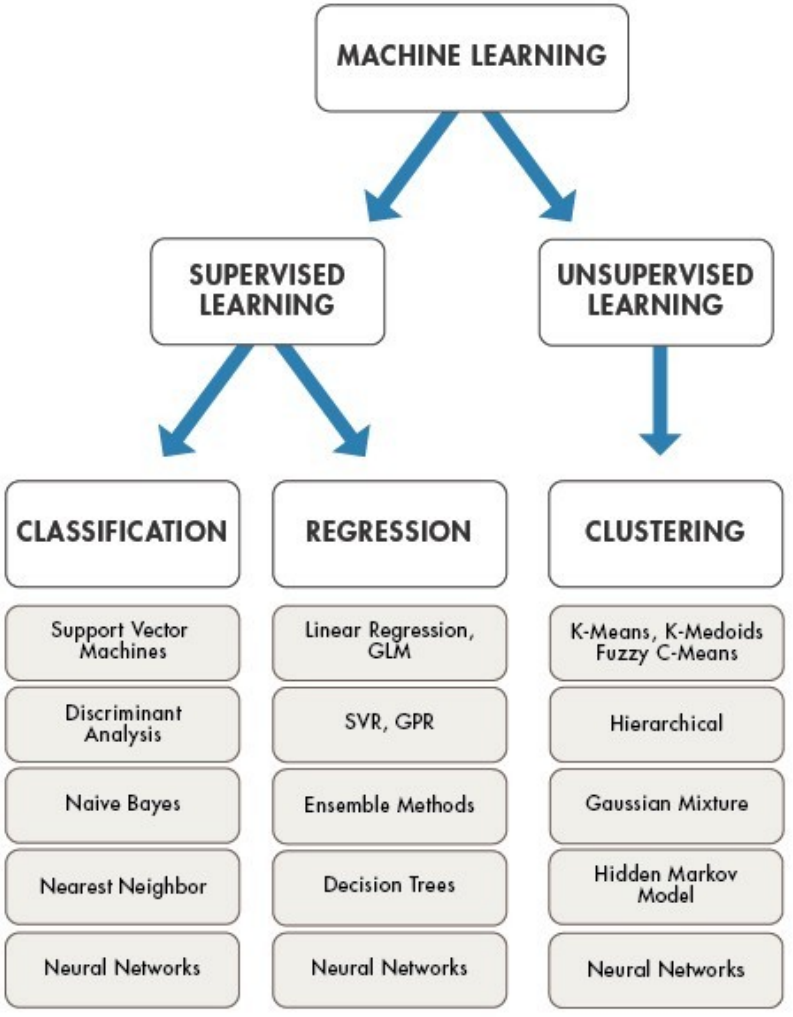
? select or reject trigger candidate



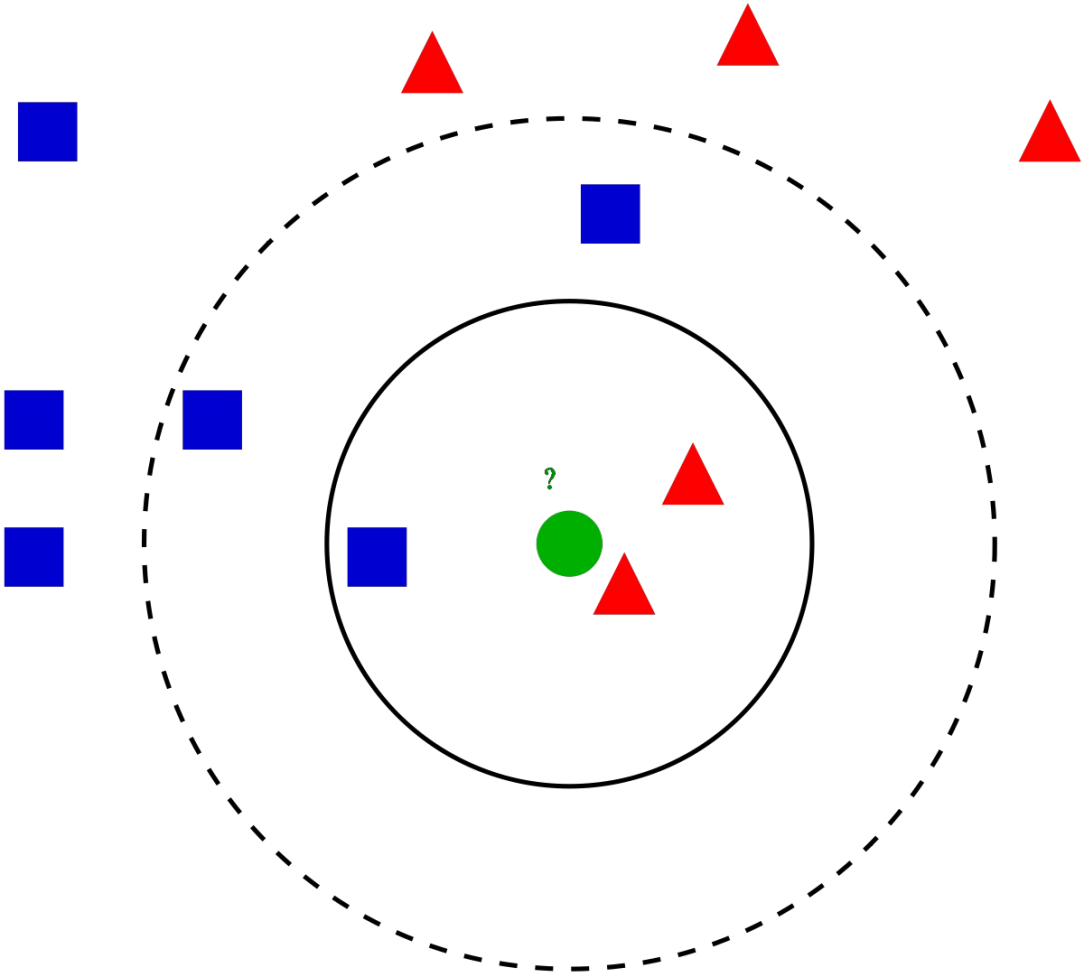


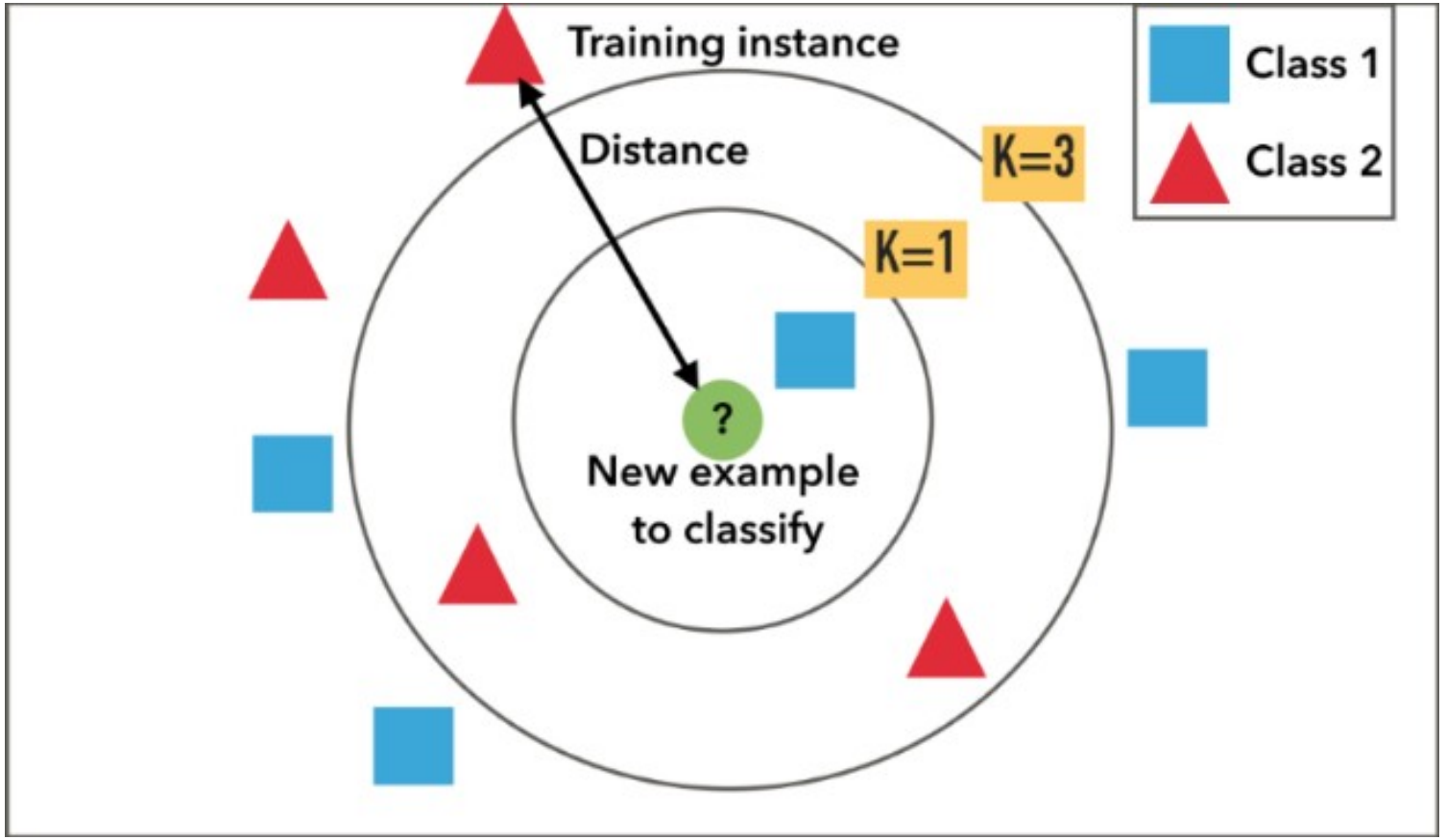


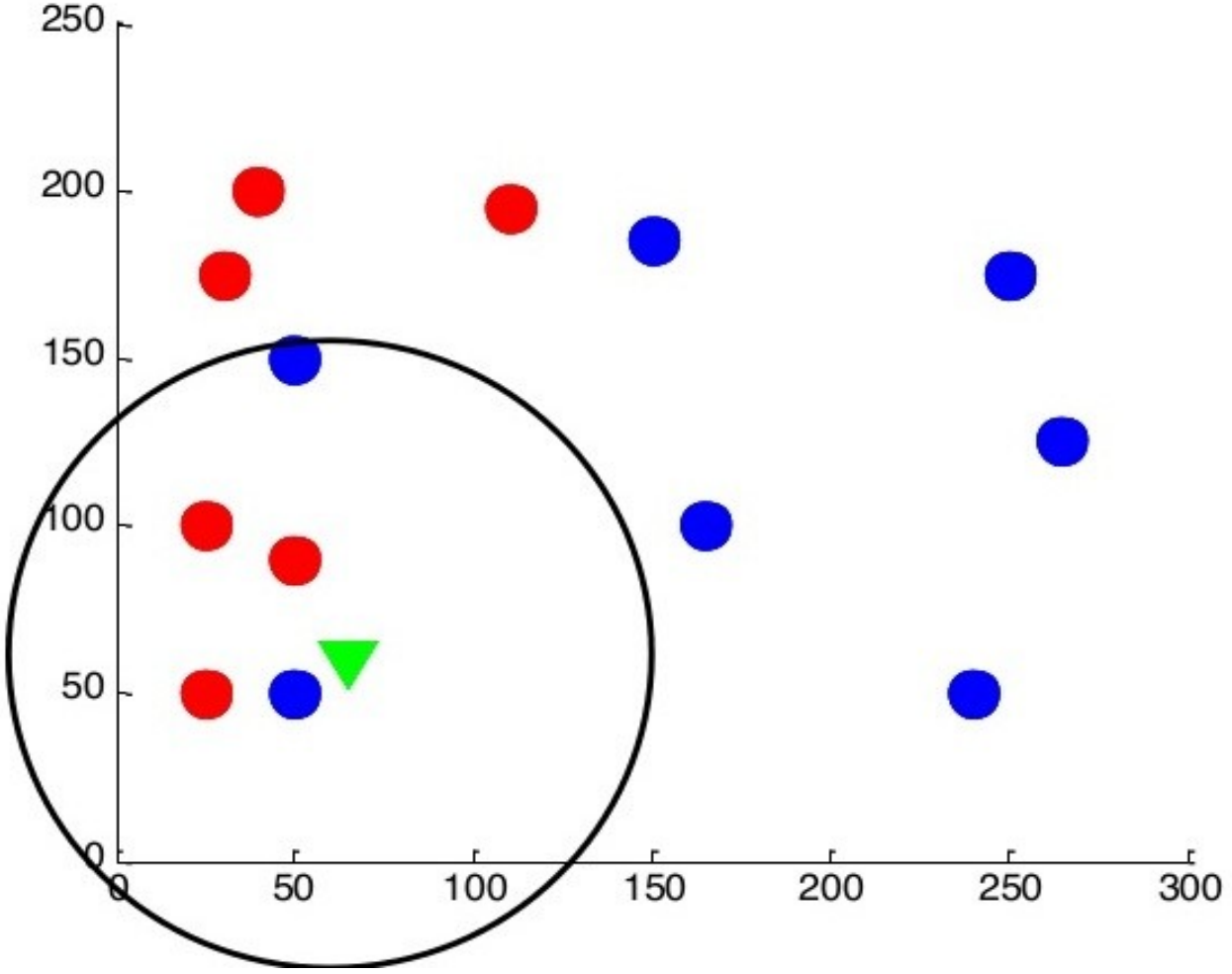


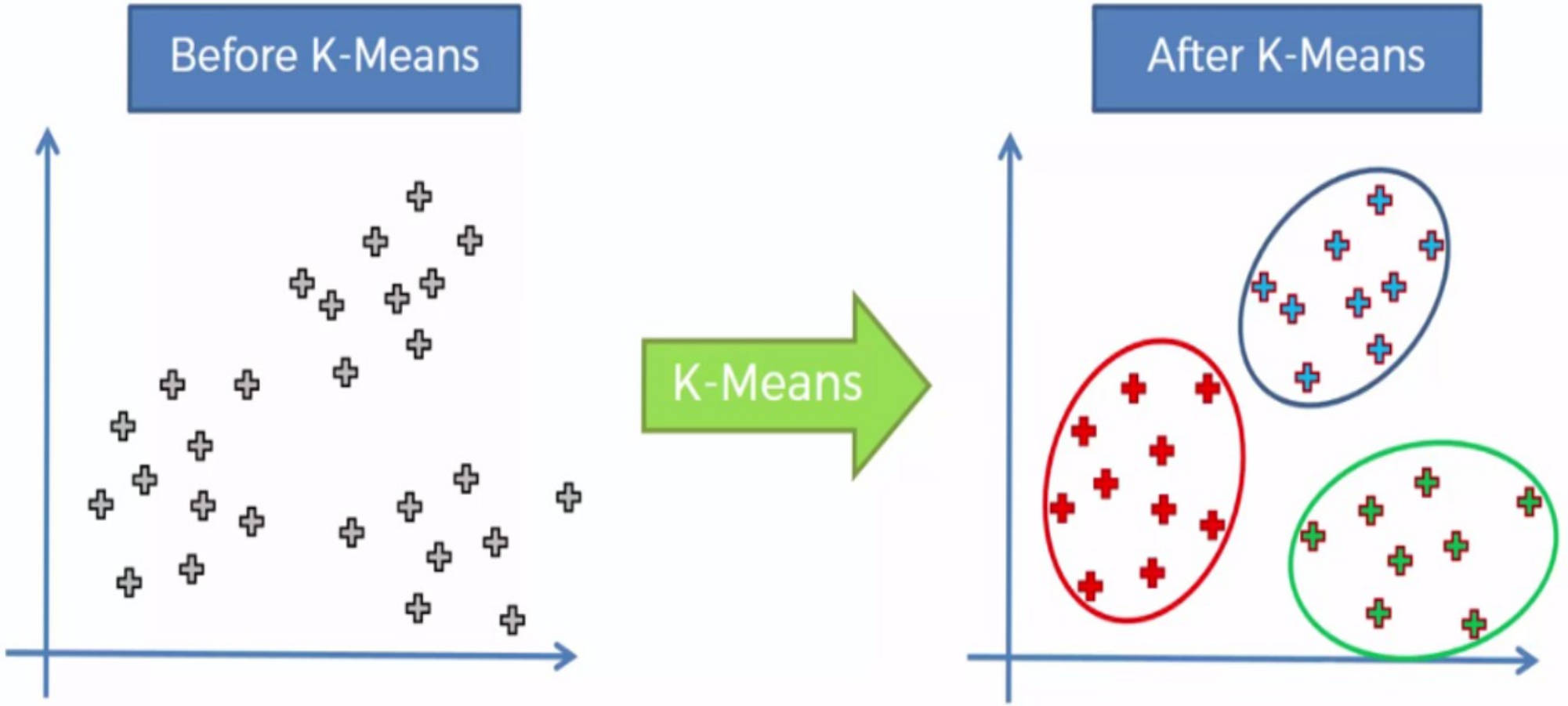

MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
CLASSIFICATION
REGRESSION
CLUSTERING
Support Vector Machines
Discriminant Analysis
Naive Bayes
Nearest Neighbor
Neural Networks
Linear Regression, GLM
SVR, GPR
Ensemble Methods
Decision Trees
Neural Networks
K-Means, K-Medoids Fuzzy C-Means
Hierarchical
Gaussian Mixture
Hidden Markov Model
Neural Networks


10 Algorithms Every Machine Learning Enthusiast Should Know
1 Logistic Regression
2 K-Nearest Neighbours
3 Naive Bayes
4 Support Vector Machines
5 Random Forest
6 Linear Regression
7 Neural Network
8 PCA
9 K-Means Clustering
10 Linear Discriminant Analysis
Prime CLASSES
+91 9703709221
www.primeclasses.in

