MACHINE LEARNING IN HIGH ENERGY PHYSICS

LECTURE #1

Alex Rogozhnikov, 2015

INTRO NOTES

- 4 days
- two lectures, two practice seminars every day
- this is introductory track to machine learning
- kaggle competition!

WHAT IS ML ABOUT?

Inference of statistical dependencies which give us ability to predict

Data is cheap, knowledge is precious

WHERE ML IS CURRENTLY USED?

- Search engines, spam detection
- Security: virus detection, DDOS defense
- Computer vision and speech recognition
- Market basket analysis, Customer relationship management (CRM)
- Credit scoring, fraud detection
- Health monitoring
- Churn prediction
- ... and hundreds more

ML IN HIGH ENERGY PHYSICS

- High-level triggers (LHCb trigger system: $40\text{MHz} \to 5\text{kHz}$)
- Particle identification
- Tagging
- Stripping line
- Analysis

Different data is used on different stages

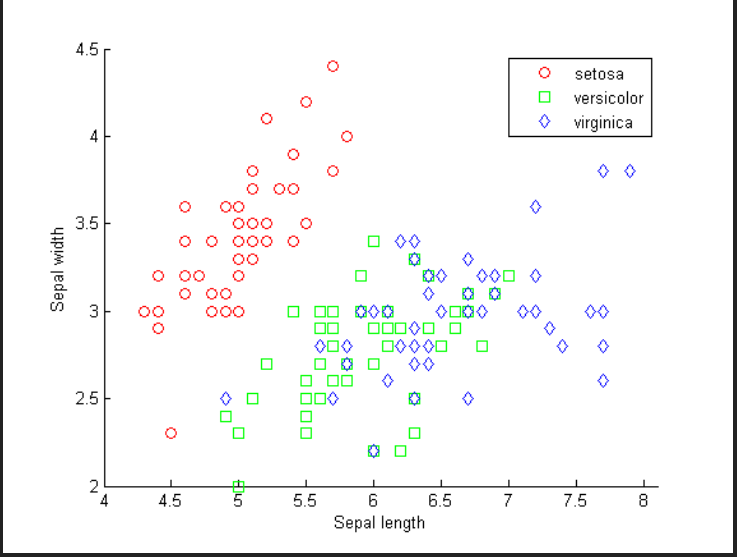
GENERAL NOTION

In supervised learning the training data is represented as set of pairs

$$x_i, y_i$$

- i is index of event
- x_i is vector of features available for event
- y_i is target — the value we need to predict

CLASSIFICATION EXAMPLE



$y_i \in Y$, Y if finite set

on the plot: $x_i \in \mathbb{R}^2, y_i \in \{0, 1, 2\}$

Examples:

- defining type of particle (or decay channel)
- $Y = \{0, 1\}$ — binary classification, 1 is signal, 0 is bck

REGRESSION

$$y \in \mathbb{R}$$

Examples:

- predicting price of house by it's positions
- predicting number of customers / money income
- reconstructing real momentum of particle

Why need automatic classification/regression?

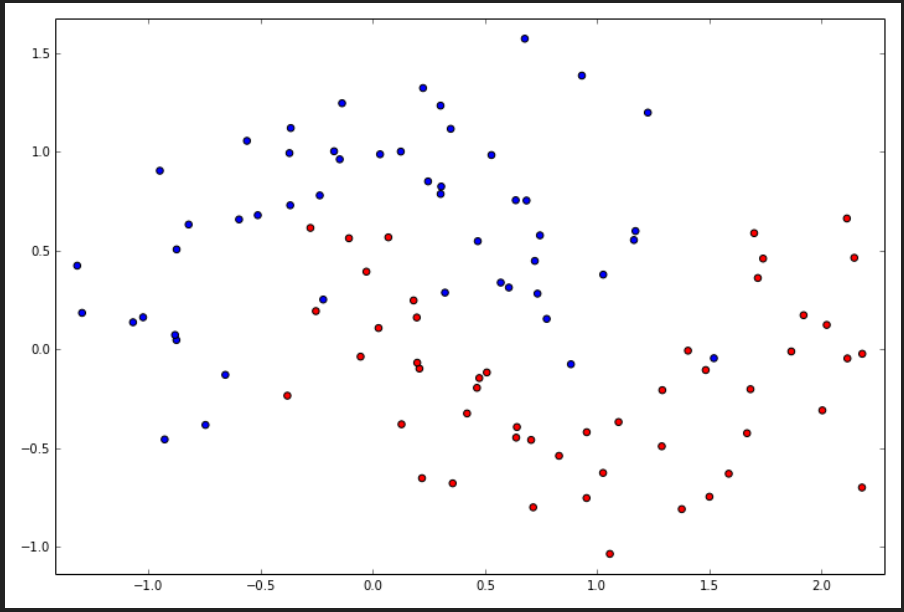
- in applications up to thousands of features
- higher quality
- much faster adaptation to new problems

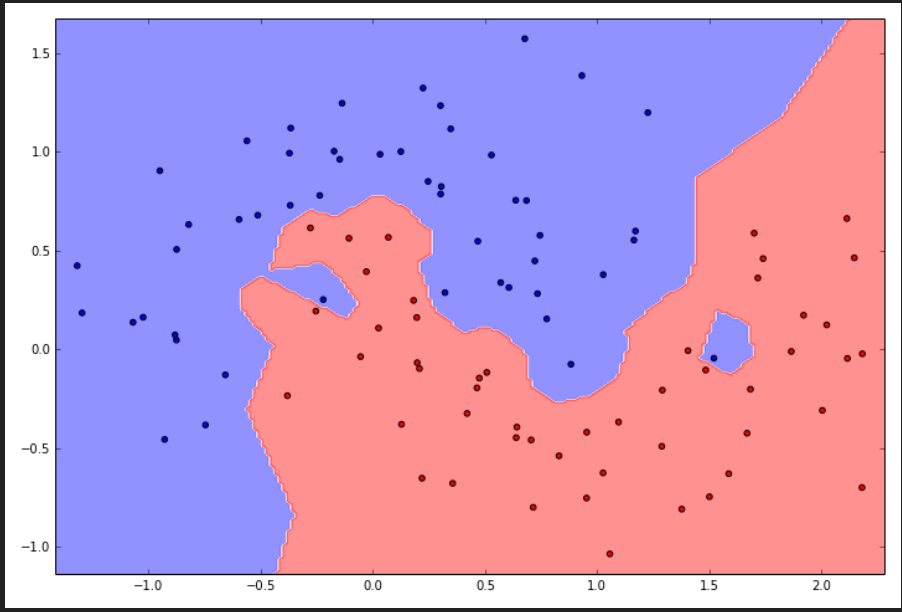
CLASSIFICATION BASED ON NEAREST NEIGHBOURS