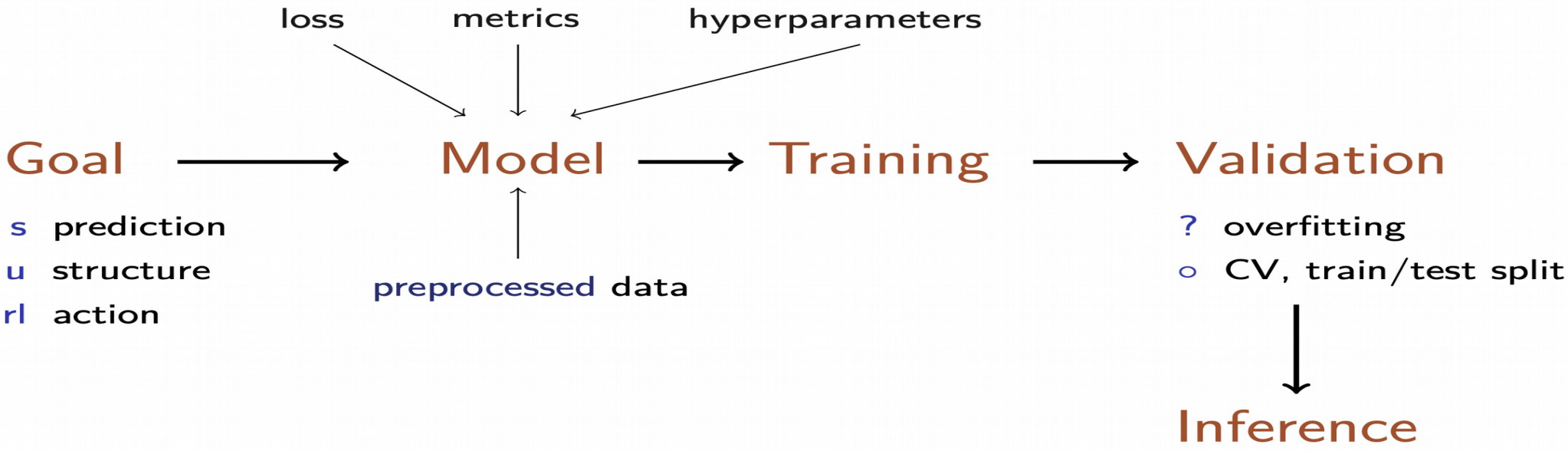



Training instance
Distance
K=3
K=1
?
New example
to classify
Class 1
Class 2

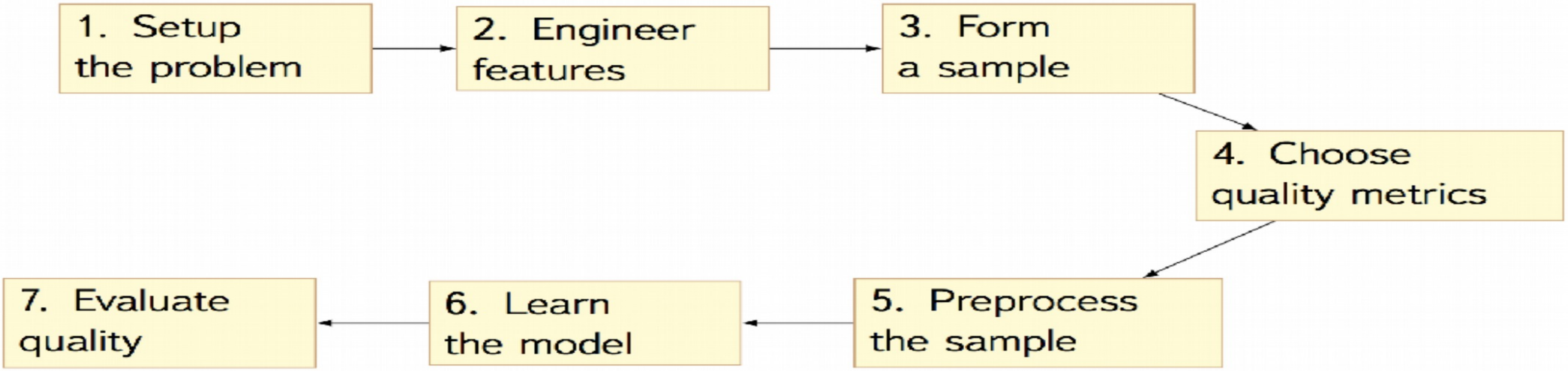

250
200
150
100
50
0
0
50
100
150
200
250
300


Before K-Means
K-Means
After K-Means

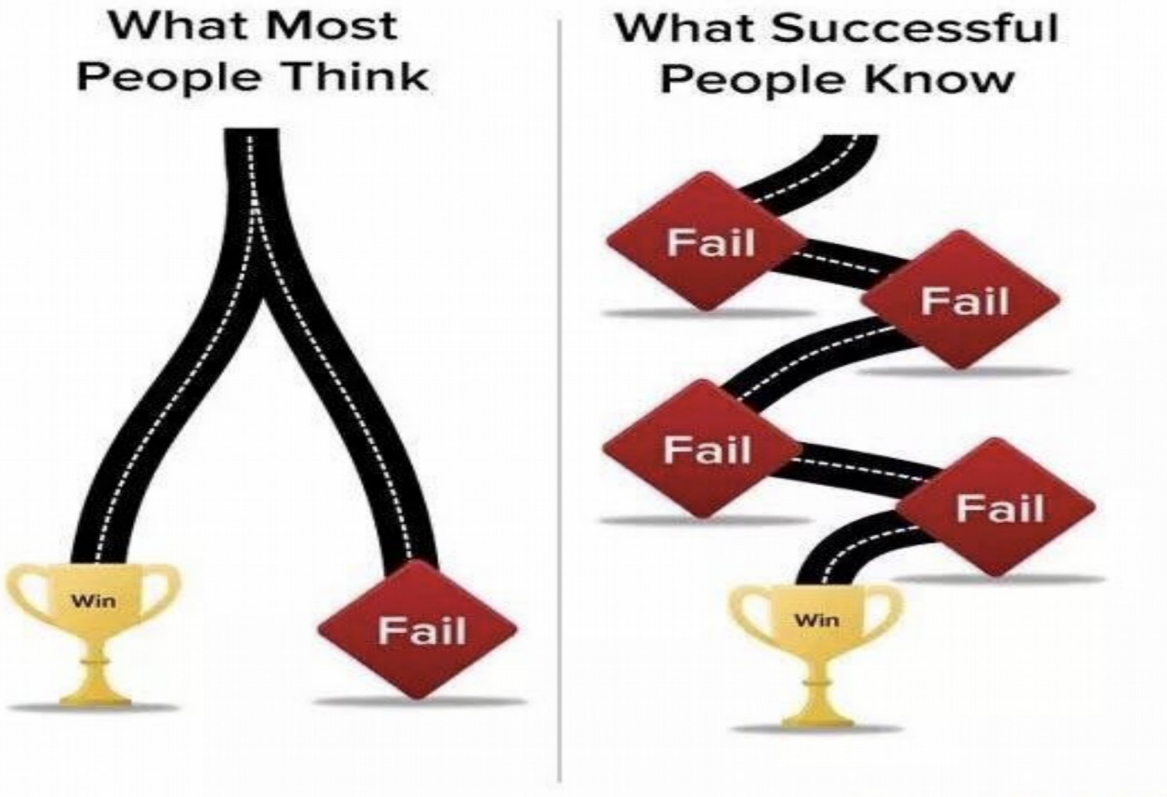
Pipeline


loss
metrics
hyperparameters
Goal
Model
Training
Validation
s prediction
u structure
rl action
preprocessed data
? overfitting
o CV, train/test split
Inference

Machine Learning Pipeline



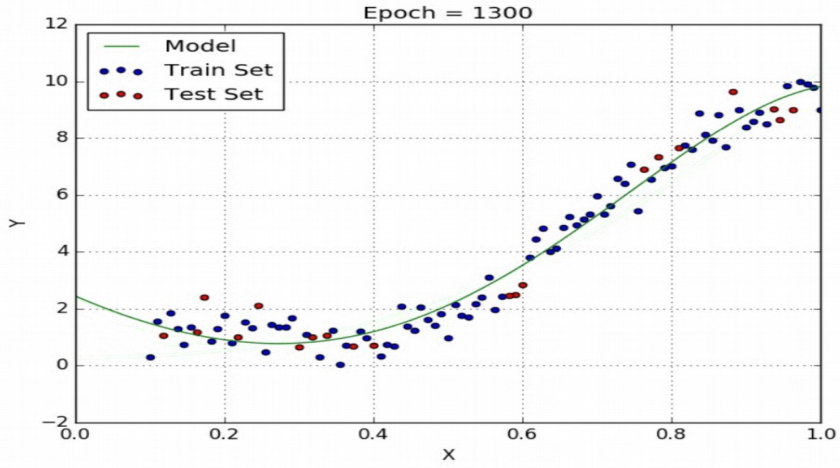
Machine Learning Pipeline



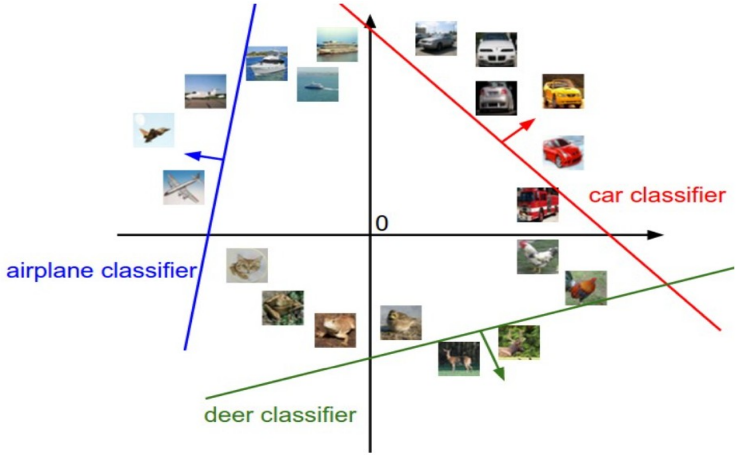
Linear models

Regression

- $\mathbb{Y} = \mathbb{R}^d$
- $a(x) = w^T x$
- L(a,Y):
  - MSE (L2) $= \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - a(x_i))^2$
  - RMSE $= \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - a(x_i))^2}$
  - MAE (L1) $= \frac{1}{\ell}\sum_{i=1}^{\ell}|y_i - a(x_i)|$
  - $R^2 = 1 - \frac{\sum_{i=1}^{\ell}(y_i - a(x_i))^2}{\sum_{i=1}^{\ell}(y_i - \overline{y})^2}$
- $L(a,Y) \to \min_w$ $\Longleftarrow$ how can we do that?



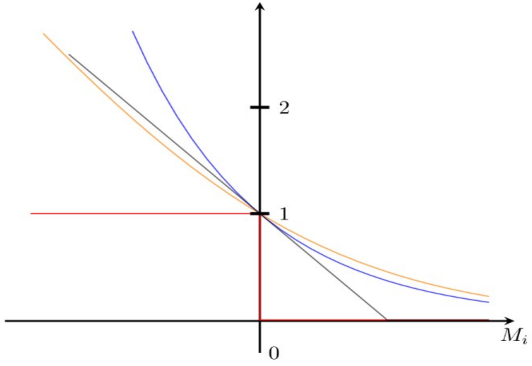
Classification



- $\mathbb{Y} = \{-1, 1\}$
- $a(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$
- $L(a, Y)$:
  - $\text{Accuracy} = \frac{1}{\ell}\sum_{i=1}^{\ell}[y_i = \text{sign}(\mathbf{w}^T\mathbf{x})]$

? $\frac{1}{\ell}\sum_{i=1}^{\ell}[y_i = a(\mathbf{x}_i)] \to \min_{\mathbf{w}}$

! Optimize upper bound $\tilde{L}(a, Y)$
  - Log loss
  - Hinge loss
  - ...

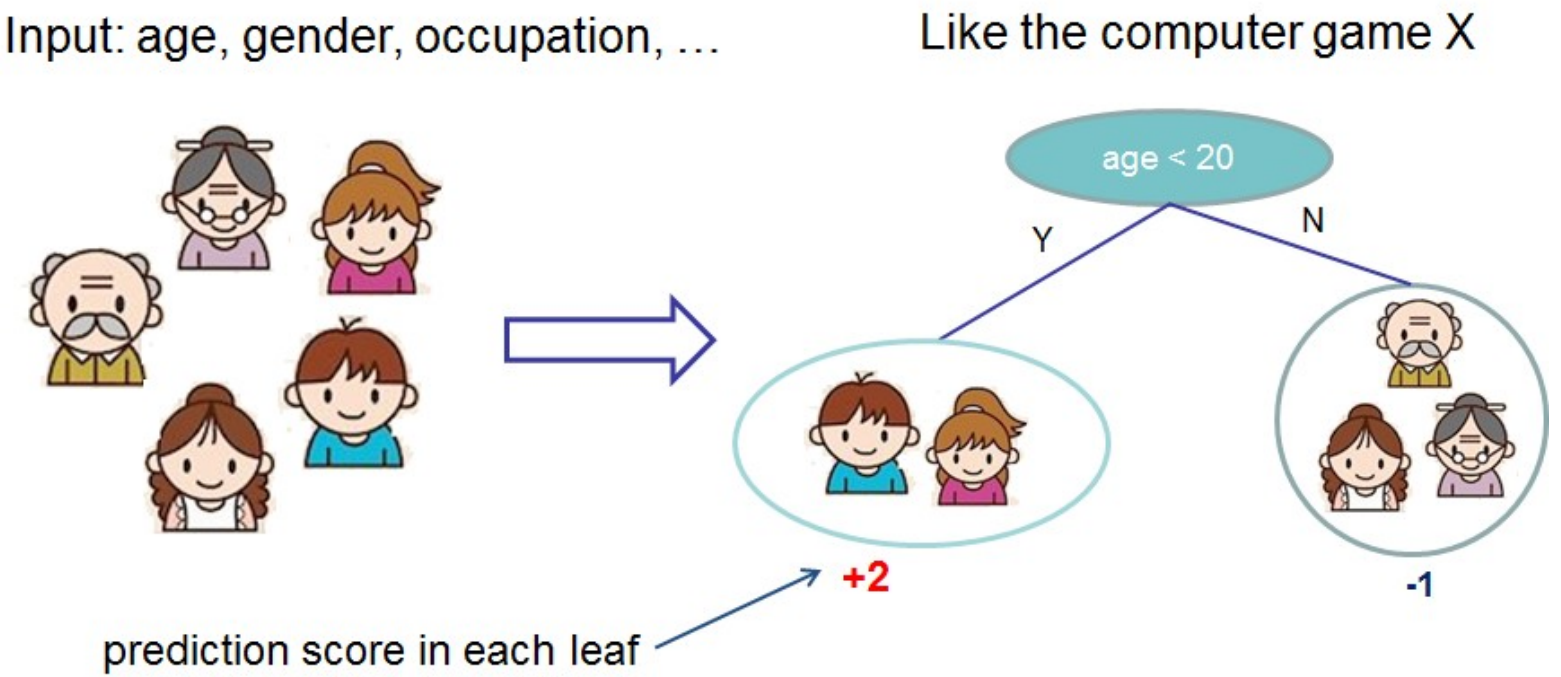


Linear models

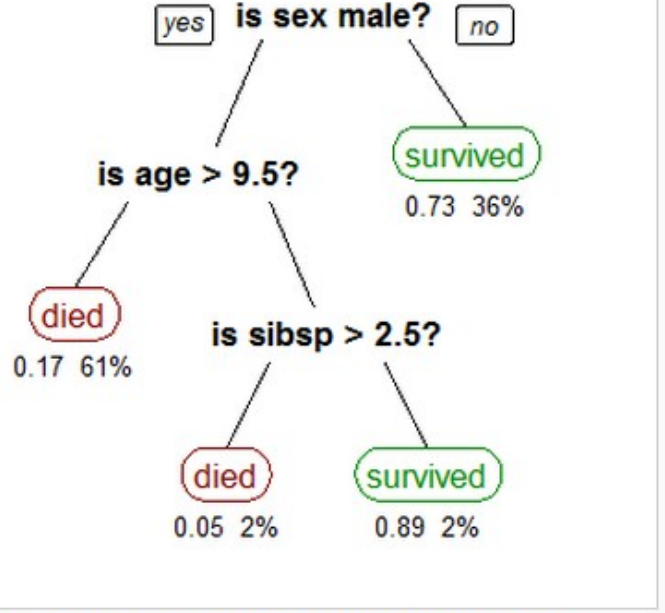
pros & cons

+ Interpretable
+ Unique solution exists
+ Nice statistical properties
+ Fast
+ Perform well with lots of features
+ OK with little data
- Don't capture non-linear correlations
- Require feature linear independence

Boosted decision trees (BDT)



Boosted decision trees (BDT)



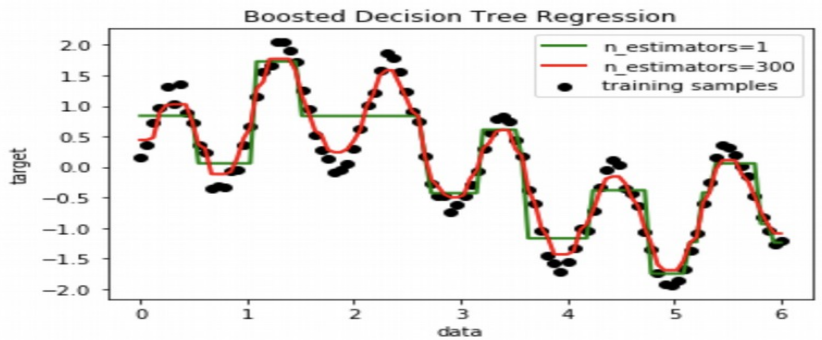
A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

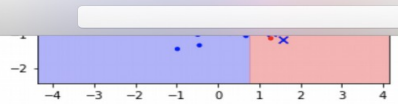
```
# Create the dataset
rng = np.random.RandomState(1)
X = np.linspace(0, 6, 100)[:, np.newaxis]
y = np.sin(X).ravel() + np.sin(6 * X).ravel() + rng.normal(0, 0.1, X.shape[0])

# Fit regression model
regr_1 = DecisionTreeRegressor(max_depth=4)

regr_2 = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4),
                           n_estimators=300, random_state=rng)

regr_1.fit(X, y)
regr_2.fit(X, y)

# Predict
y_1 = regr_1.predict(X)
y_2 = regr_2.predict(X)
```

```
Automatically created module for IPython interactive environment
```

In [7]:

```
# Plot the results
plt.figure()
plt.scatter(X, y, c="k", label="training samples")
plt.plot(X, y_1, c="g", label="n_estimators=1", linewidth=2)
plt.plot(X, y_2, c="r", label="n_estimators=300", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Boosted Decision Tree Regression")
plt.legend()
plt.show()
```

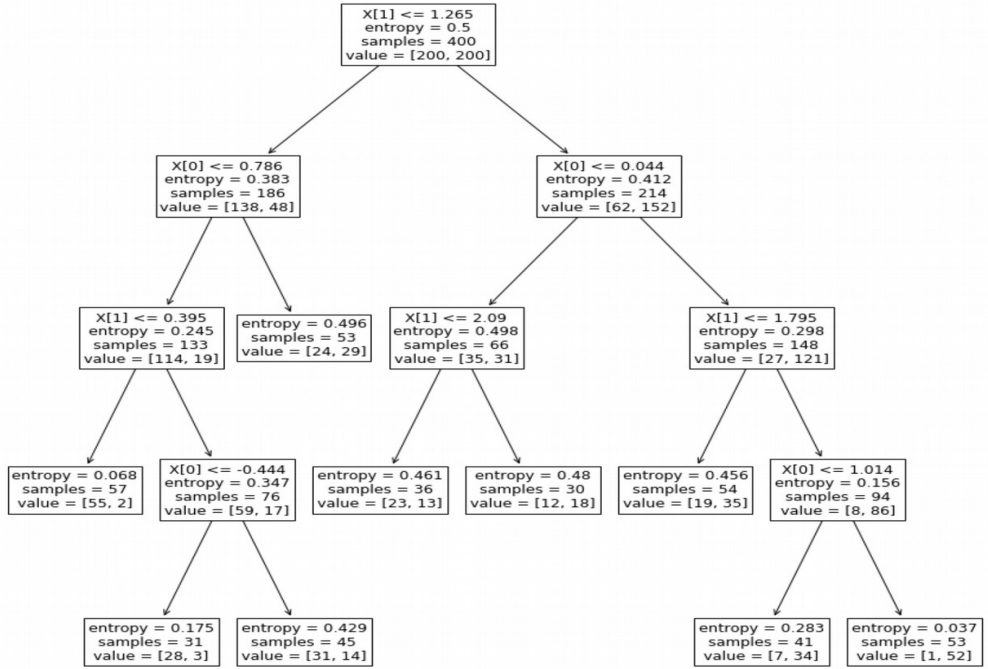




As promissed, trees can be visualized

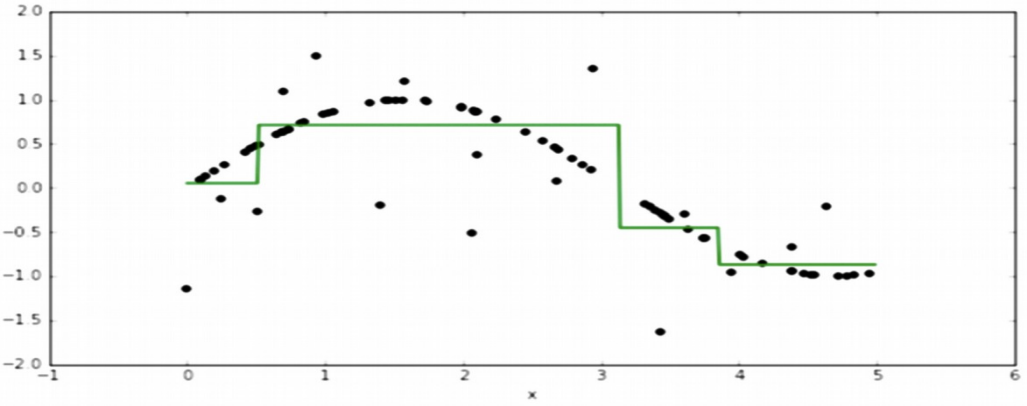
In [0]:
```
import sklearn.tree
```

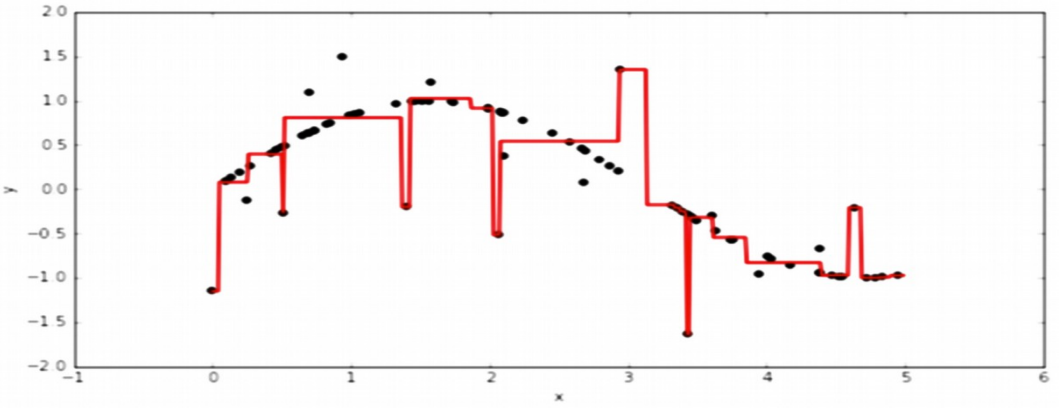
In [0]:
```
fig, ax = plt.subplots(figsize=(15, 15))
sklearn.tree.plot_tree(clf, ax=ax);
```



Trees

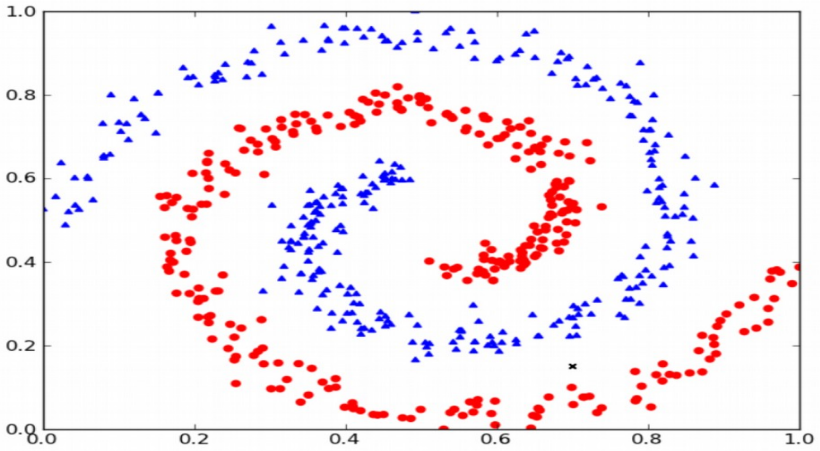
Regression

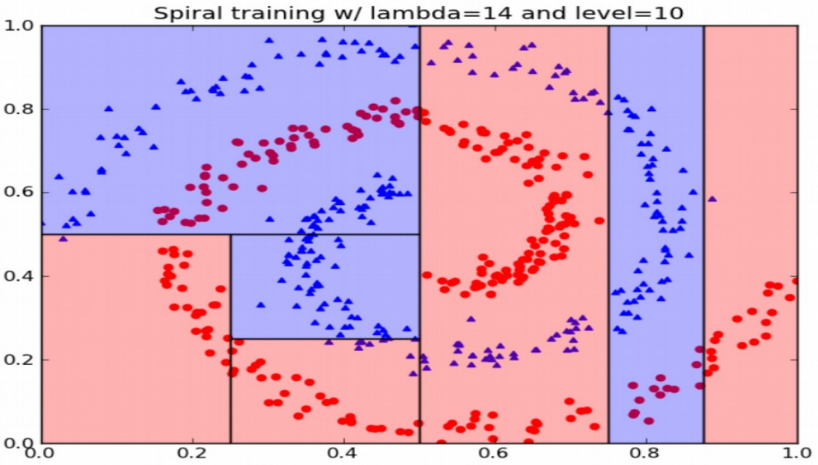




Trees

Classification



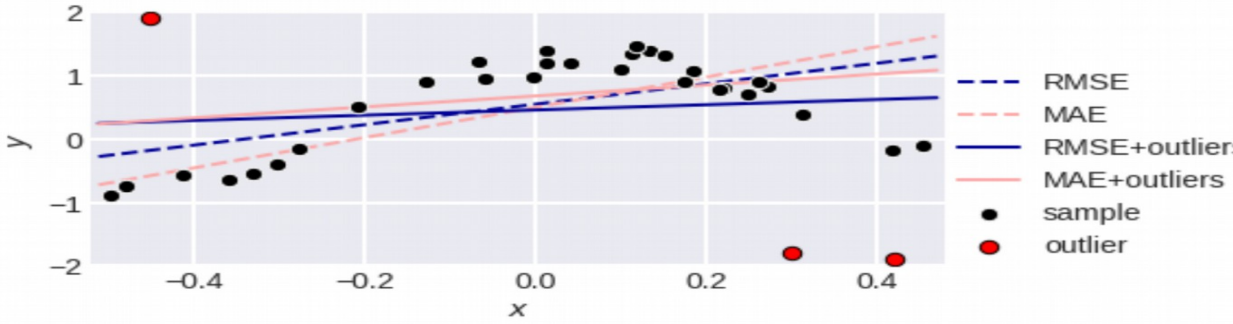


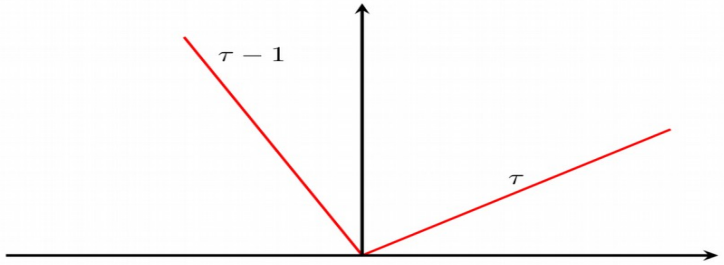
nice one on metrics [1], [2]

Metrics

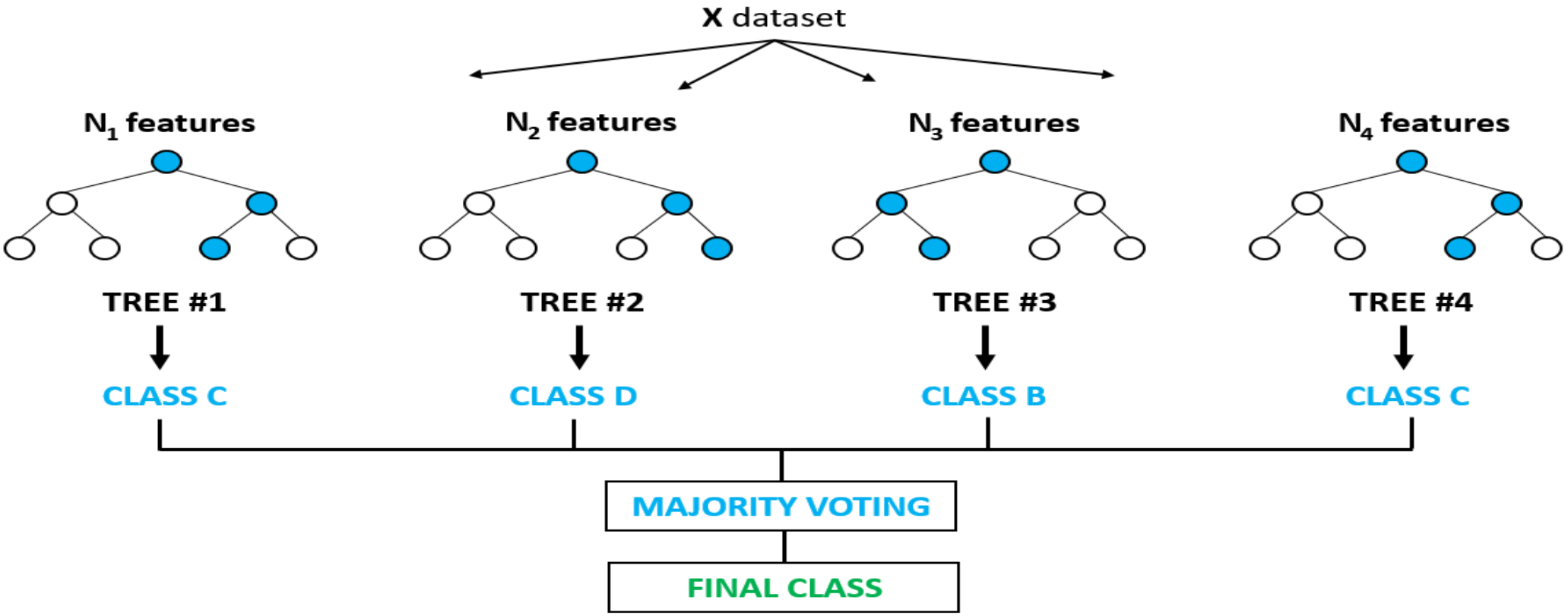
Regression

- MSE $= \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - a(x_i))^2$
  - sensitive to outliers
  - easy to optimize
- MAE $= \frac{1}{\ell}\sum_{i=1}^{\ell}|y_i - a(x_i)|$
  - robust to outliers
  - gradient problems (no)
- $R^2 = 1 - \frac{\sum_{i=1}^{\ell}(y_i - a(x_i))^2}{\sum_{i=1}^{\ell}(y_i - \overline{y})^2}$
  - explained variance
  - comparison with naive mean model
- Logloss



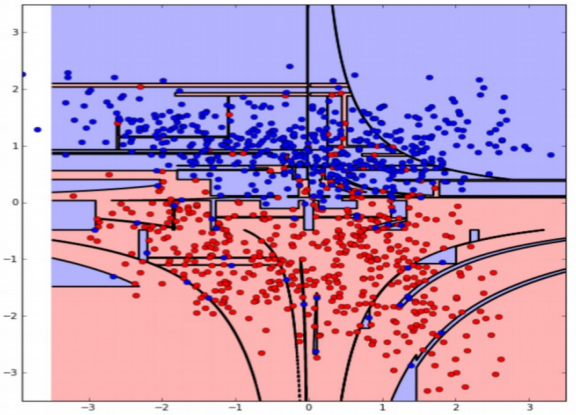


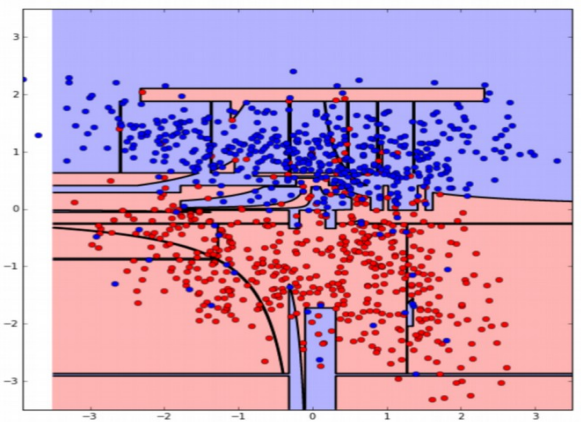
Random Forest



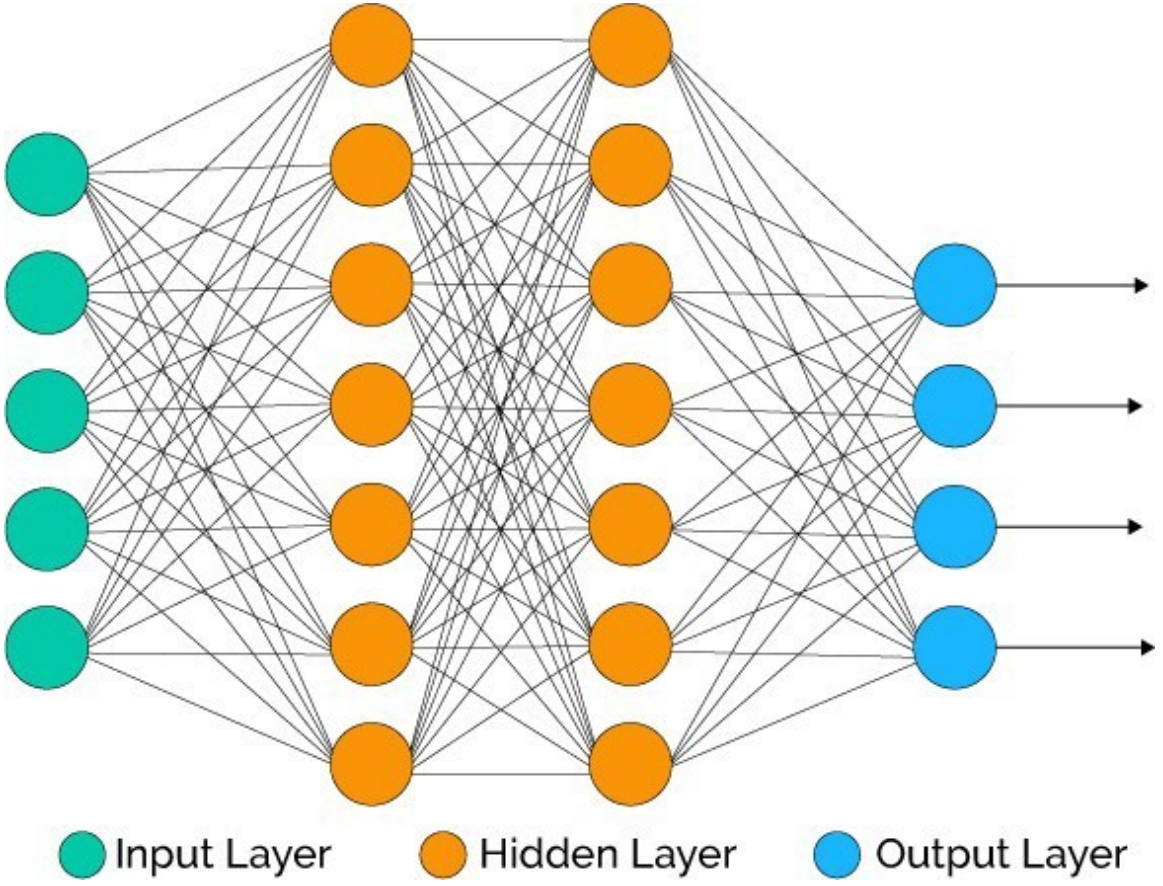
Trees [recap]

+ interpretable
+ categorical features
+ don't care about feature linear dependence
+ no need to scale inputs
- weak learners
- can overfit
- large variance with sample variation

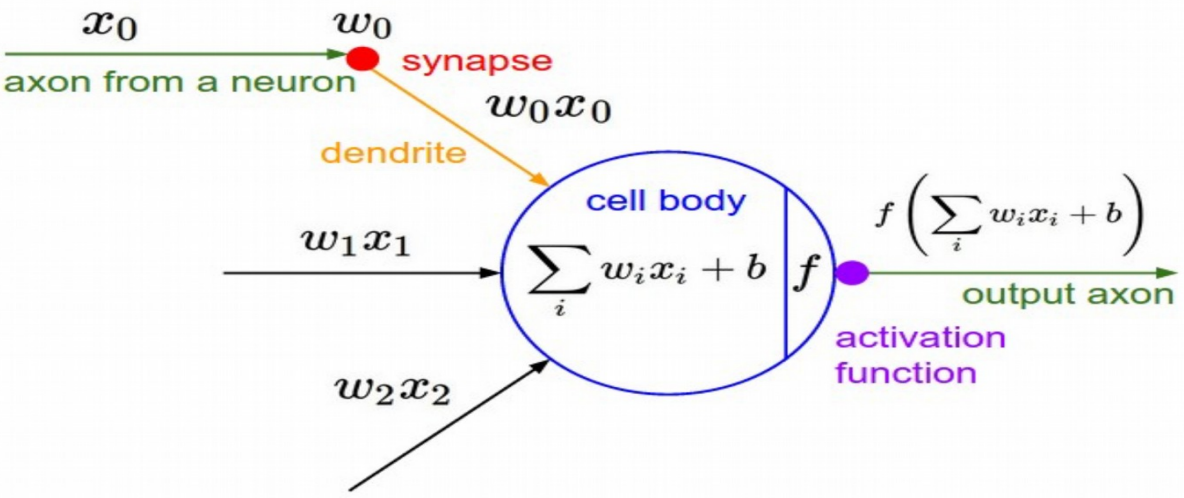




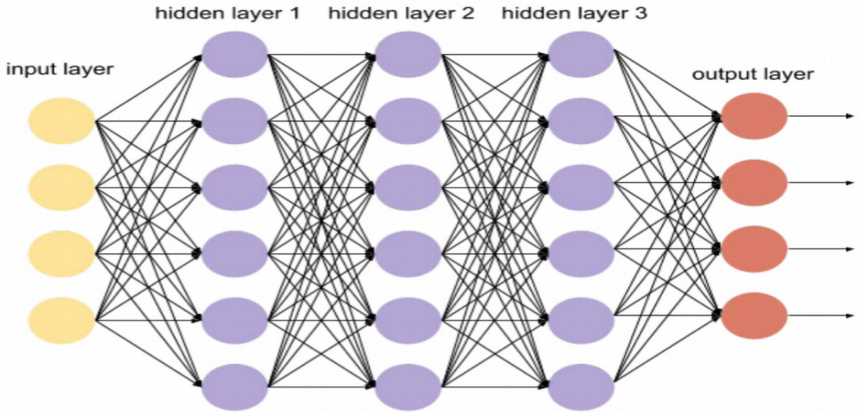
Artificial Neural Network (ANN)