Given training set of objects and their labels $\{x_i, y_i\}$ we predict the label for new observation.

$$\hat{y} = y_j, \qquad j = \arg\min_i \rho(x, x_i)$$

VISUALIZATION OF DECISION RULE

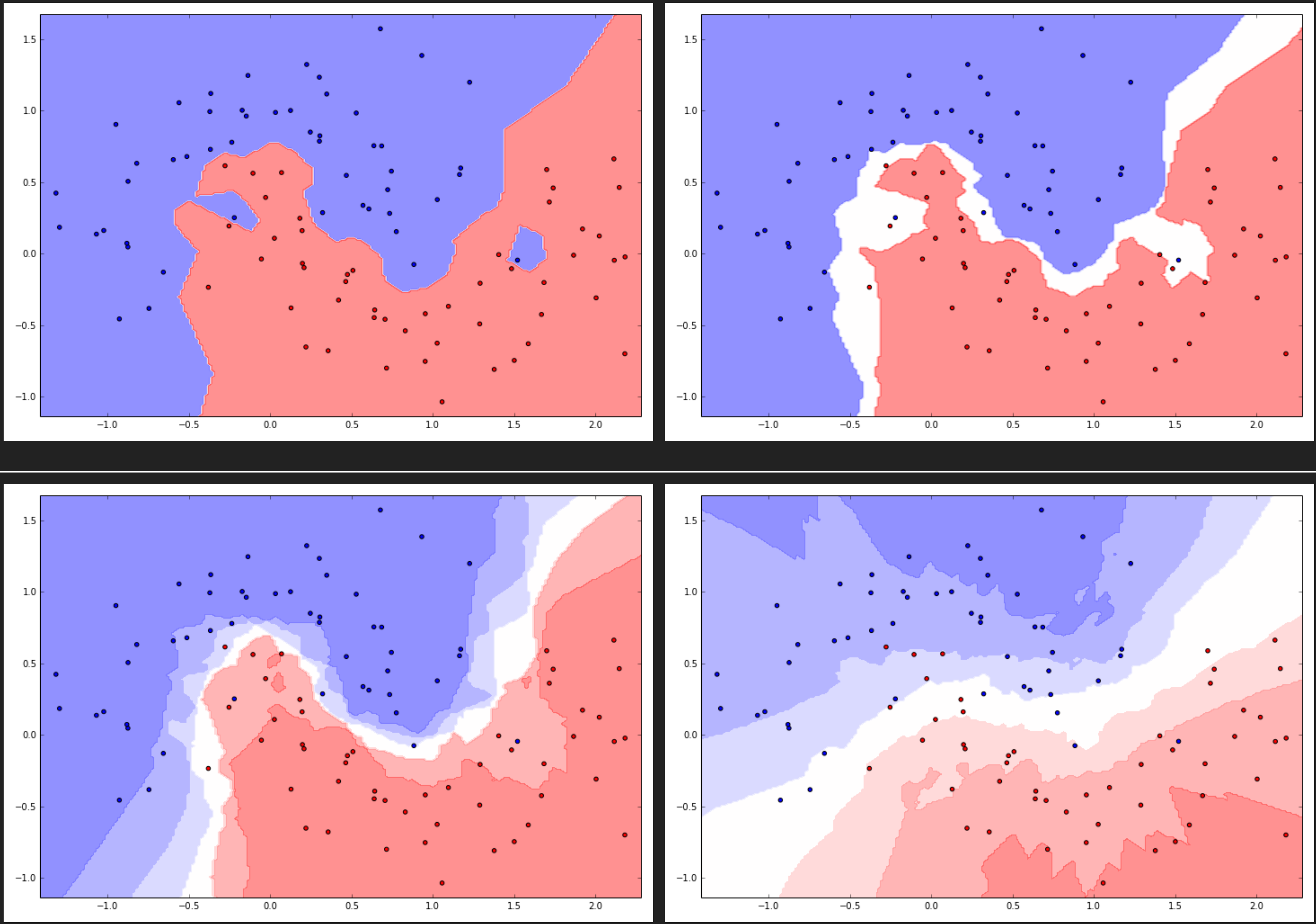




k NEAREST NEIGHBOURS

A better way is to use k neighbours:

$$p_i(x) = \frac{\#\text{ of knn events in class } i}{k}$$


1.5 1.0 0.5 0.0 −0.5 −1.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
1.5 1.0 0.5 0.0 −0.5 −1.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
1.5 1.0 0.5 0.0 −0.5 −1.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
1.5 1.0 0.5 0.0 −0.5 −1.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0