Neuron


x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$

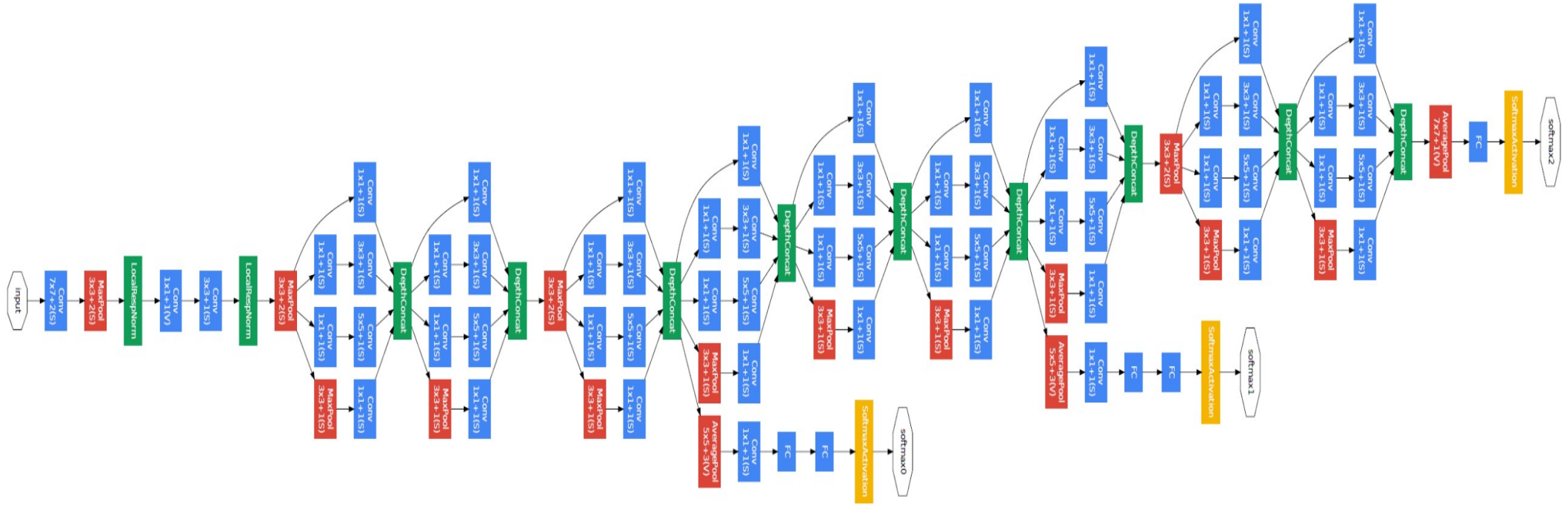
Neural network



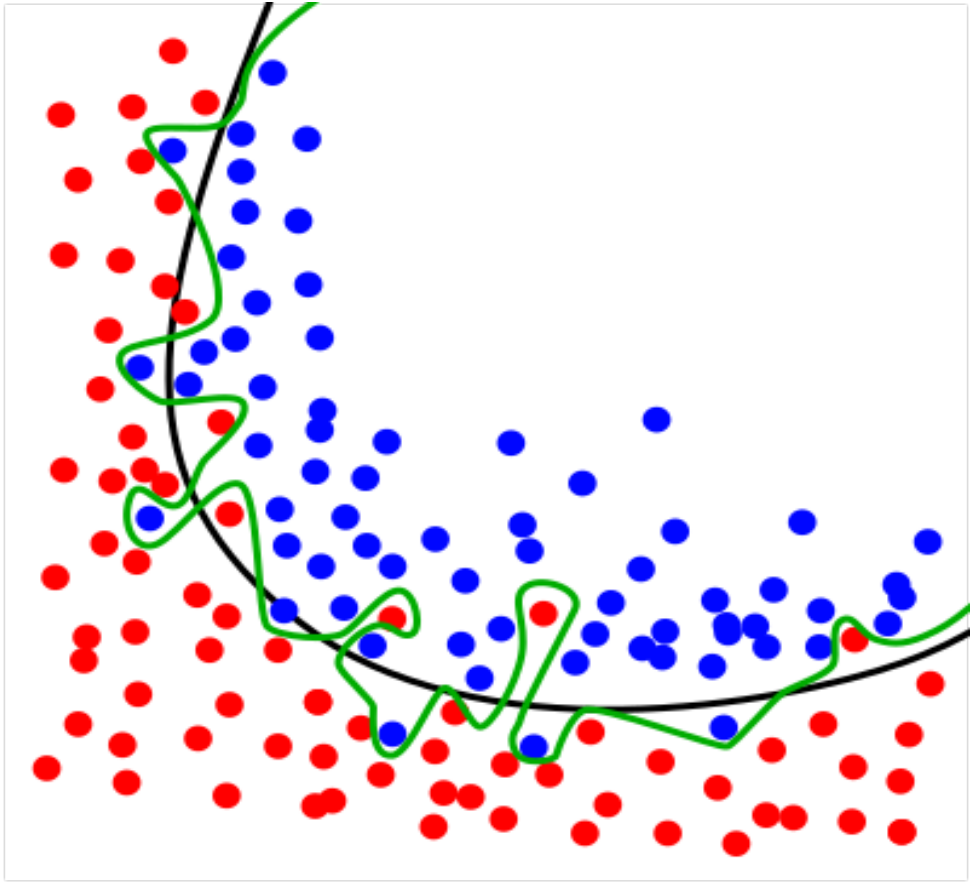
$$h_t = f(W h_{t-1} + b)$$

$$a(x) = f_3(W_3 f_2(W_2 f_1(W_1 f_0(W_0 x + b_0) + b_1) + b_2) + b_3)$$

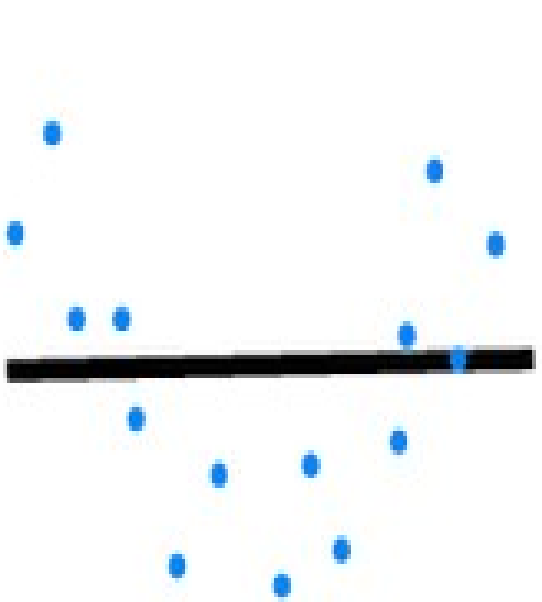
GoogLeNet Network



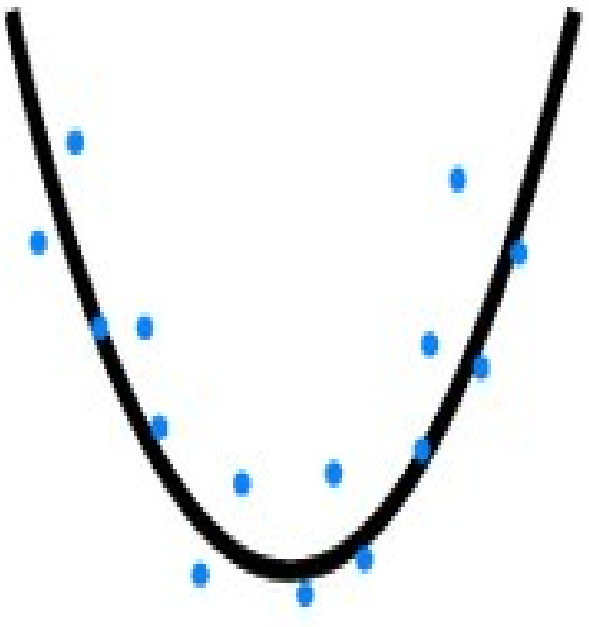
Overtraining



Overtraining



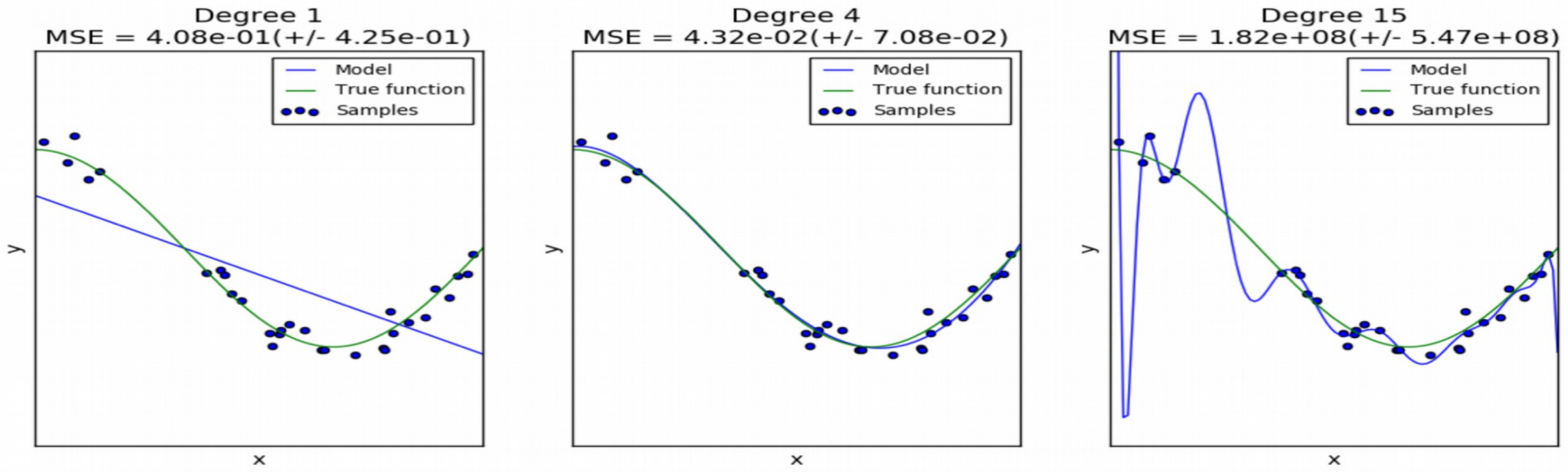
Underfitting



Desired

Overfitting

Overfitting



История про танки

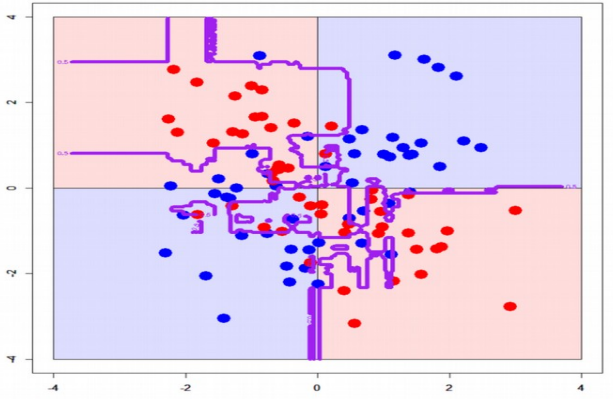
Классификатор: есть танки на снимке или нет

Overfitting

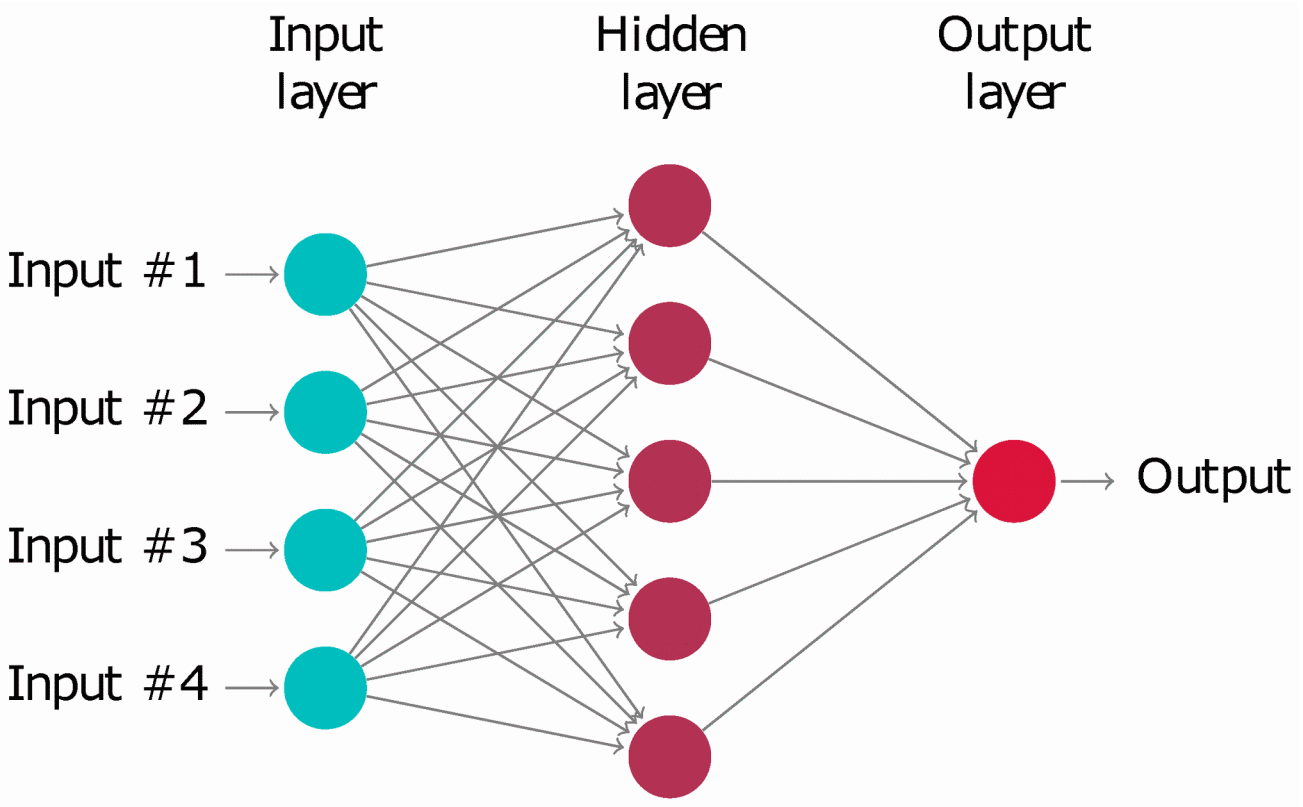
- Often looks like
  - $a(x) = 0.5 + 12458922x + 43983740x^2 + \ldots + 2740x^9$

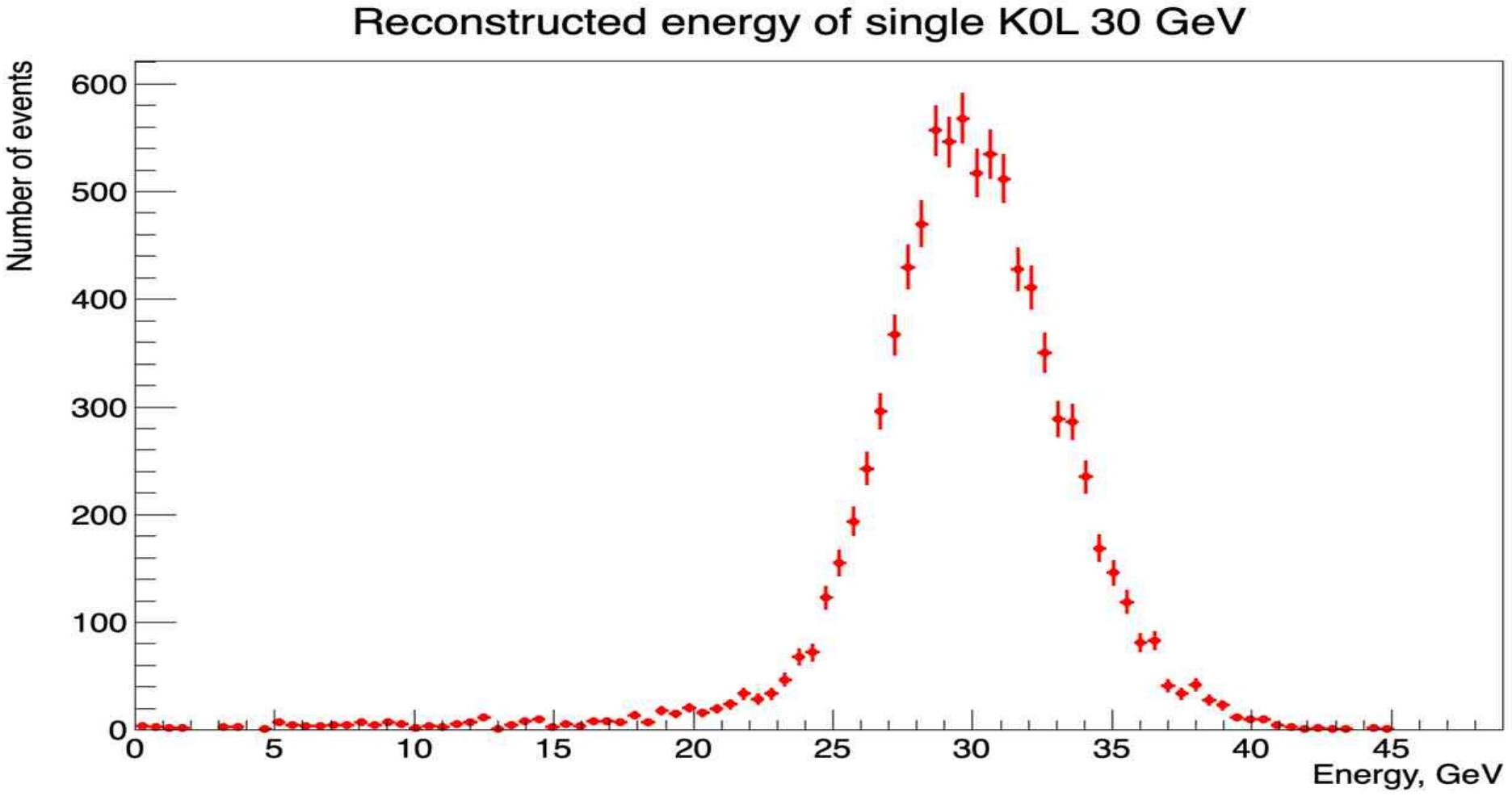
  $\rightarrow$ regularization
- Generalization:
  - underfitting - bad on train, bad on unseen
  - overfitting - excellent on train, bad on unseen

  $\rightarrow$ validate on hold-out data

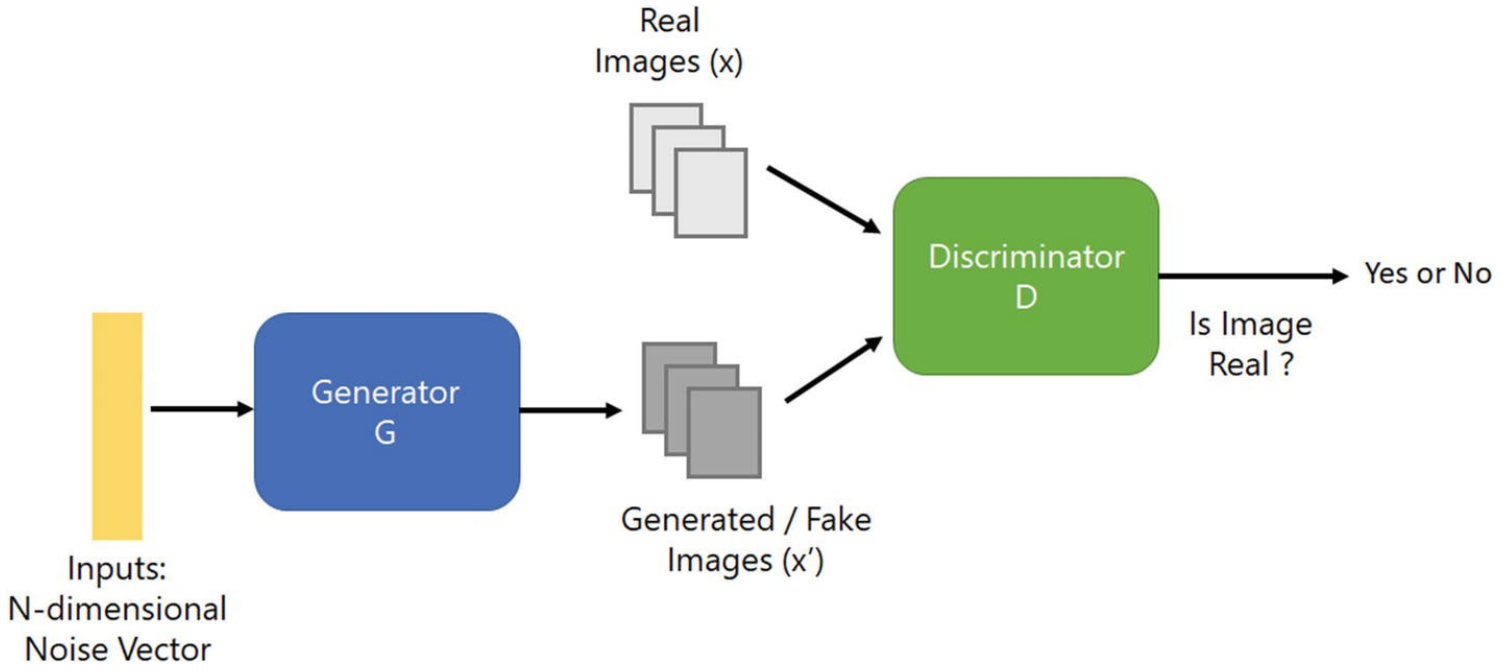


My work with ANN




Reconstructed energy of single K0L 30 GeV
Number of events
600
500
400
300
200
100
0
0
5
10
15
20
25
30
35
40
45
Energy, GeV

Generative adversarial network (GAN)



Generative Models in Cosmology

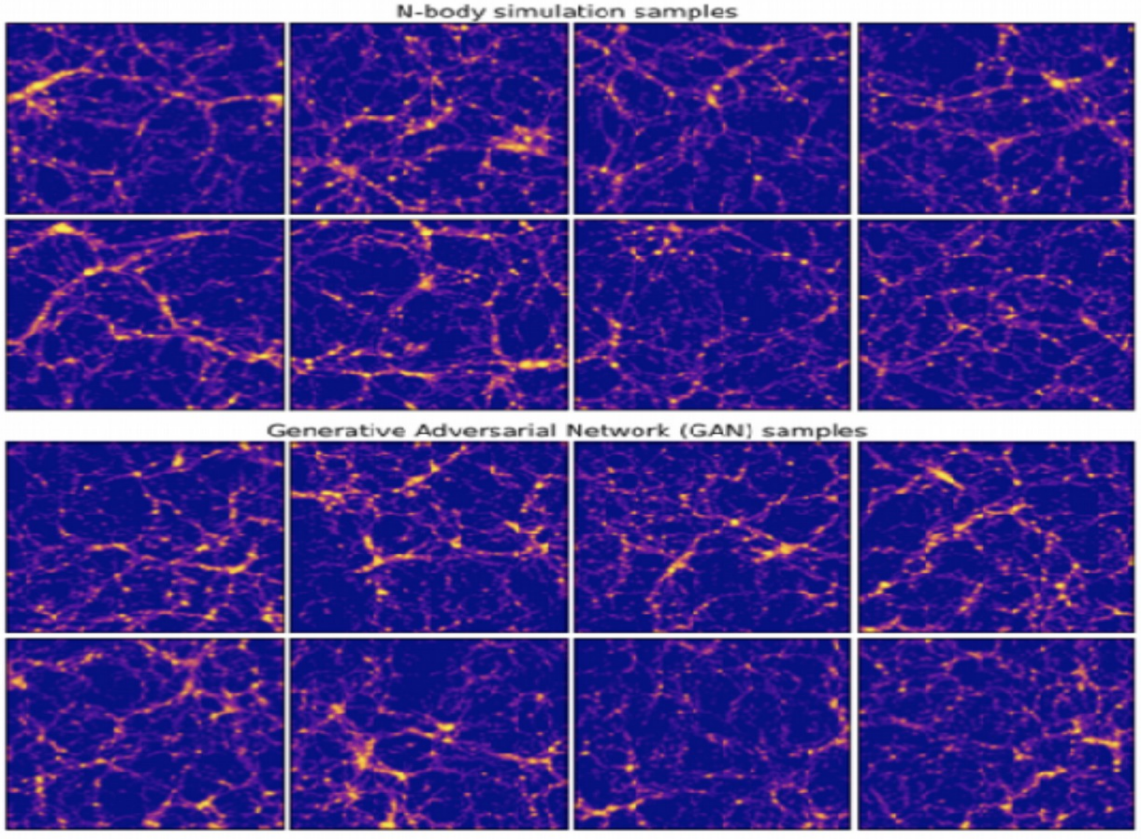
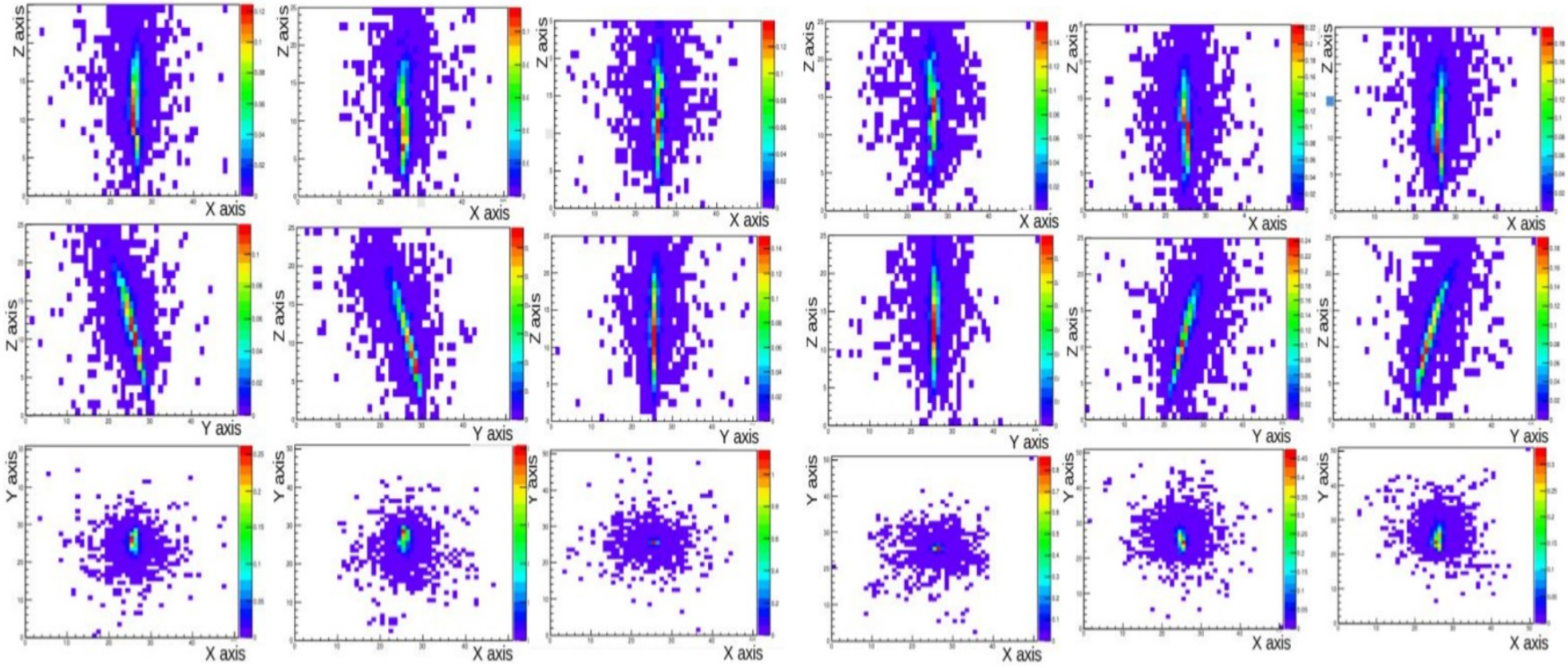


Figure 3: Samples from N-body simulation (top two rows) and from GAN (bottom two rows) for the box size of 100 Mpc. In this figure, transformation S1 with $k = 7$ was applied.

Rodríguez et al,
arXiv:1801.09070

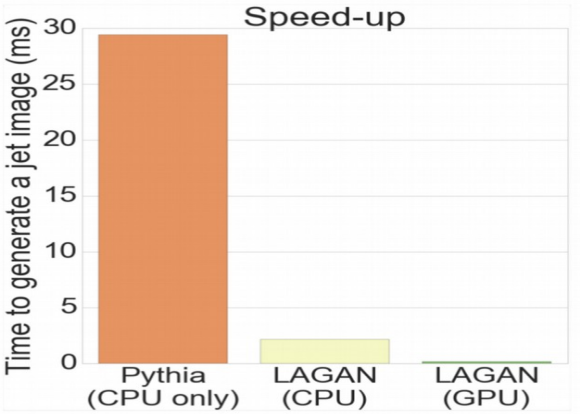

Z axis
X axis
Y axis
G4
GAN
G4
GAN
G4
GAN

GAN

motivation

- Really fast simulation

  in fact $10^2 \div 10^5$ faster than Geant
- Save space
- Rare events



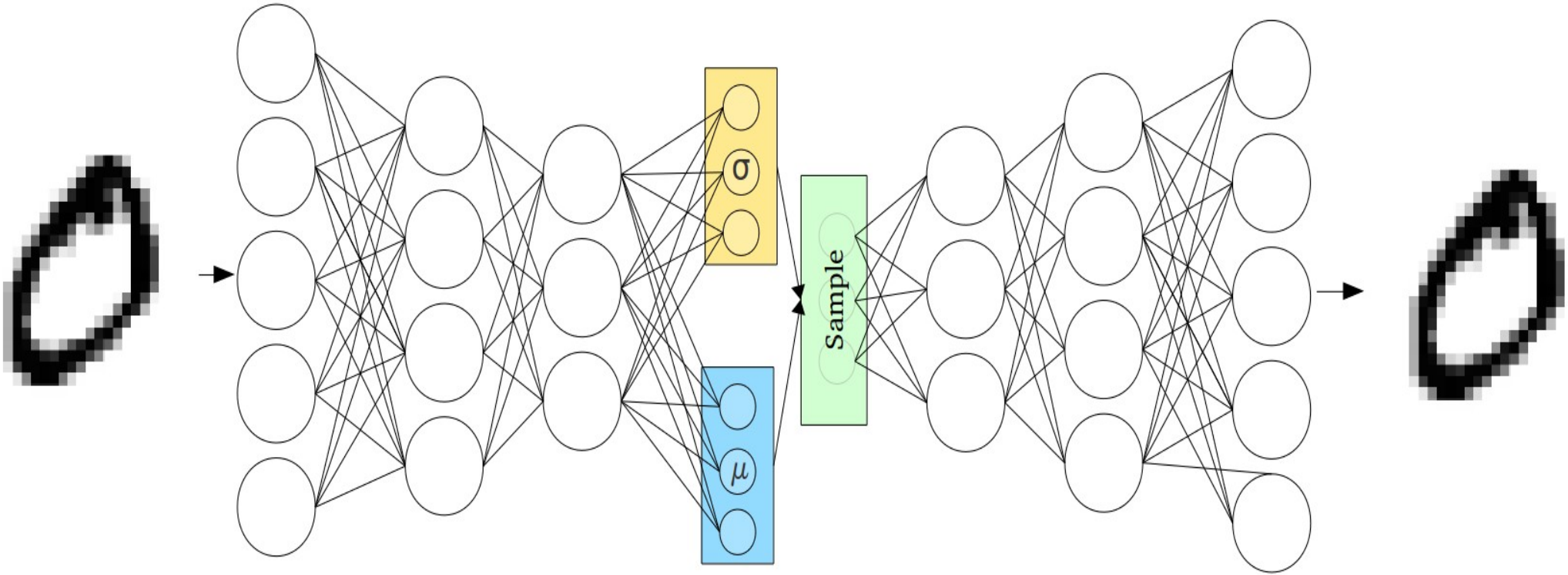
Generation Method	Hardware	Batch Size	milliseconds/shower
GEANT4	CPU	N/A	1772
CALOGAN	CPU	1	13.1
		10	5.11
		128	2.19
		1024	2.03
	GPU	1	14.5
		4	3.68
		128	0.021
		512	0.014
		1024	0.012

[GANs zoo]

GAN

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

Variational autoencoder



The Toolkit for Multivariate Data Analysis with ROOT (TMVA)

- ROOT: is the analysis framework used by most (HEP)-physicists

- Idea: rather than just implementing new MVA techniques and making them available in ROOT:

  - Have one common platform / interface for all MVA classifiers
  - Have common data pre-processing capabilities
  - Train and test all classifiers on same data sample and evaluate consistently
  - Provide common analysis (ROOT scripts) and application framework
  - Provide access with and without ROOT, through macros, C++ executables or python

Information from Multivariate Data Analysis with TMVA (June 19, 2008) by Andreas Hoecker

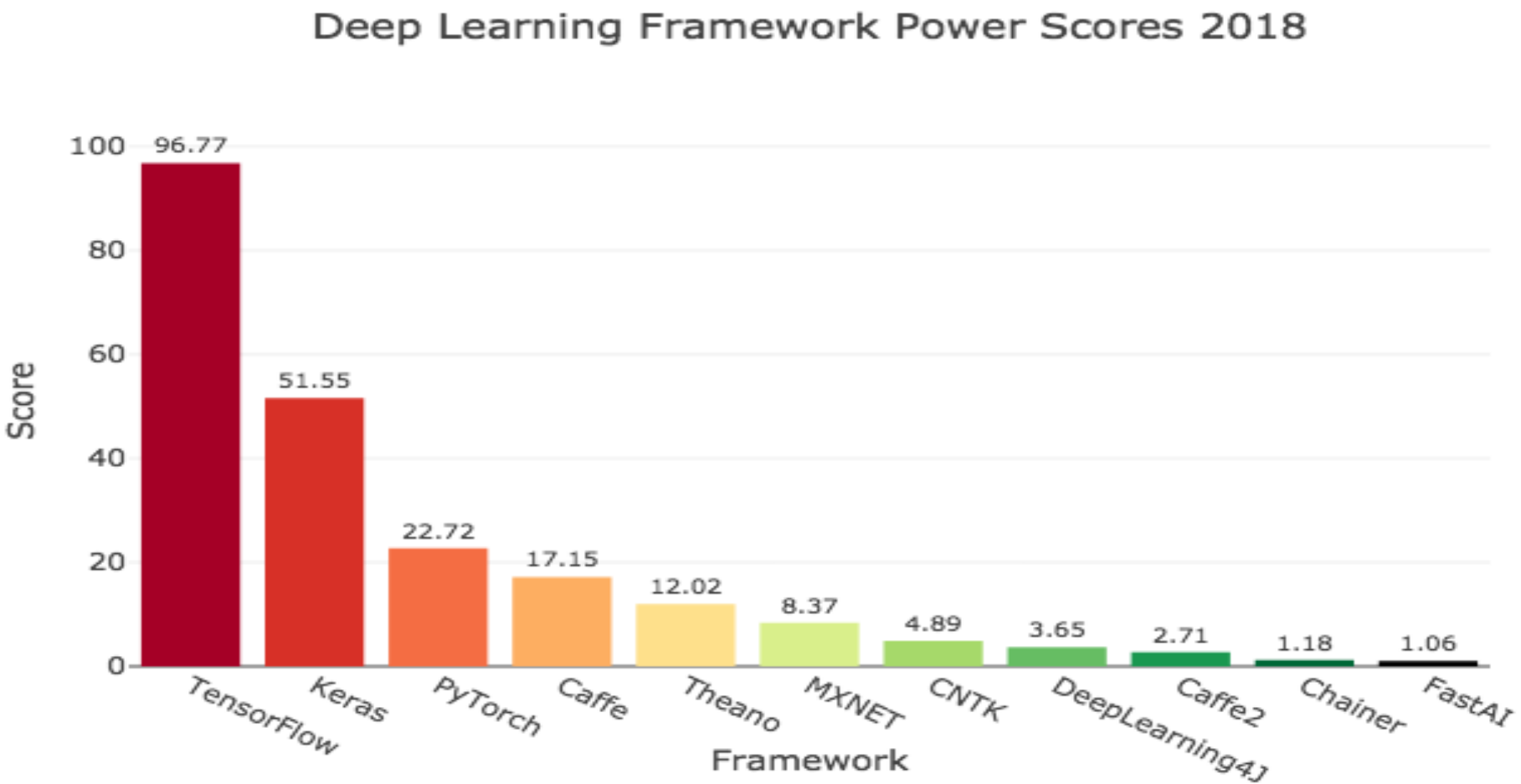
ML Frameworks

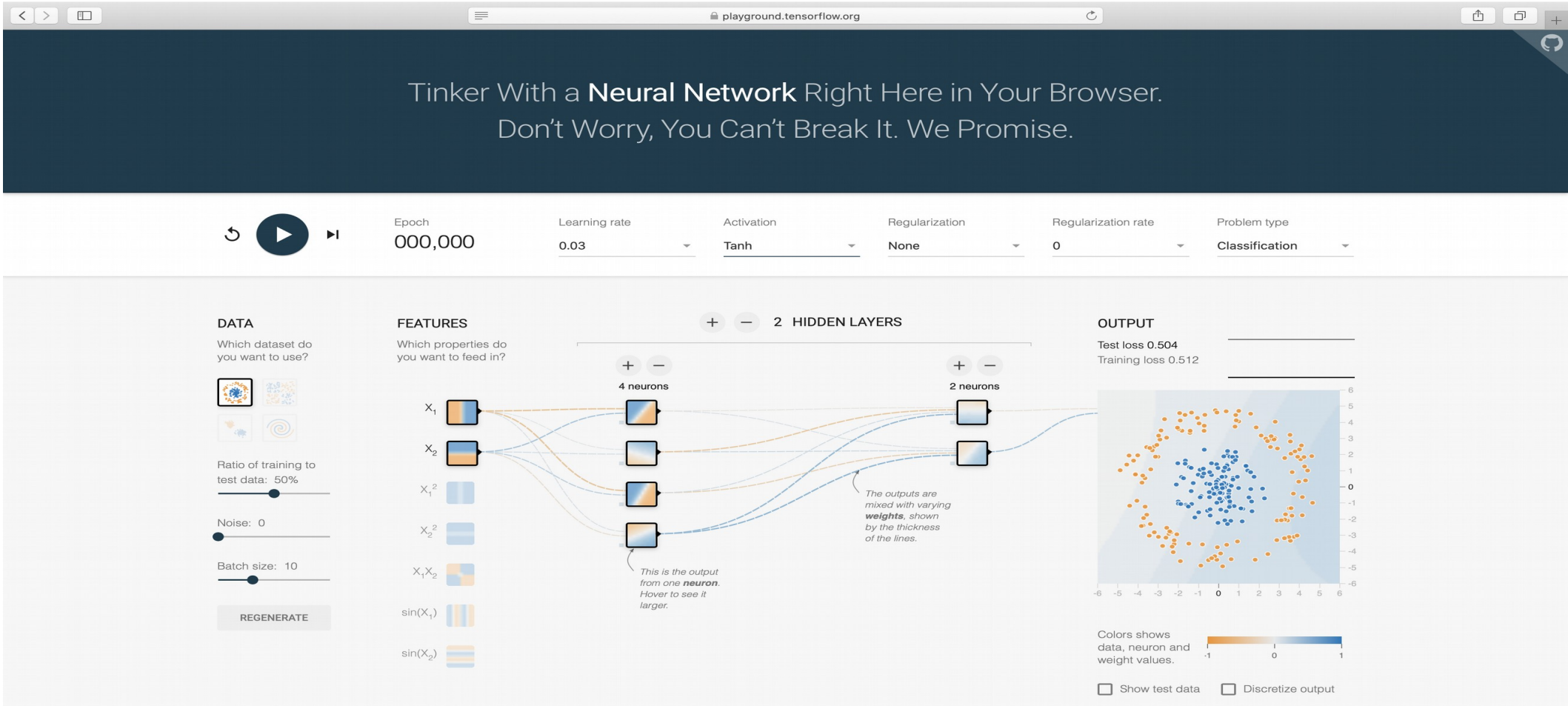
TF vs PyTorch comparison [link]

Frameworks

you should know about

- TensorFlow
- Keras (TF & Theano back-end)
- PyTorch
- ? Theano


Deep Learning Framework Power Scores 2018
Score
100
80
60
40
20
0
96.77
51.55
22.72
17.15
12.02
8.37
4.89
3.65
2.71
1.18
1.06
TensorFlow
Keras
PyTorch
Caffe
Theano
MXNET
CNTK
DeepLearning4J
Caffe2
Chainer
FastAI
Framework


playground.tensorflow.org
Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.504
Training loss 0.512
Colors shows data, neuron and weight values.
Show test data
Discretize output

localhost

Home | ann

jupyter ann Last Checkpoint: Last Monday at 2:56 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from scipy import stats

import tensorflow.keras as keras
import tensorflow.keras.backend as k

from tensorflow.keras.models import Model, load_model, Sequential
from tensorflow.keras.layers import Input, Dense, Activation, BatchNormalization, Dropout
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, Callback, TensorBoard
from tensorflow.keras.utils import Sequence, plot_model

from IPython.display import Image
```

In [2]:
```
from root_pandas import read_root
```
```
Welcome to JupyROOT 6.18/04
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import Ridge
```

In [4]:
```
fDataName = "./gun_k0L_allgev_FTFP_BERT_90000evt_ILD_l5_v02steel.root"
fDataName_mc = "./mc_gun_k0L_allgev_FTFP_BERT_90000evt_ILD_l5_v02steel.root"
fTreeName = "tree"
data = read_root(fDataName, fTreeName)
data_mc = read_root(fDataName_mc, fTreeName)
```

In [5]:
```
x = data
y = data_mc
```

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.1)
```

In [7]:
```
x_train.shape
```
Out[7]:
```
(7962, 10)
```

In [8]:
```
sns.pairplot(data_mc, kind="kde")
```
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x15478cad0>
```

In [ ]:
```
def metric1(y_true, y_pred):
    return ( y_pred - y_true ) / ( y_true )
```

In [ ]:
```
def metric2(y_true, y_pred):
    return k.mean( ( y_pred - y_true ) / ( y_true ) )
```

In [ ]:
```
model = Sequential()
model.add(Dense(64, activation='tanh', input_dim=x_train.shape[1]))
model.add(Dropout(0.5))
model.add(Dense(32, activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(1))

'''
a = Input(shape=x_train.shape[1])
b = Dense(8)(a)
b = Dropout(rate=0.5)(b)
b = Activation('tanh')(b)
b = Dense(4)(a)
b = Dropout(rate=0.5)(b)
b = Activation('tanh')(b)
c = Dense(1)(b)
model = Model(inputs = a, outputs = c)
'''

model.compile(loss='mean_squared_error', metrics=[metric1, metric2], optimizer='adam')
#model.compile(loss=loss1, metrics=[metric1, metric2], optimizer='adam')

model.summary()
```

In [ ]:
```
history = model.fit(x_train, y_train, epochs = 200, validation_data = (x_test, y_test))
```

In [ ]:
```
#a = model.evaluate( x_train, y_train )
#b = model.evaluate( x_test, y_test )
```

In [ ]:
```
history.history.keys()
```

In [ ]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
```

3.3 Building our Neural Network

Our NN consists of input, output and 1 hidden layer. We are using ReLU as activation function of the hidden layer and `softmax` for our output layer. As an additional bonus we will use `Dropout` — simple way to reduce overfitting during the training of our network. Let's wrap our model in a little helper function:

```
def multilayer_perceptron(x, weights, biases, keep_prob):
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
    layer_1 = tf.nn.relu(layer_1)
    layer_1 = tf.nn.dropout(layer_1, keep_prob)
    out_layer = tf.matmul(layer_1, weights['out']) + biases['out']
    return out_layer
```

Let's set the number of neurons in the hidden layer to 38 and randomly initialize the weights and biases considering their proper dimensions:

```
n_hidden_1 = 38
n_input = train_x.shape[1]
n_classes = train_y.shape[1]

weights = {
    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),
    'out': tf.Variable(tf.random_normal([n_hidden_1, n_classes]))
}

biases = {
    'b1': tf.Variable(tf.random_normal([n_hidden_1])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}

keep_prob = tf.placeholder("float")
```

We will train our model for 5,000 epochs (training steps) with a batch size

pytorch_example.py | pytorch_dropout.py

```
31
32  # the model will be a sequence of layers
33  model = torch.nn.Sequential()
34
35
36  # ANN with layers [784] -> [500] -> [300] -> [10]
37  # NOTE: the "p" is p_drop, not p_keep
38  model.add_module("dropout1", torch.nn.Dropout(p=0.2))
39  model.add_module("dense1", torch.nn.Linear(D, 500))
40  model.add_module("relu1", torch.nn.ReLU())
41  model.add_module("dropout2", torch.nn.Dropout(p=0.5))
42  model.add_module("dense2", torch.nn.Linear(500, 300))
43  model.add_module("relu2", torch.nn.ReLU())
44  model.add_module("dropout3", torch.nn.Dropout(p=0.5))
45  model.add_module("dense3", torch.nn.Linear(300, K))
46  # Note: no final softmax!
47  # just like Tensorflow, it's included in cross-entropy function
48
49
50  # define a loss function
51  # other loss functions can be found here:
52  # http://pytorch.org/docs/master/nn.html#loss-functions
53  loss = torch.nn.CrossEntropyLoss(size_average=True)
54  # Note: this returns a function!
55  # e.g. use it like: loss(logits, labels)
```

sklearn

Машинное обучение в несколько строк

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

scikit-learn.org

Install User Guide API Examples More Go

scikit-learn

Machine Learning in Python

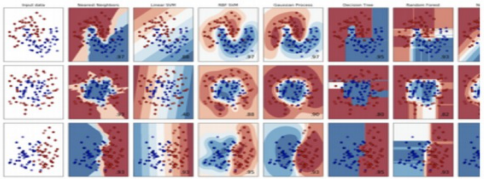
Getting Started | What's New in 0.22.2 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

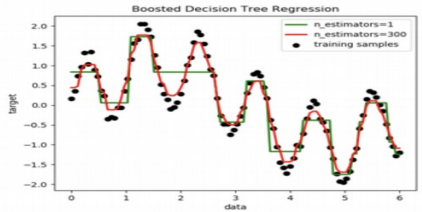


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

Preprocessing

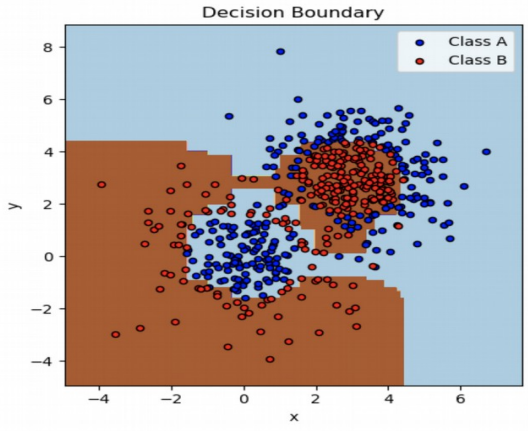
Feature extraction and normalization.