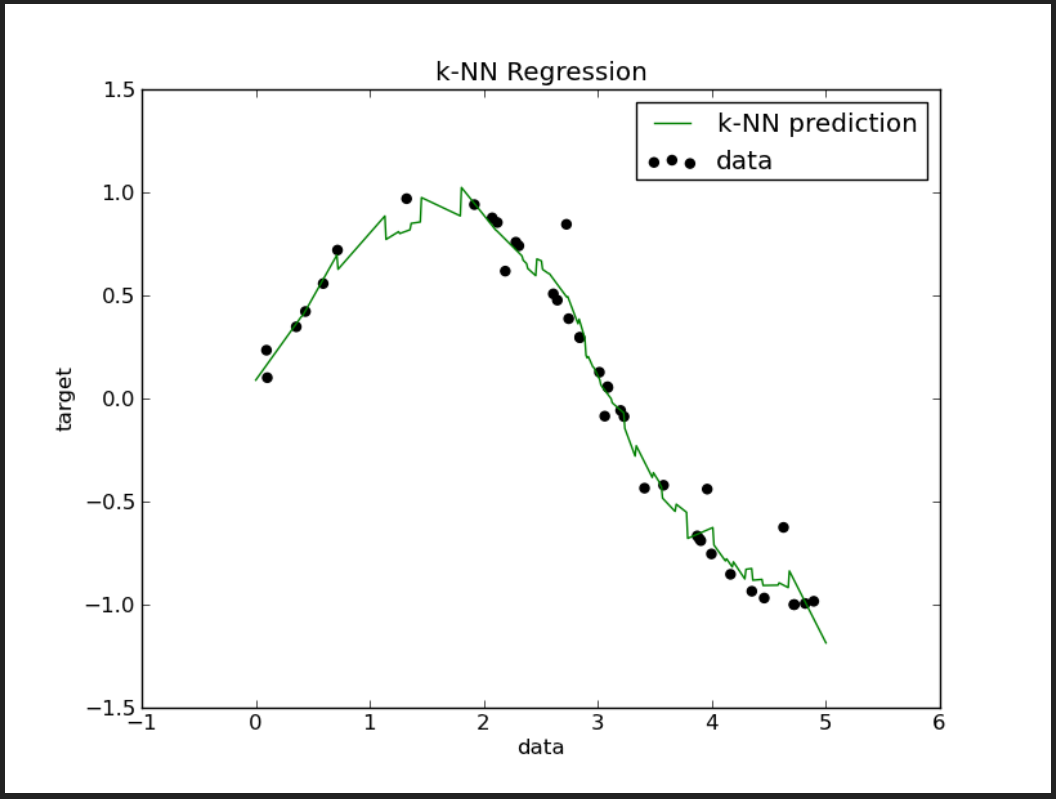
k = 1, 2, 5, 30

OVERFITTING

- what is the quality of classification on training dataset when $k = 1$?
- answer: it is ideal (closest neighbor is event itself)
- quality is lower when $k > 1$
- this doesn't mean $k = 1$ is the best,
  it means we cannot use training events to estimate quality
- when classifier's decision rule is too complex and captures details from training data that are not relevant to distribution, we call this overfitting (more details tomorrow)