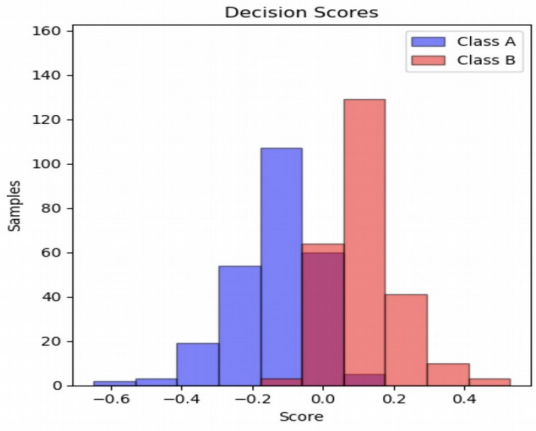
Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...

Two-class AdaBoost

This example fits an AdaBoosted decision stump on a non-linearly separable classification dataset composed of two "Gaussian quantiles" clusters (see `sklearn.datasets.make_gaussian_quantiles`) and plots the decision boundary and decision scores. The distributions of decision scores are shown separately for samples of class A and B. The predicted class label for each sample is determined by the sign of the decision score. Samples with decision scores greater than zero are classified as B, and are otherwise classified as A. The magnitude of a decision score determines the degree of likeness with the predicted class label. Additionally, a new dataset could be constructed containing a desired purity of class B, for example, by only selecting samples with a decision score above some value.

Note: Click here to download the full example code or to run this example in your browser via Binder






```
print(__doc__)

# Author: Noel Dawe <noel.dawe@gmail.com>
#
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_gaussian_quantiles


# Construct dataset
```

sklearn.tree.DecisionTreeRegressor

class sklearn.tree.**DecisionTreeRegressor**(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree regressor.

Read more in the User Guide.

Parameters:

criterion : *{"mse", "friedman_mse", "mae"}, default="mse"*
The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, "friedman_mse", which uses mean squared error with Friedman's improvement score for potential splits, and "mae" for the mean absolute error, which minimizes the L1 loss using the median of each terminal node.

New in version 0.18: Mean Absolute Error (MAE) criterion.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.

Jupyter Notebook / Jupyterlab

IP[y]: Notebook spectrogram Last Checkpoint: a few seconds ago (autosaved)

IPython (Python 3)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Simple spectral analysis

An illustration of the Discrete Fourier Transform using windowing, to reveal the frequency content of a sound signal.

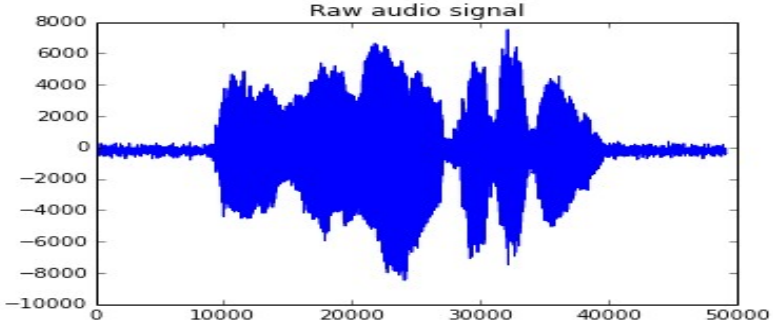
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

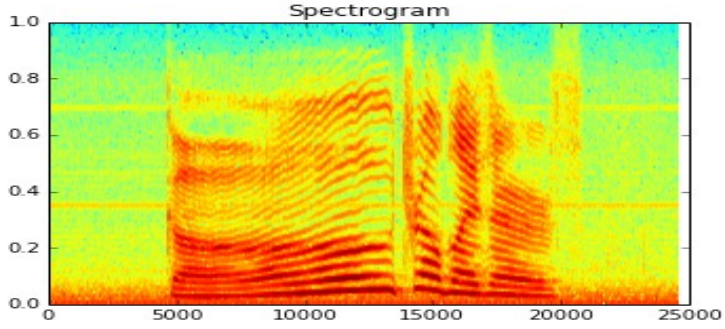
We begin by loading a datafile using SciPy's audio file support:

In [1]:
```
from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

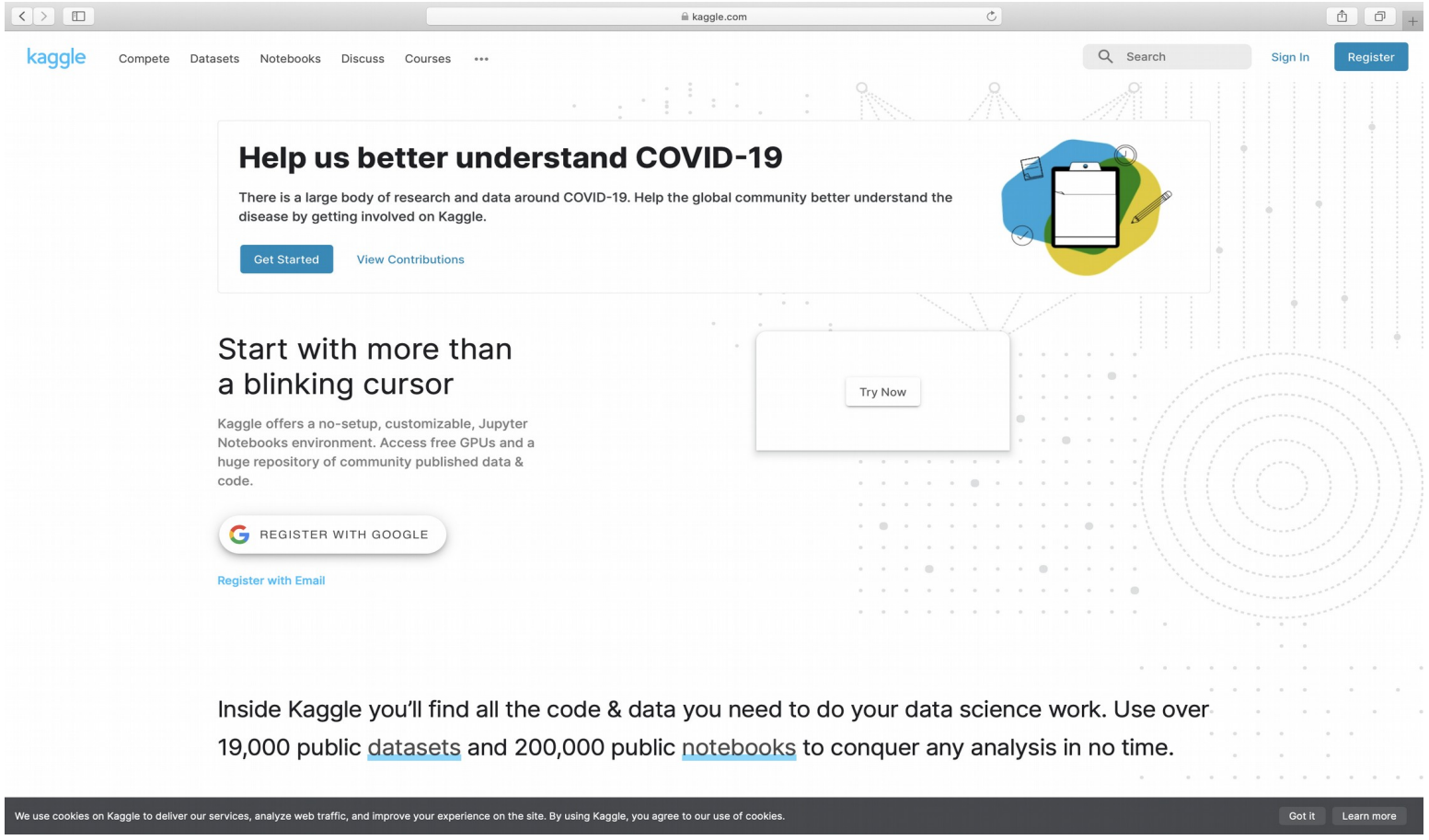
And we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

In [2]:
```
%matplotlib inline
from matplotlib import pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```





Kaggle


kaggle.com
kaggle
Compete
Datasets
Notebooks
Discuss
Courses
Search
Sign In
Register
Help us better understand COVID-19
There is a large body of research and data around COVID-19. Help the global community better understand the disease by getting involved on Kaggle.
Get Started
View Contributions
Start with more than a blinking cursor
Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data & code.
REGISTER WITH GOOGLE
Register with Email
Try Now
Inside Kaggle you'll find all the code & data you need to do your data science work. Use over 19,000 public datasets and 200,000 public notebooks to conquer any analysis in no time.
We use cookies on Kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using Kaggle, you agree to our use of cookies.
Got it
Learn more

Kaggle

- one of the largest data communities in the world
- ML competitions
- public datasets and kernels (code snippets)
- short-form AI education

Higgs challenge

kaggle

open data and baselines for practice!

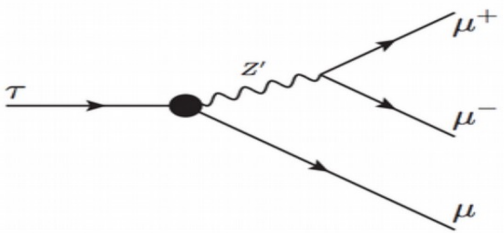
2014 Higgs Boson challenge by ATLAS [1]

2015 $\tau \to \mu\mu\mu$ by LHCb and Yandex [2]

2018 TrackML by CERN [3], [4]

huge impact on ML community in HEP

TrackML



GitHub / GitLab

github.com

Search or jump to... / Pull requests Issues Marketplace Explore

Ete Remi
rete

Follow

HEP computing

DESY
Hamburg, Germany
remi.ete@gmail.com
http://lyorete.sytes.net

Block or report user

Overview | Repositories 63 | Projects 0 | Stars 18 | Followers 16 | Following 43

Pinned

iLCSoft/MarlinMT
Modular Analysis and Reconstruction for the LINear Collider - Multi-threaded version
C++ 2

iLCSoft/LCCalibration
Automated energy calibration procedure for Linear Colliders
Python 3

iLCSoft/SIO
Simple IO package
C 2

LCEve
Linear Collider event display based on LCIO and experimental Eve7
C++

iLCSoft/LCIO
Linear Collider I/O
C++ 7 15

DQM4hep/DQM4hep
Data quality monitoring for high energy physics. Doc available at
C++ 2 6

953 contributions in the last year

Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar
Mon
Wed
Fri

Learn how we count contributions. Less More

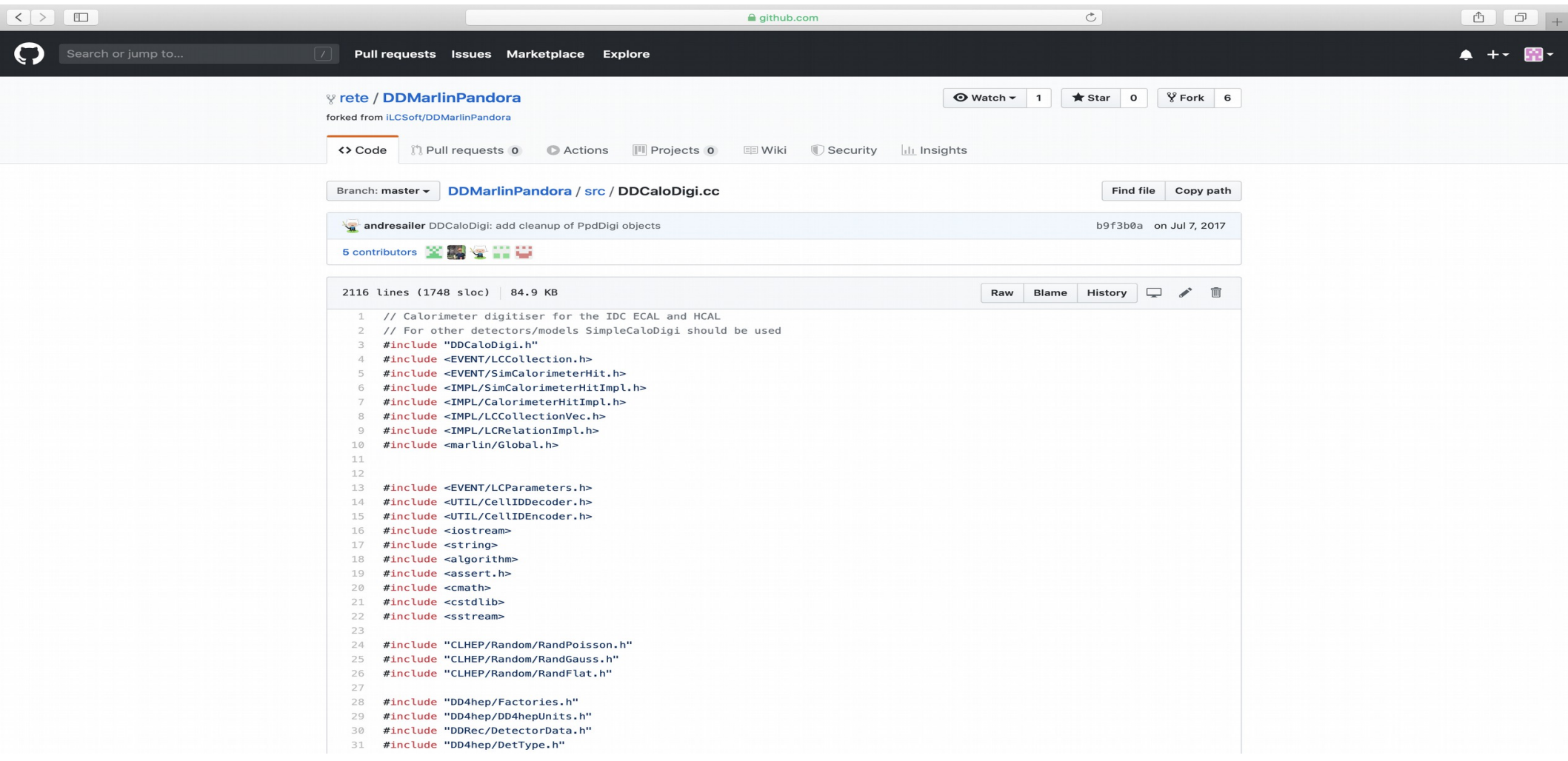
Contribution activity

2020
2019
2018

April 1, 2020

rete has no activity yet for this period.

March 2020


github.com
Search or jump to...
Pull requests
Issues
Marketplace
Explore
rete / DDMarlinPandora
forked from iLCSoft/DDMarlinPandora
Watch 1
Star 0
Fork 6
Code
Pull requests 0
Actions
Projects 0
Wiki
Security
Insights
Branch: master
DDMarlinPandora / src / DDCaloDigi.cc
Find file
Copy path
andresailer DDCaloDigi: add cleanup of PpdDigi objects
b9f3b0a on Jul 7, 2017
5 contributors
2116 lines (1748 sloc) | 84.9 KB
Raw
Blame
History
// Calorimeter digitiser for the IDC ECAL and HCAL
// For other detectors/models SimpleCaloDigi should be used
#include "DDCaloDigi.h"
#include <EVENT/LCCollection.h>
#include <EVENT/SimCalorimeterHit.h>
#include <IMPL/SimCalorimeterHitImpl.h>
#include <IMPL/CalorimeterHitImpl.h>
#include <IMPL/LCCollectionVec.h>
#include <IMPL/LCRelationImpl.h>
#include <marlin/Global.h>
#include <EVENT/LCParameters.h>
#include <UTIL/CellIDDecoder.h>
#include <UTIL/CellIDEncoder.h>
#include <iostream>
#include <string>
#include <algorithm>
#include <assert.h>
#include <cmath>
#include <cstdlib>
#include <sstream>
#include "CLHEP/Random/RandPoisson.h"
#include "CLHEP/Random/RandGauss.h"
#include "CLHEP/Random/RandFlat.h"
#include "DD4hep/Factories.h"
#include "DD4hep/DD4hepUnits.h"
#include "DDRec/DetectorData.h"
#include "DD4hep/DetType.h"

IF YOU WANT TO BE OK.

IF YOU WANT TO BE
GEORGE BUSH.
RIDE A BULL.
AND DRINK MUCH PUNSCH!

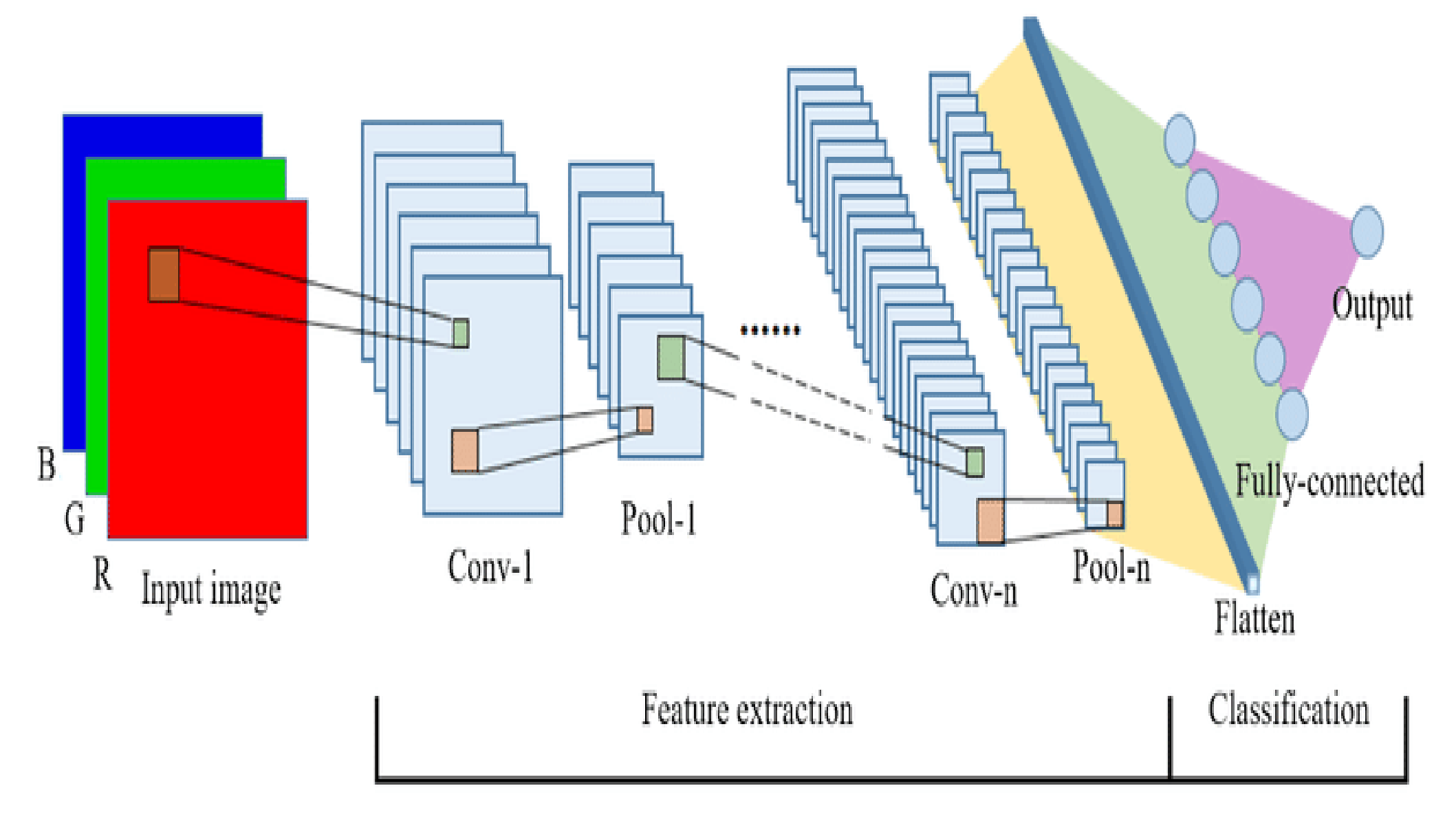
IF YOU WANT TO BE
OBAMA.
OVERRIDE
ГИПОПОТАМА!

41

BOTAY ENGLISH EVERY DAY!

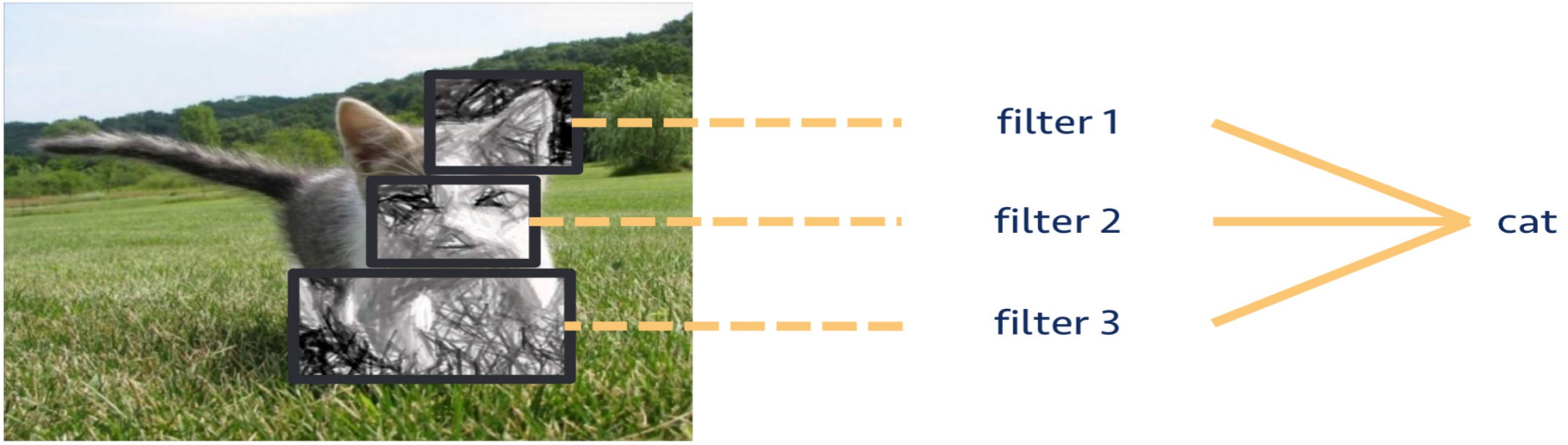
СПАСИБО КОТЛАС ИНТЕРНЕТ ЗА НАШЕ СЧАСТЛИВОЕ ДЕТСТВО!

Backup Slides


B
G
R
Input image
Conv-1
Pool-1
******
Conv-n
Pool-n
Flatten
Fully-connected
Output
Feature extraction
Classification

Convolutional NN

patterns

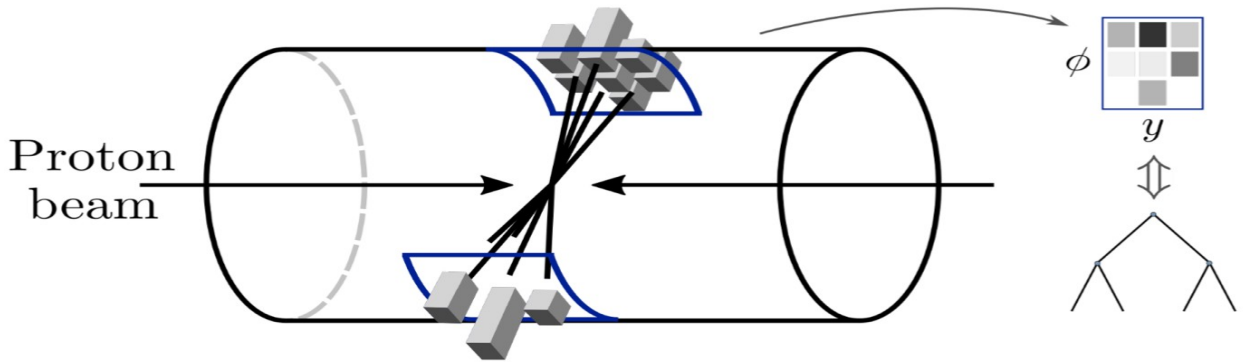


What's a jet, anyway?

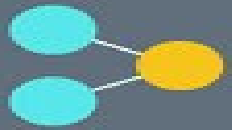
A jet can be described in a variety of ways

- Sum of its particle constituents
- Image of energy deposits in rapidity-azimuth
- Complete basis of observables (e.g. energy flow polynomials)
- Image of primary Lund declustering sequence
- Binary tree associated with the jet clustering sequence
- ...

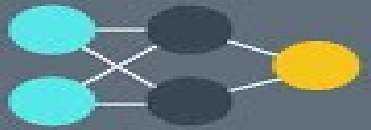
Most of them involve some trade-off between interpretability, representability, robustness and performance.



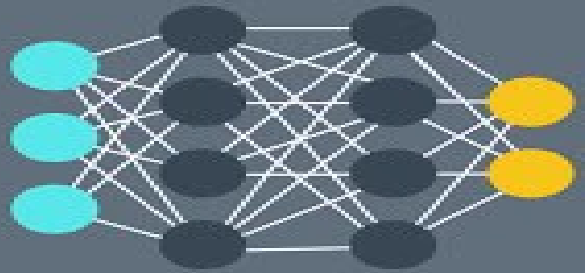
NEURAL NETWORK ARCHITECTURE TYPES



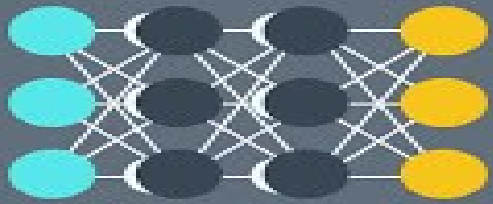
SINGLE LAYER PERCEPTRON



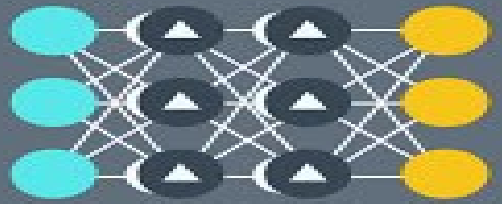
RADIAL BASIS NETWORK



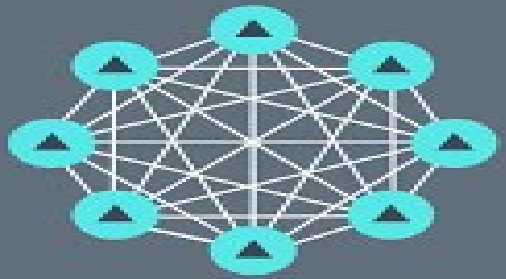
MULTI LAYER PERCEPTRON



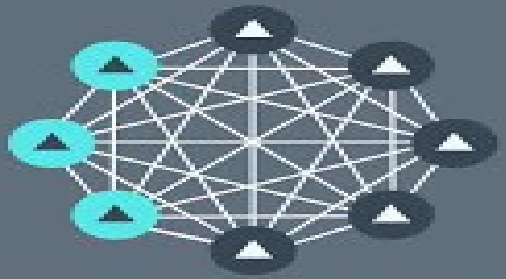
RECURRENT NEURAL NETWORK



LSTM RECURRENT NEURAL NETWORK



HOPFIELD NETWORK



BOLTZMANN MACHINE

INPUT UNIT

OUTPUT UNIT

HIDDEN UNIT

FEEDBACK WITH MEMORY UNIT

BACKFED INPUT UNIT

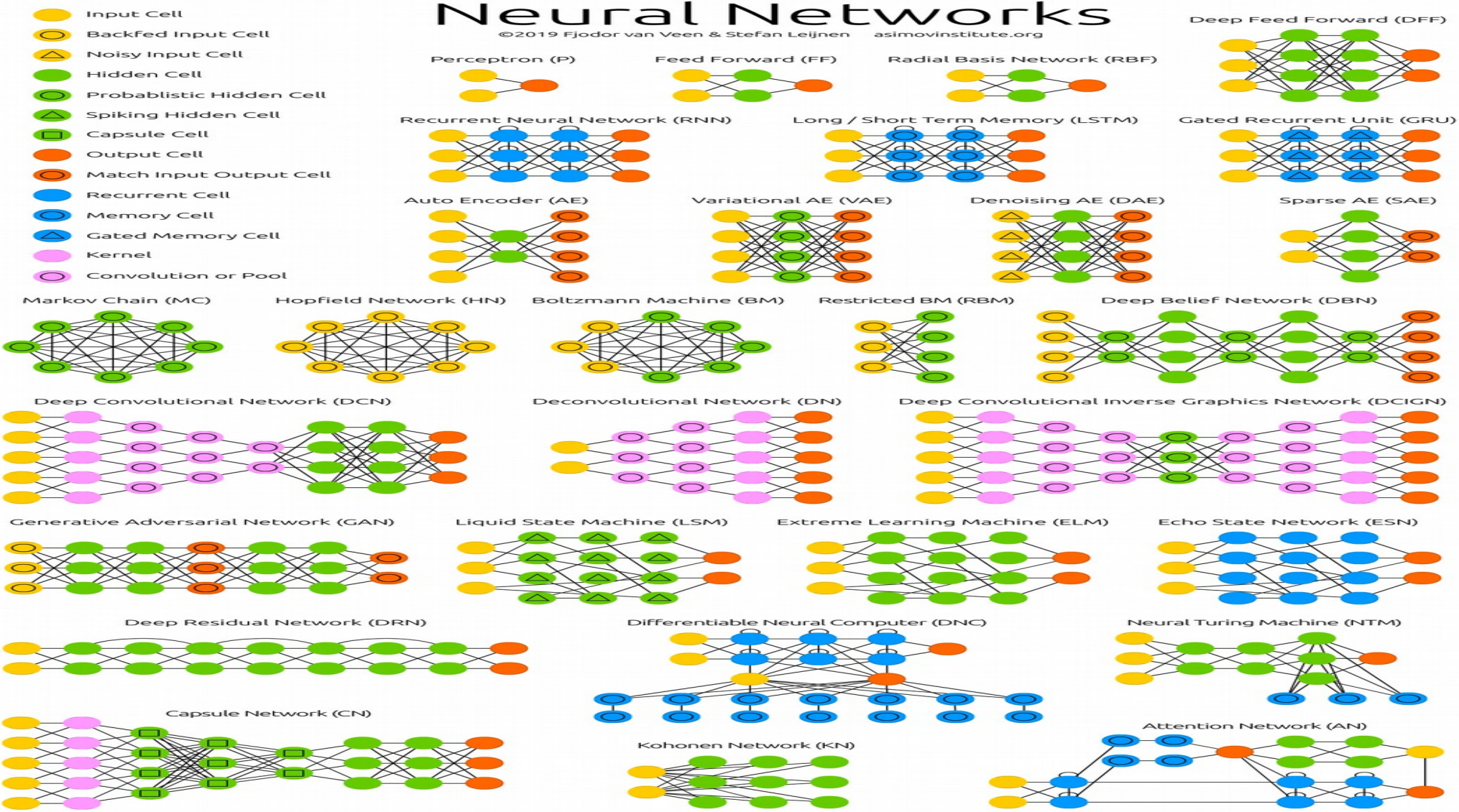
PROBABILISTIC HIDDEN UNIT

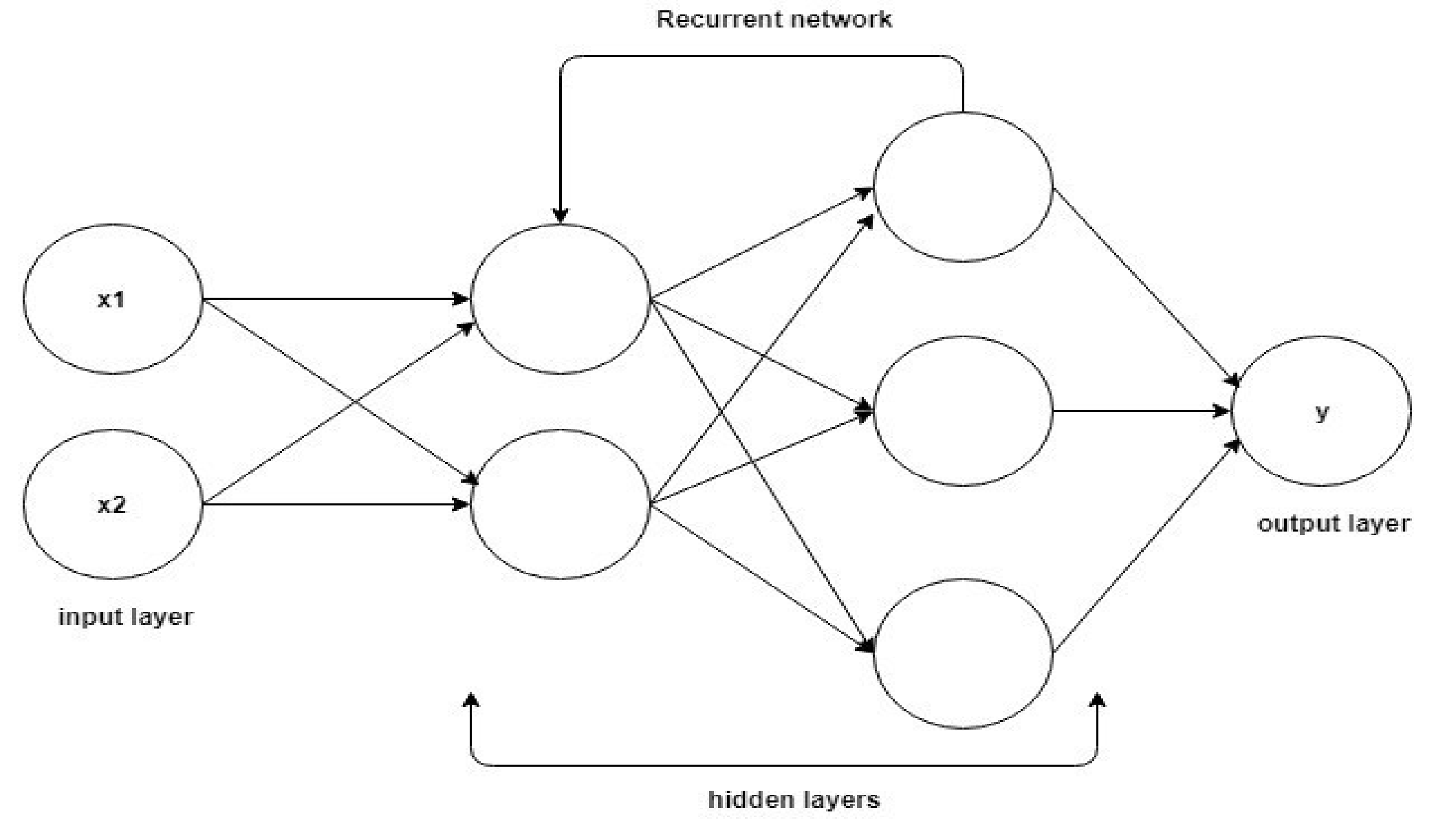
A mostly complete chart of

Neural Networks

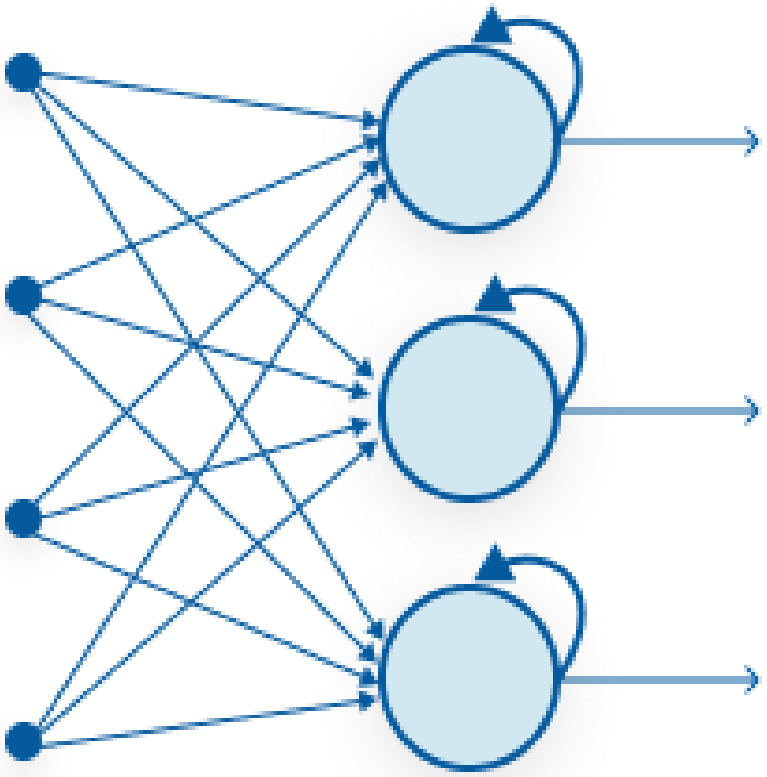
 asimovinstitute.org

- Input Cell
- Backfed Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Gated Memory Cell
- Kernel
- Convolution or Pool