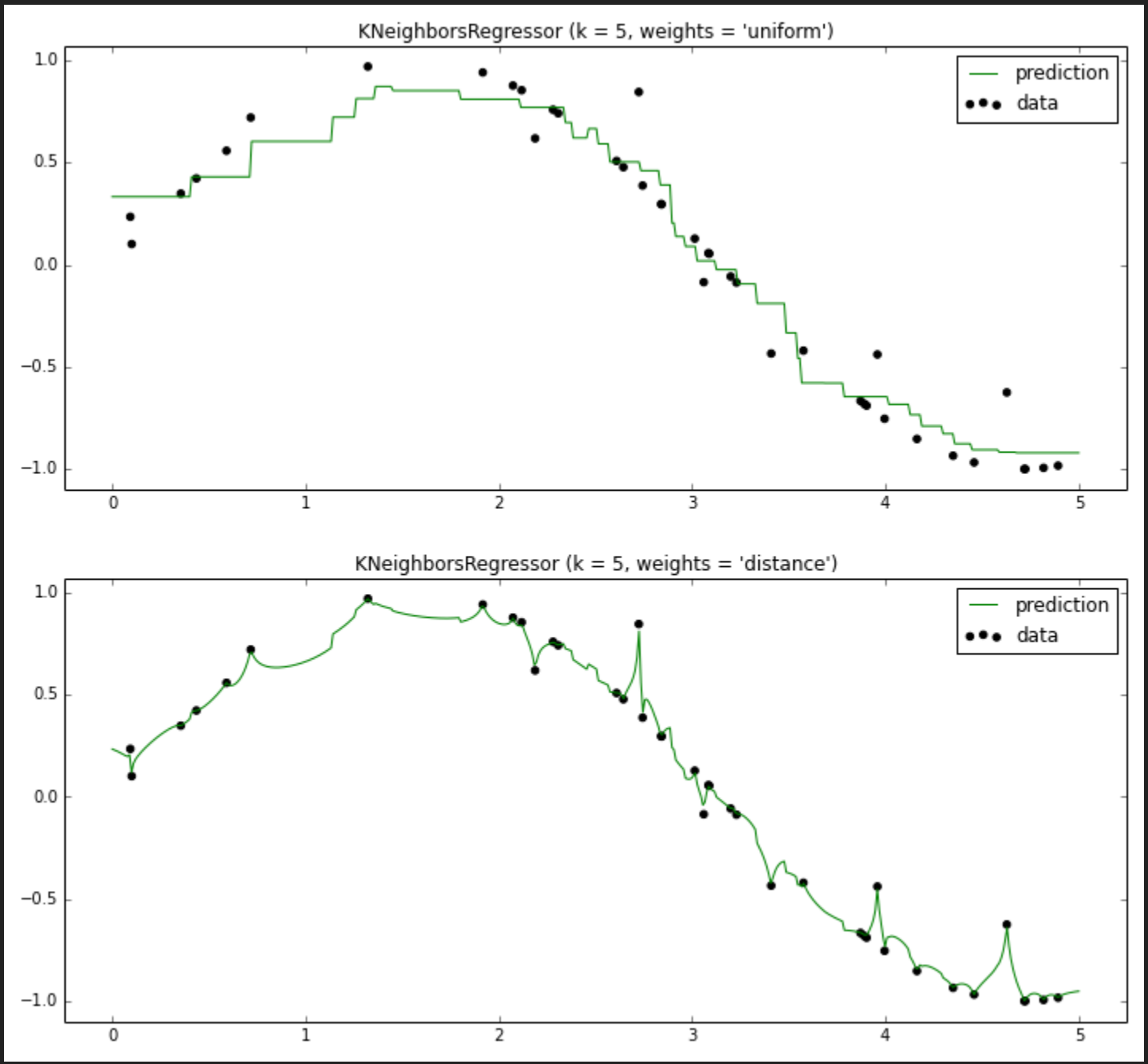
KNN REGRESSOR

Regression with nearest neighbours is done by averaging of output



$$\hat{y} = \frac{1}{k} \sum_{j \in \text{knn}(x)} y_j$$

KNN WITH WEIGHTS

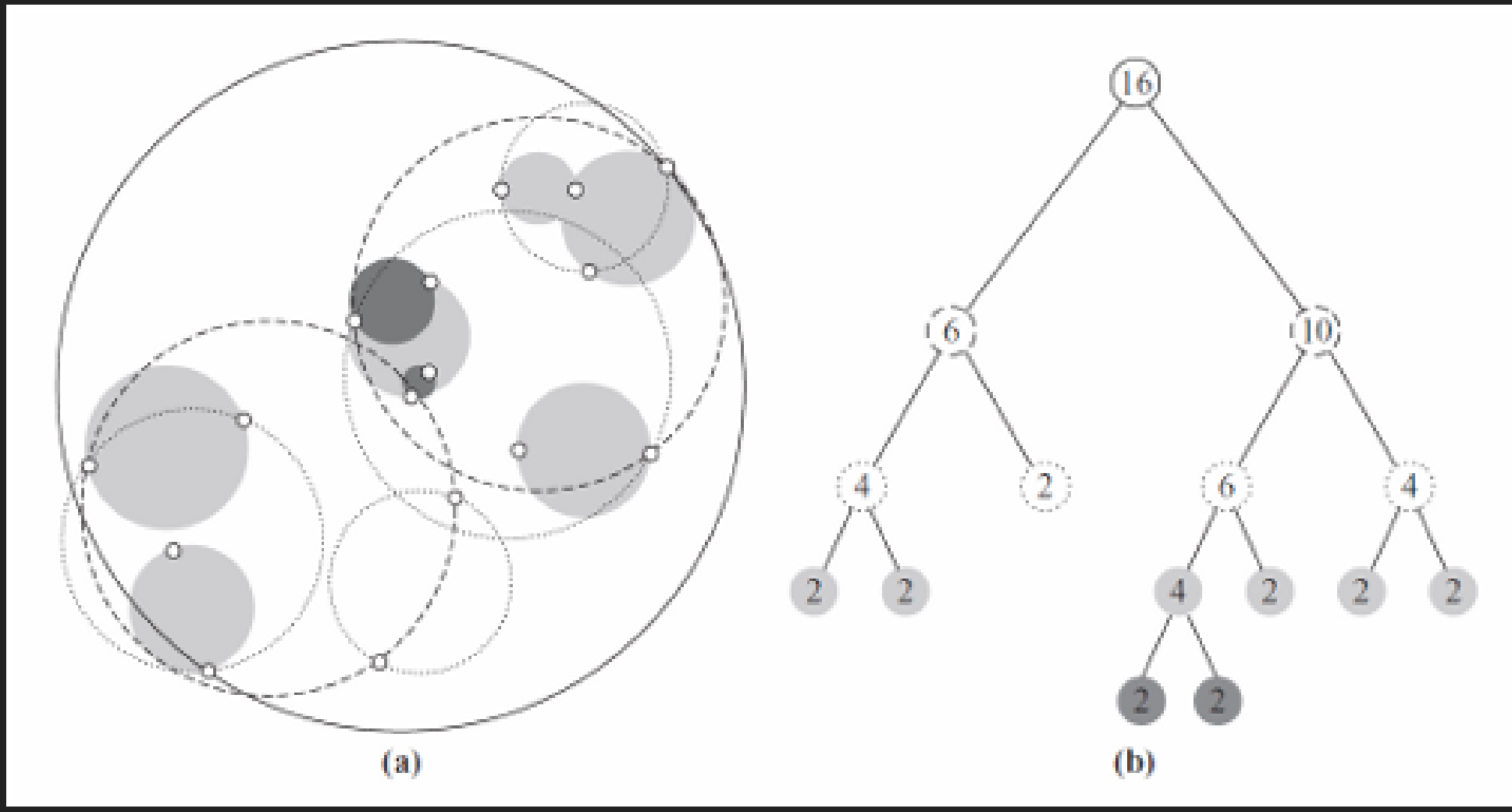

KNeighborsRegressor (k = 5, weights = 'uniform')
prediction
data
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
KNeighborsRegressor (k = 5, weights = 'distance')
prediction
data
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5

COMPUTATIONAL COMPLEXITY

Given that dimensionality of space is d and there are n training samples:

- training time ~ O(save a link to data)
- prediction time: $n \times d$ for each sample

SPACIAL INDEX: BALL TREE



BALL TREE

- training time ~ $O(d \times n \log(n))$
- prediction time ~ $\log(n) \times d$ for each sample

Other option exist: KD-tree.

OVERVIEW OF KNN

1. Awesomely simple classifier and regressor
2. Have too optimistic quality on training data
3. Quite slow, though optimizations exist
4. Hard times with data of high dimensions
5. Too sensitive to scale of features

SENSITIVITY TO SCALE OF FEATURES

Euclidean distance:

$$\rho(x, y)^2 = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2$$

Change scale fo first feature:

$$\rho(x, y)^2 = (10x_1 - 10y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2$$

$$\rho(x, y)^2 \sim 100(x_1 - y_1)^2$$

Scaling of features frequently increases quality.

DISTANCE FUNCTION MATTERS

- Minkowski distance $\rho^p(x, y) = \sum_i (x_i - y_i)^p$
- Canberra $\rho(x, y) = \sum_i \dfrac{|x_i - y_i|}{|x_i| + |y_i|}$
- Cosine metric $\rho(x, y) = \dfrac{< x, y >}{|x|\ |y|}$

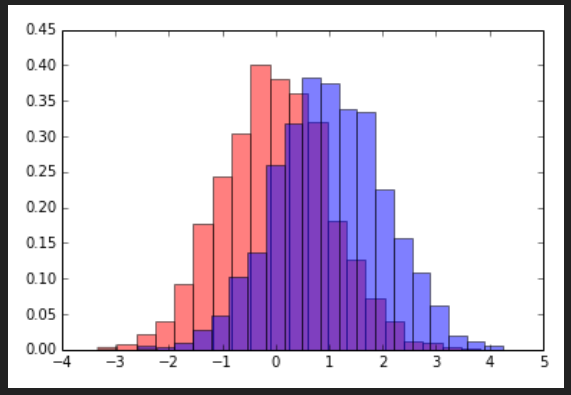
x MINUTES BREAK

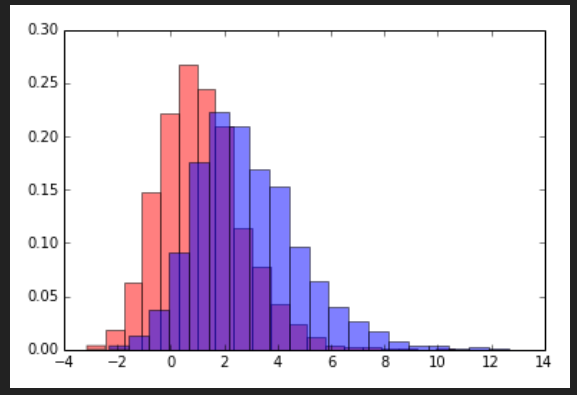
RECAPITULATION

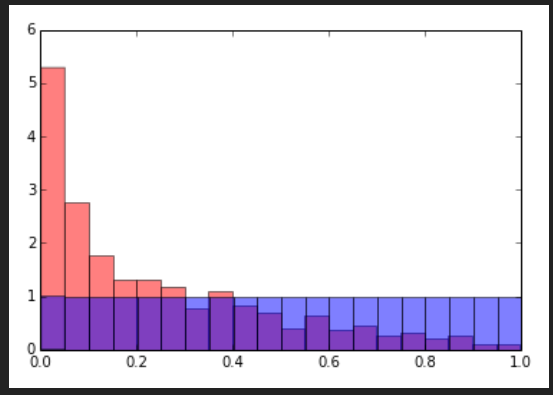
1. Statistical ML: problems
2. ML in HEP
3. k nearest neighbours classifier and regressor.