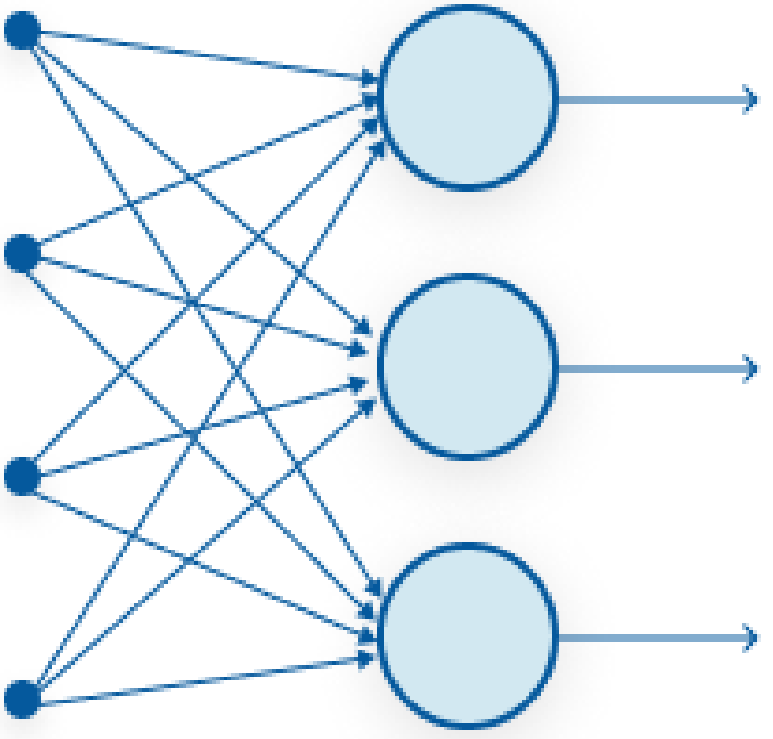



Recurrent network
x1
x2
input layer
y
output layer
hidden layers

Recurrent Neural Network structure



Recurrent Neural Network



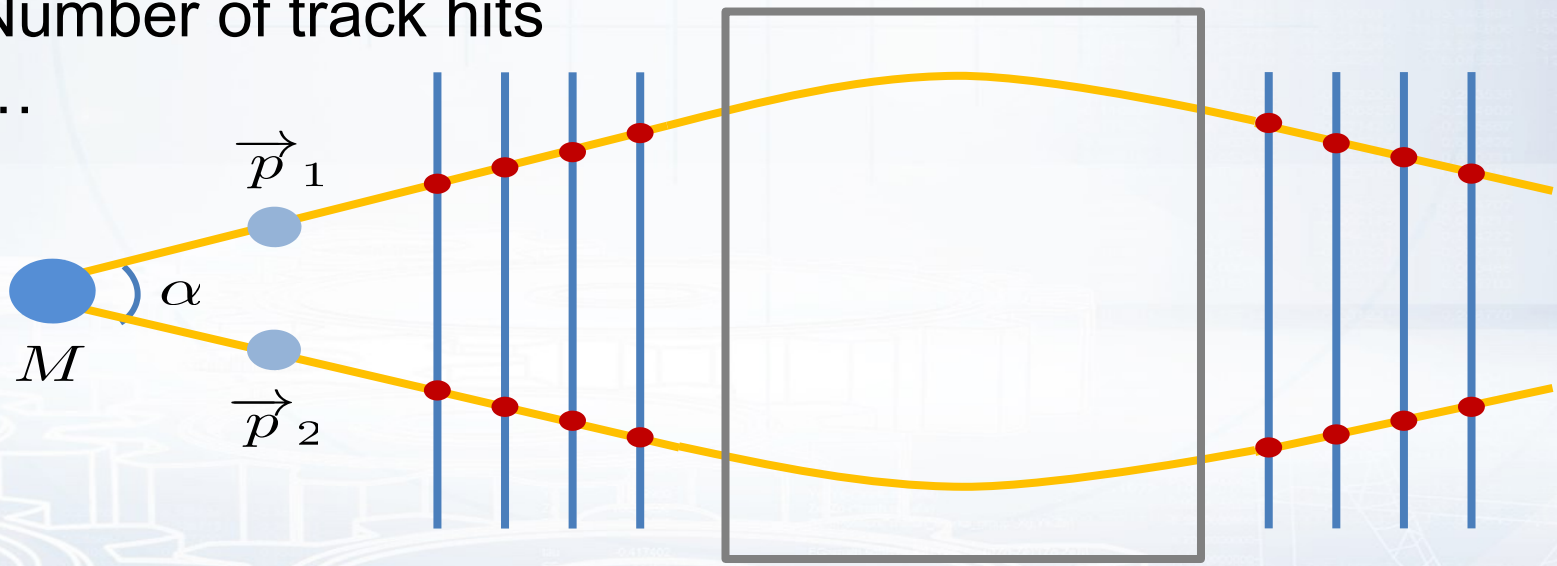
Feed-Forward Neural Network

Machine Learning Challenges

- **Precise and fast particle tracking:**
  - **single tracks, shower,** jets.
- **Particle identification;**
- Fast and accurate online data processing and filtering;
- Anomaly detection:
  - data quality monitoring;
  - infrastructure monitoring.
- **Detector design optimization (bayesian optimization, surrogate modelling).**

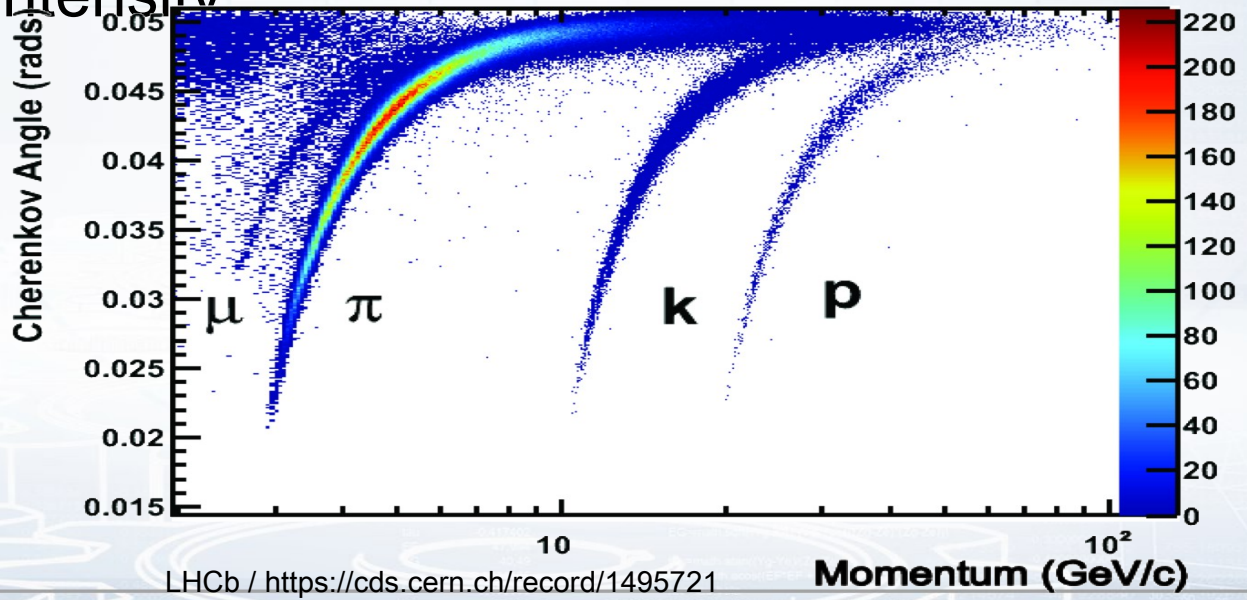
Tracking system features

- Particle momentum
- Particle charge
- Track parameters
- Quality of track fit
- Number of track hits
- …



RICH features

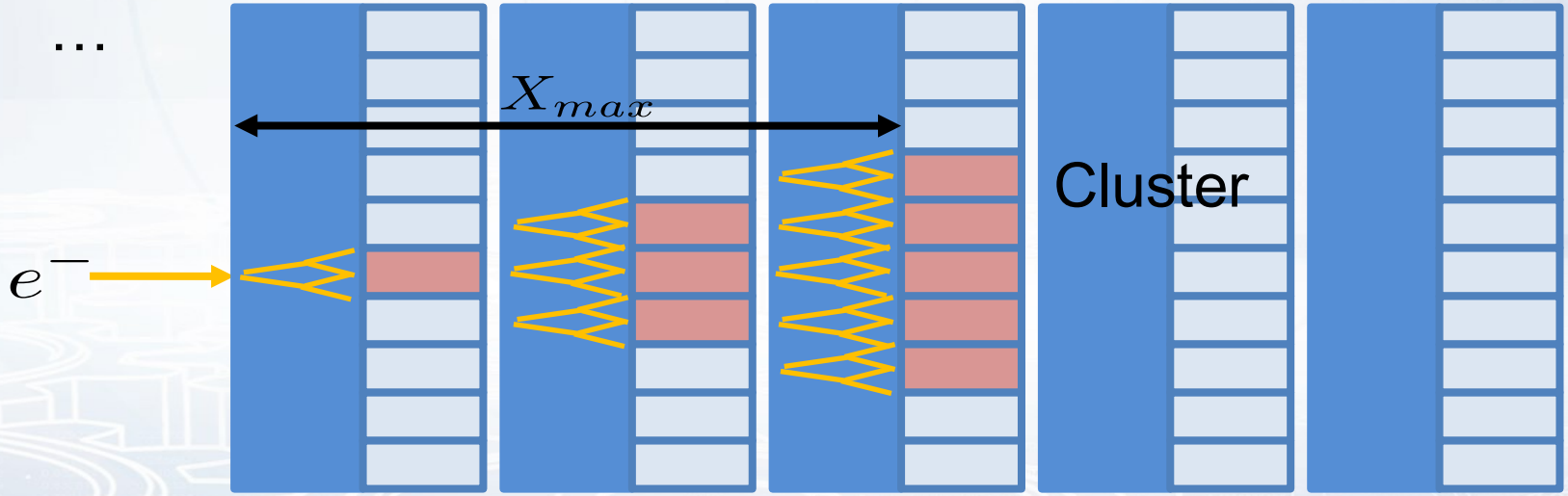
- Angle θ
- Quality of angle reconstruction
- Reconstructed particle type
- Reconstructed particle energy
- Light intensity
- …



LHCb / https://cds.cern.ch/record/1495721

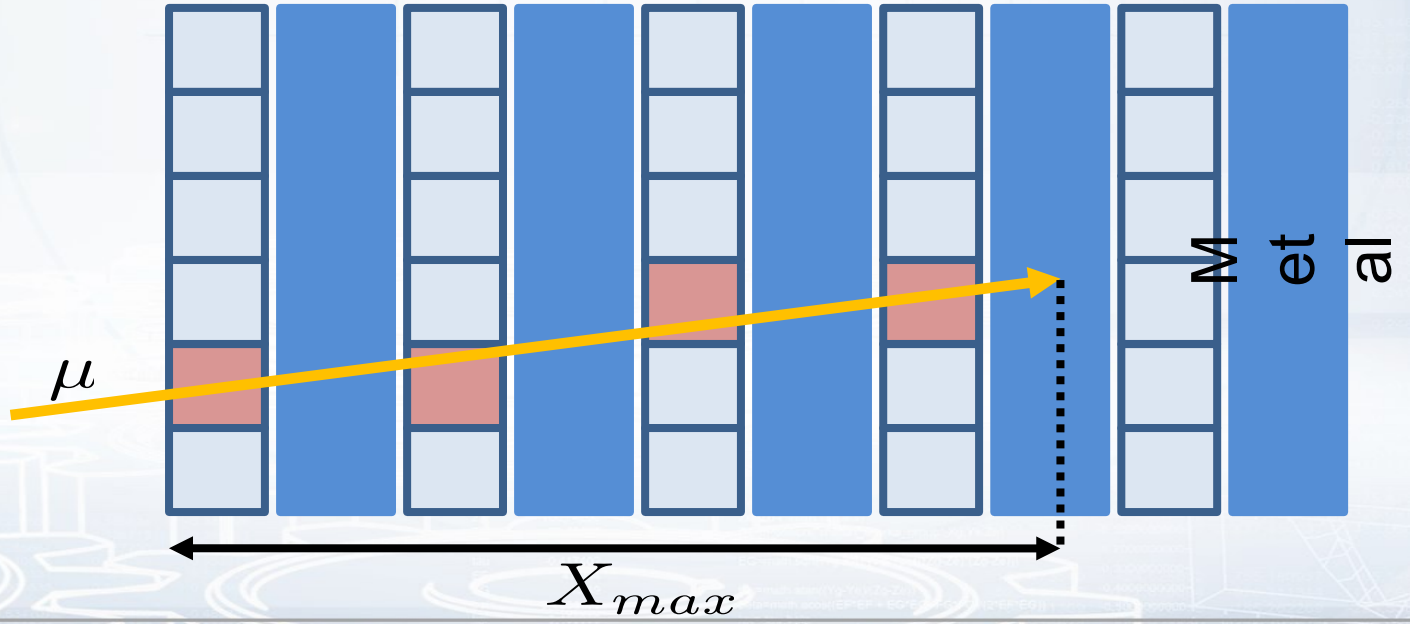
Calorimeter features

- Measured particle enegry
- Shower parameters: X_{max} ,width, ...
- Number of clusters in each layer
- Intensity of the clusters
- Distance form track of the original particle
- …



Muon chambers features

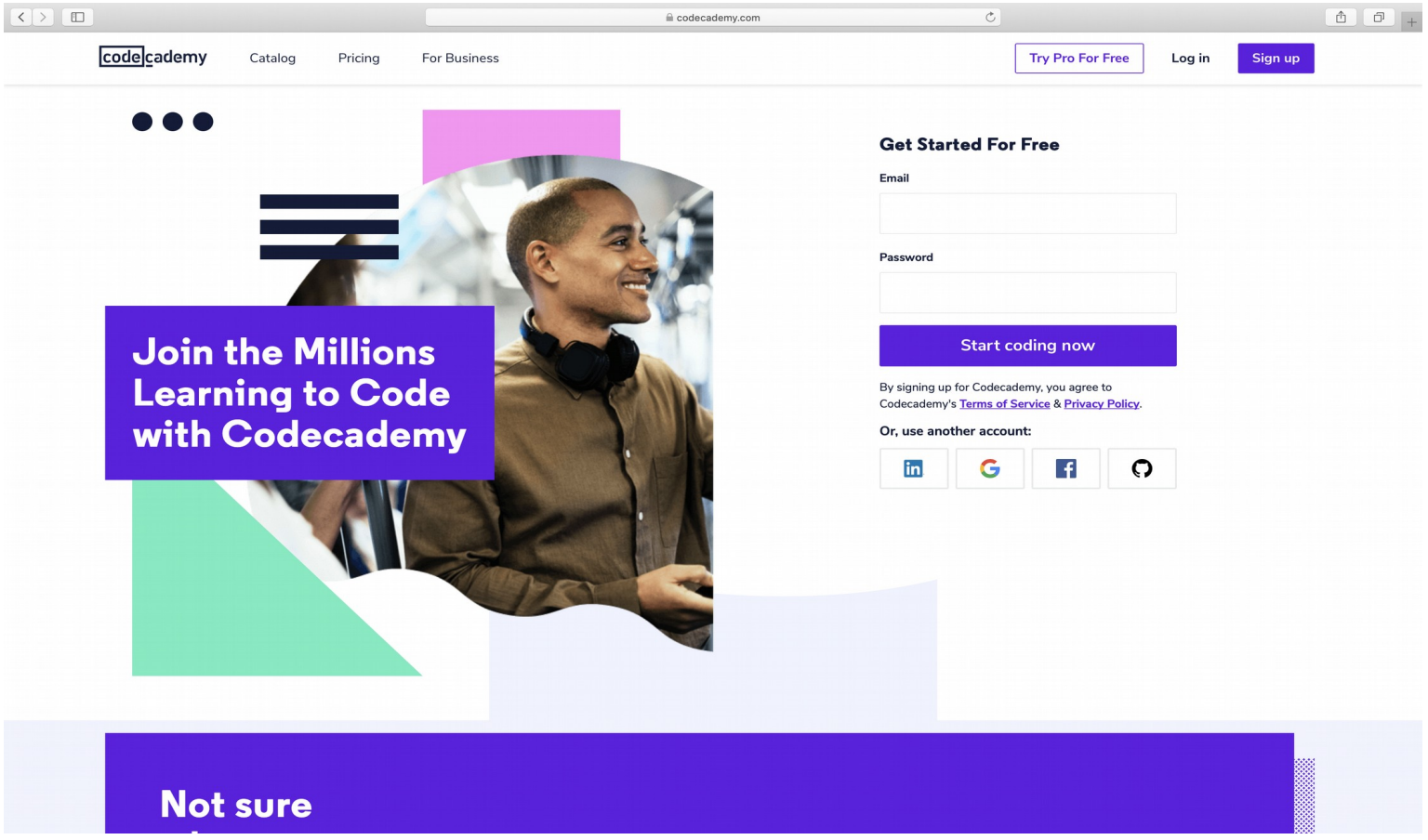
- Muon track parameters
- Quality of track fit
- Number of active chambers
- Distance between the track and the active chambers X_{max}
-
- …



Machine Learning Challenge Examples

- **Precise and fast particle tracking:**
  - **single tracks, shower,** jets.
- **Particle identification;**
- Fast and accurate online data processing and filtering;
- Anomaly detection:
  - data quality monitoring;
  - infrastructure monitoring.
- **Detector design optimization (bayesian optimization);**
- **Data Analysis (signal from background separation);**
- Simulation:
  - speed-up simulation using generative models;
  - simulator parameters optimization (tuning).

Codecademy


codecademy.com
codecademy
Catalog
Pricing
For Business
Try Pro For Free
Log in
Sign up
Join the Millions Learning to Code with Codecademy
Get Started For Free
Email
Password
Start coding now
By signing up for Codecademy, you agree to Codecademy's Terms of Service & Privacy Policy.
Or, use another account:
Not sure

Linear Regression

Model: x ⟶ wx +b ⟶ y^{pred}

Objective function:

$$L = \frac{1}{N}\sum_i (y_i - y_i^{pred})^2$$

Optimization (iterative):

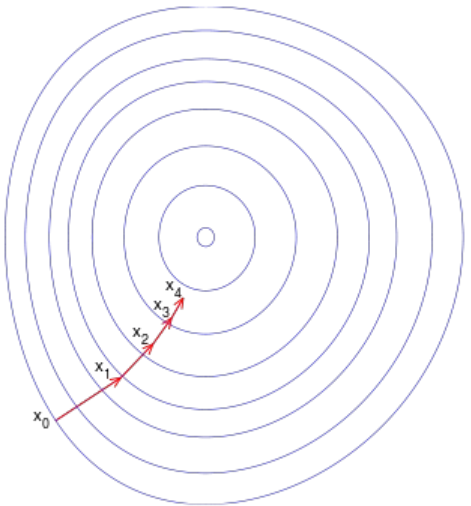
$$w_{i+1} := w_i - \alpha \cdot \frac{\partial L}{\partial w_i}$$

Recap: Gradient Descent

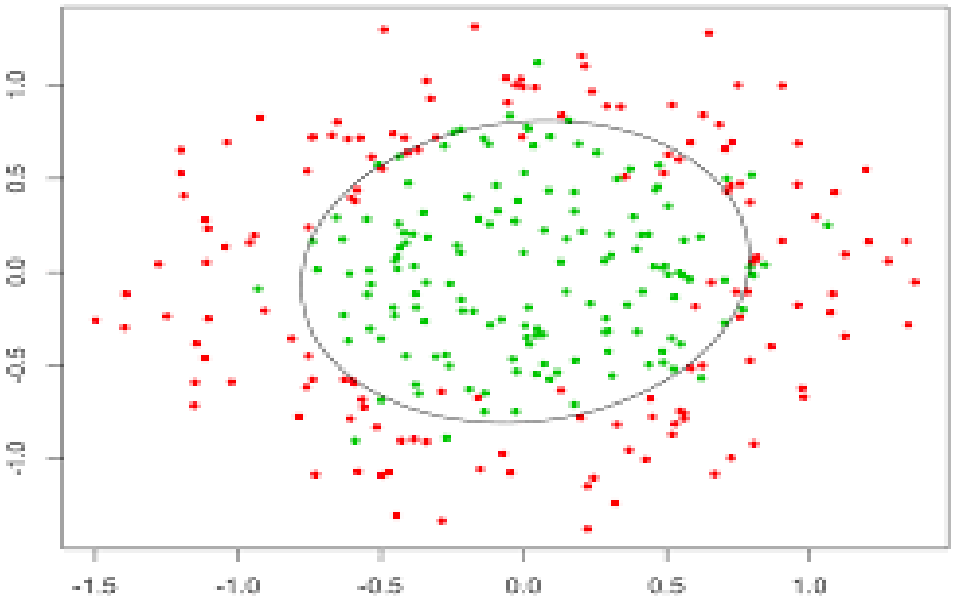
Update:

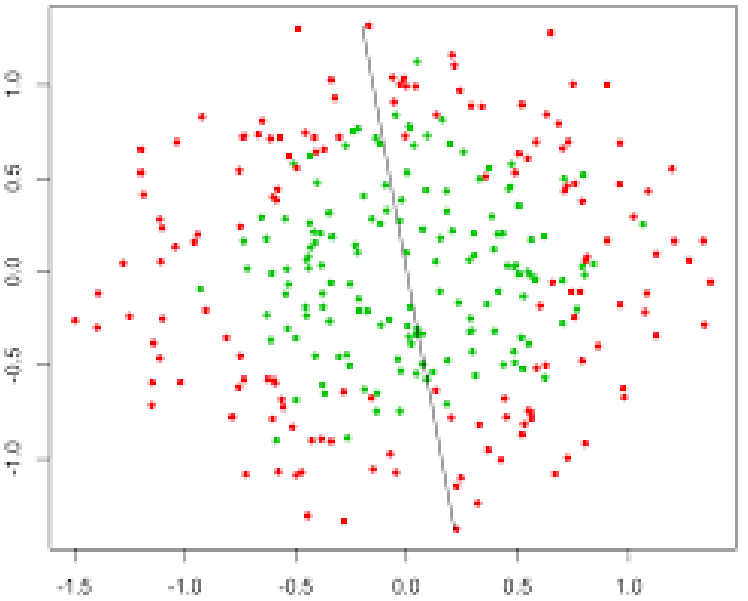
$$w_{i+1} := w_i - \alpha \cdot \frac{\partial L}{\partial w_i}$$

- a – learning rate a<<1
- L – loss function



Nonlinear dependencies





What we have

What we want

How to learn that?