MEASURING QUALITY OF BINARY CLASSIFICATION

The classifier's output in binary classification is real variable

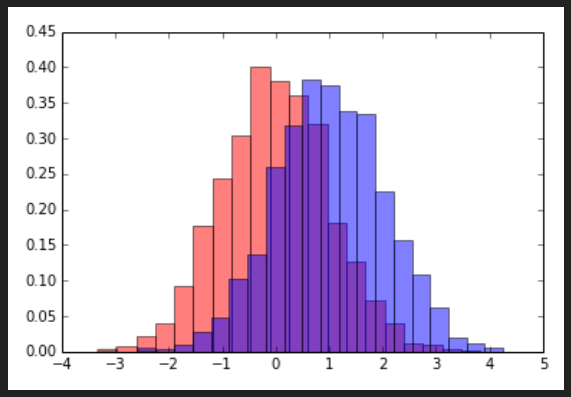


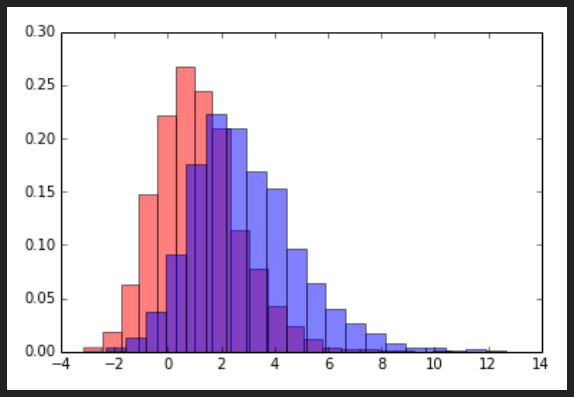


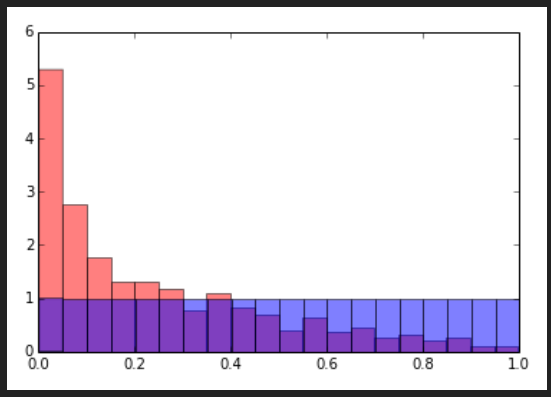


- Which classifier is better?
- All of them are identical

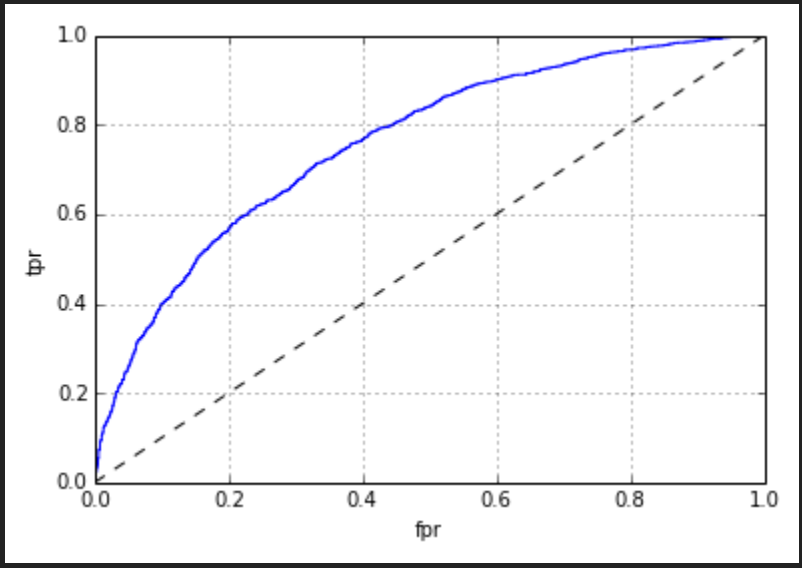
ROC CURVE







These distributions have the same ROC curve:
(ROC curve is passed signal vs passed bck dependency)



ROC CURVE DEMONSTRATION

ROC CURVE

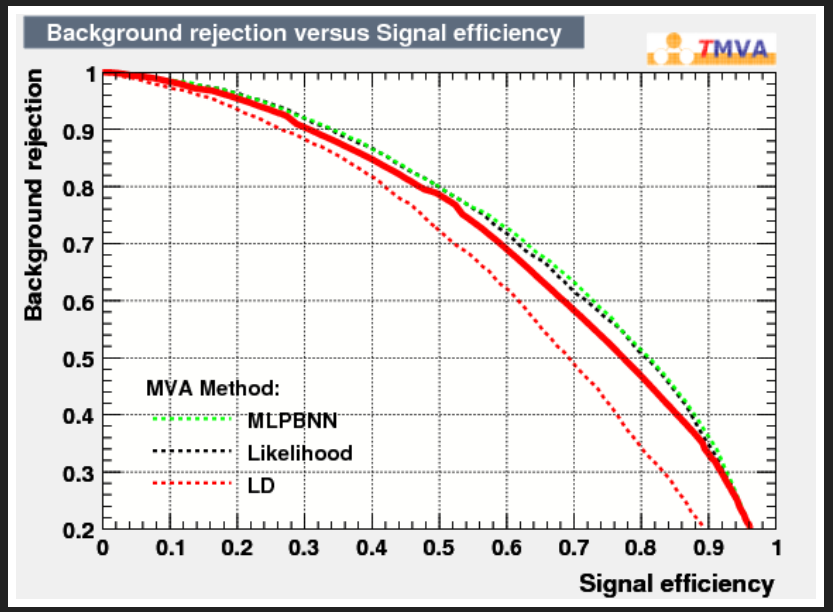
- Contains important information:
  all possible combinations of signal and background efficiencies you may achieve by setting threshold
- Particular values of thresholds (and initial pdfs) don't matter, ROC curve doesn't contain this information
- ROC curve = information about order of events:

```
s s b s b ...  b b s b b
```

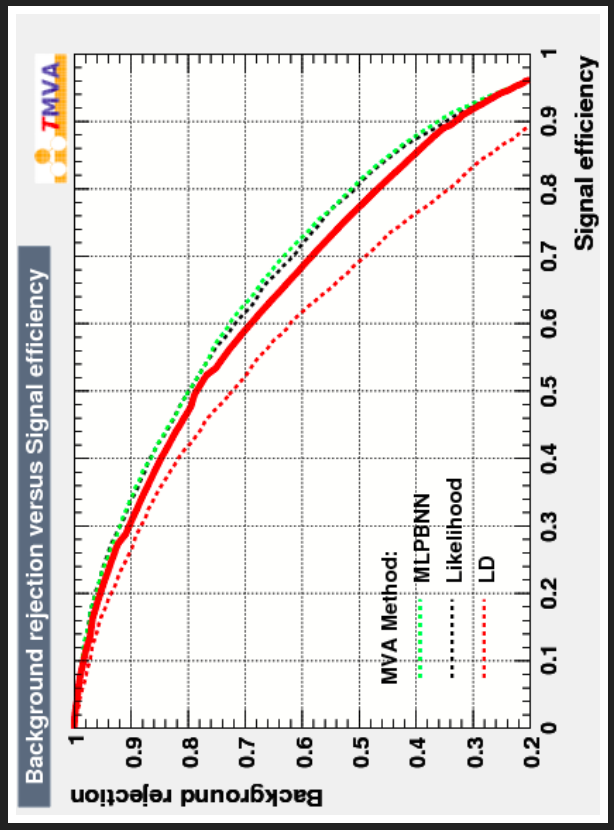
- Comparison of algorithms should be based on information from ROC curve

TERMINOLOGY AND CONVENTIONS

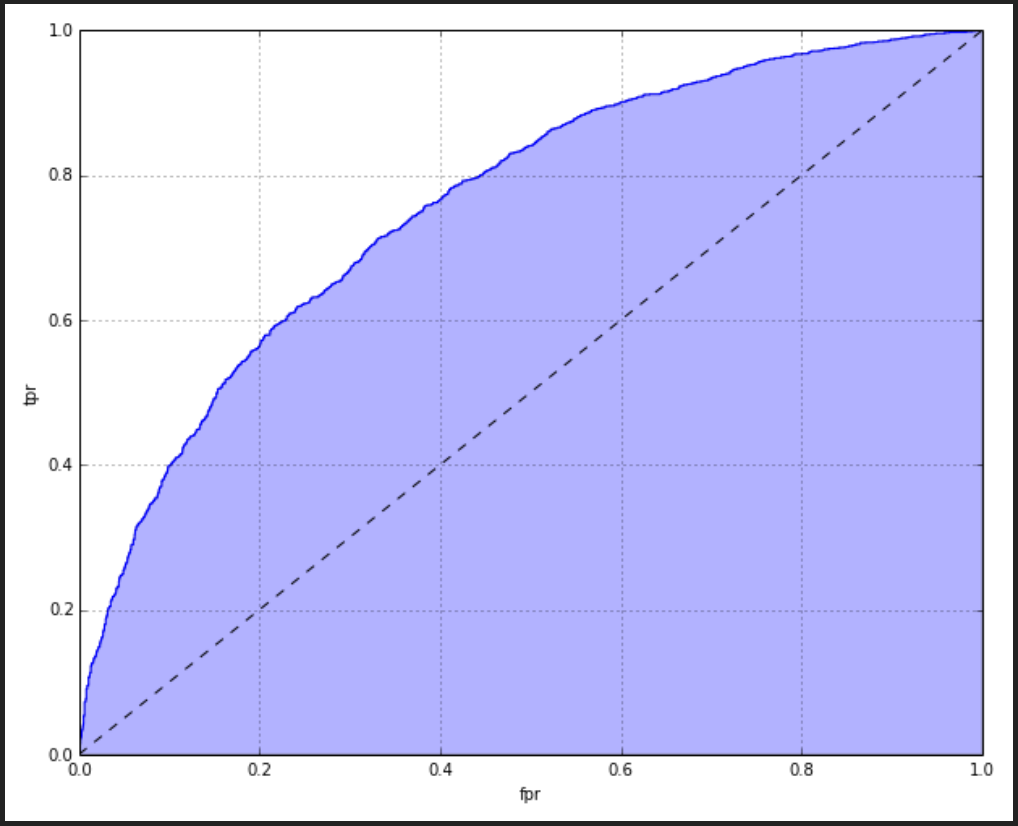
- fpr = background efficiency = b
- tpr = signal efficiency = s



→

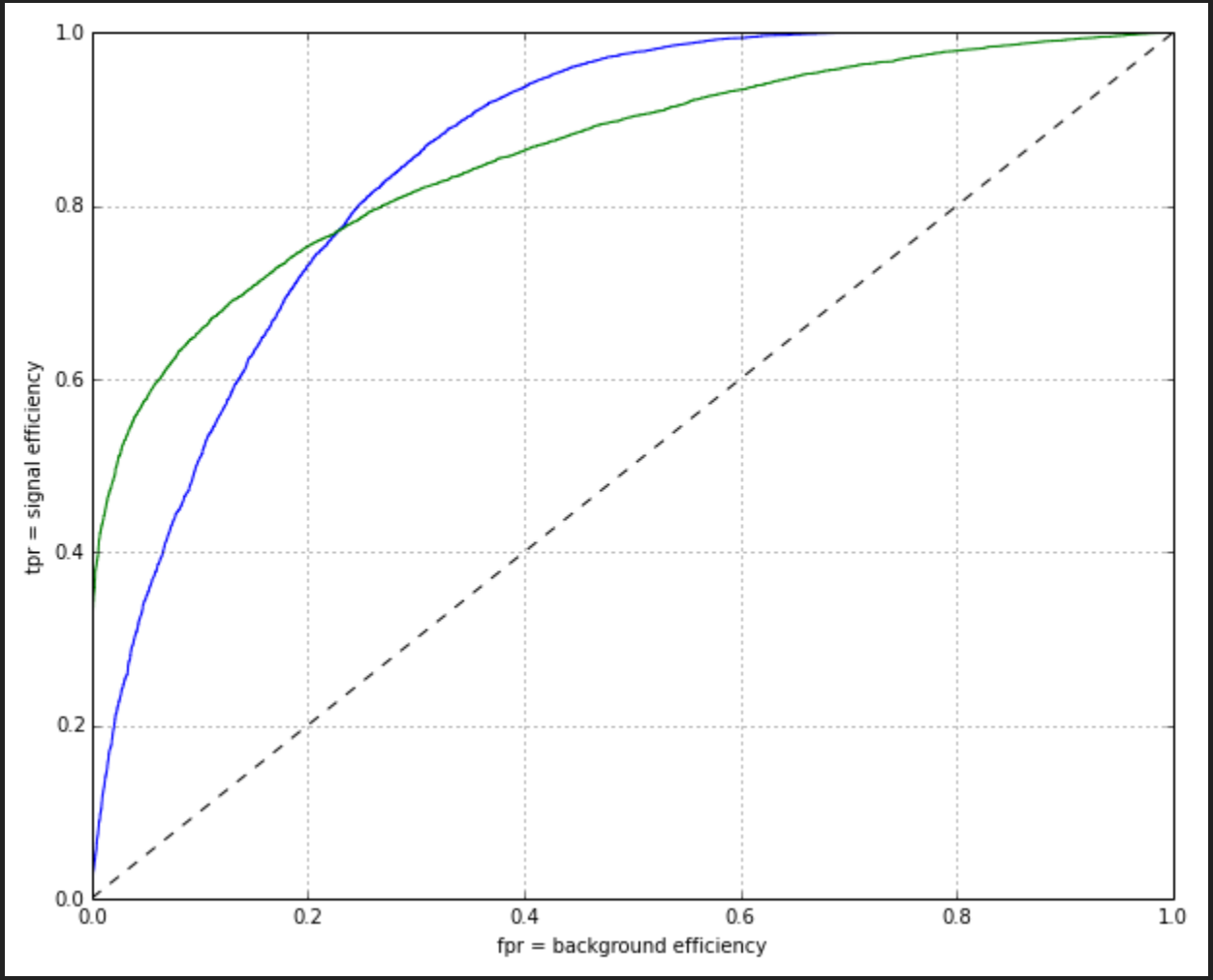


ROC AUC
(AREA UNDER THE ROC CURVE)



$ROC\ AUC = P(x < y)$ where x, y are predictions of random background and signal events.

Which classifier is better for triggers?
(they have the same ROC AUC)



STATISTICAL MACHINE LEARNING

Machine learning we use in practice is based on statistics

1. Main assumption: the data is generated from probabilistic distribution:

$$p(x, y)$$

2. Does there really exist the distribution of people / pages?
3. In HEP these distributions do exist

OPTIMAL CLASSIFICATION. OPTIMAL BAYESIAN CLASSIFIER

Assuming that we know real distributions $p(x, y)$ we reconstruct using Bayes' rule

$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{p(y)p(x|y)}{p(x)}$$

$$\frac{p(y = 1 \mid x)}{p(y = 0 \mid x)} = \frac{p(y = 1)\ p(x \mid y = 1)}{p(y = 0)\ p(x \mid y = 0)}$$

LEMMA (NEYMAN–PEARSON):

The best classification quality is provided by $\frac{p(y = 1 \mid x)}{}$

The best classification quality is provided by $p(y = 0 \mid x)$

(optimal bayesian classifier)

OPTIMAL BINARY CLASSIFICATION

- Optimal bayesian classifier has highest possible ROC curve.
- Since the classification quality depends only on order, $p(y = 1 \mid x)$ gives optimal classification quality too!

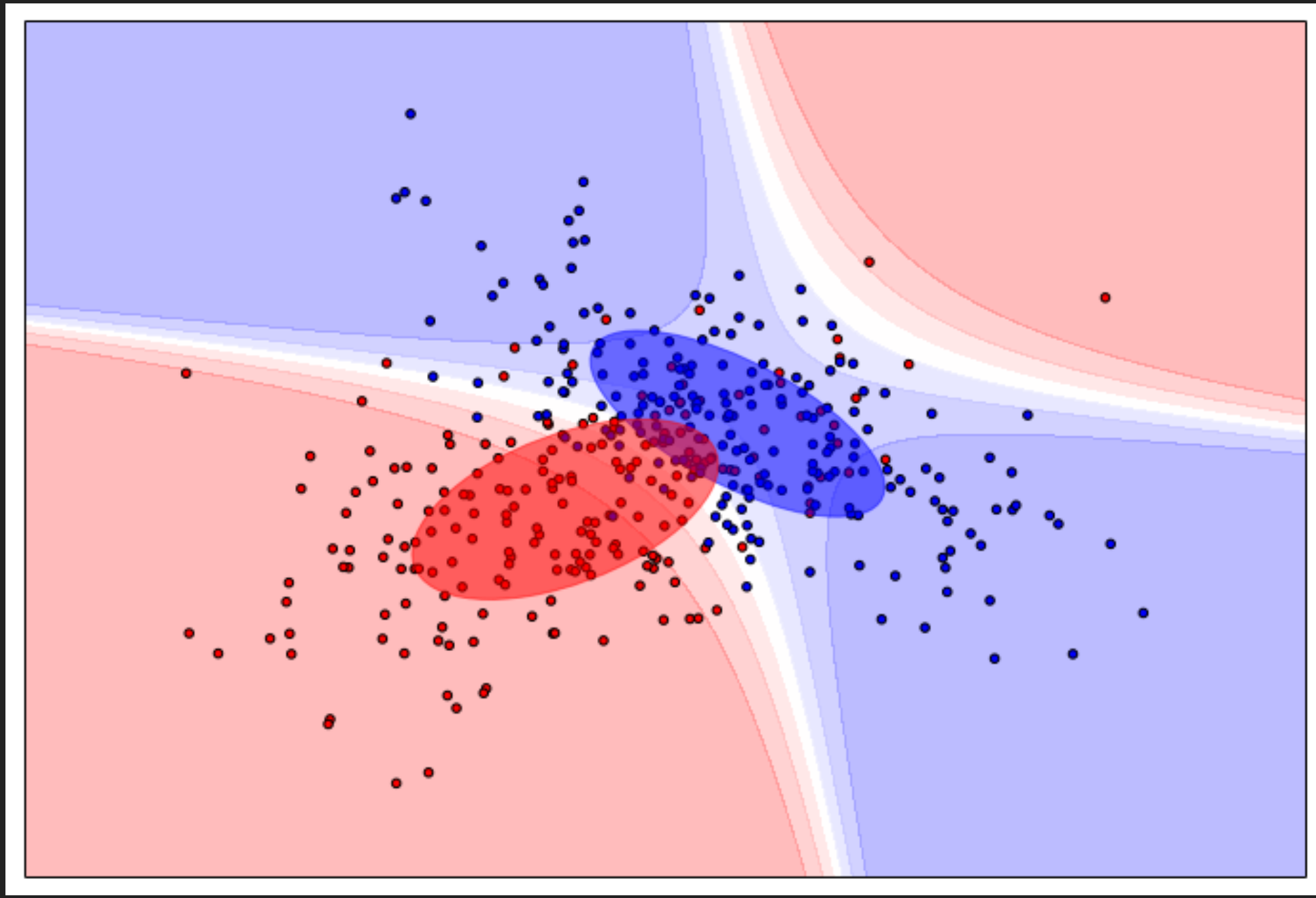
$$\frac{p(y = 1 \mid x)}{p(y = 0 \mid x)} = \frac{p(y = 1)\ p(x \mid y = 1)}{p(y = 0)\ p(x \mid y = 0)}$$

FISHER'S QDA (QUADRATIC DISCRIMINANT ANALYSIS)

Reconstructing probabilities $p(x \mid y = 1), p(x \mid y = 0)$ from data, assuming those are multidimensional normal distributions:

$$p(x \mid y = 0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

$$p(x \mid y = 1) \sim \mathcal{N}(\mu_1, \Sigma_1)$$



QDA COMPLEXITY

n samples, d dimensions

- training takes $O(nd^2 + d^3)$

  computing covariation matrix $O(nd^2)$

  inverting covariation matrix $O(d^3)$

- prediction takes $O(d^2)$ for each sample

$$f(x) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

QDA

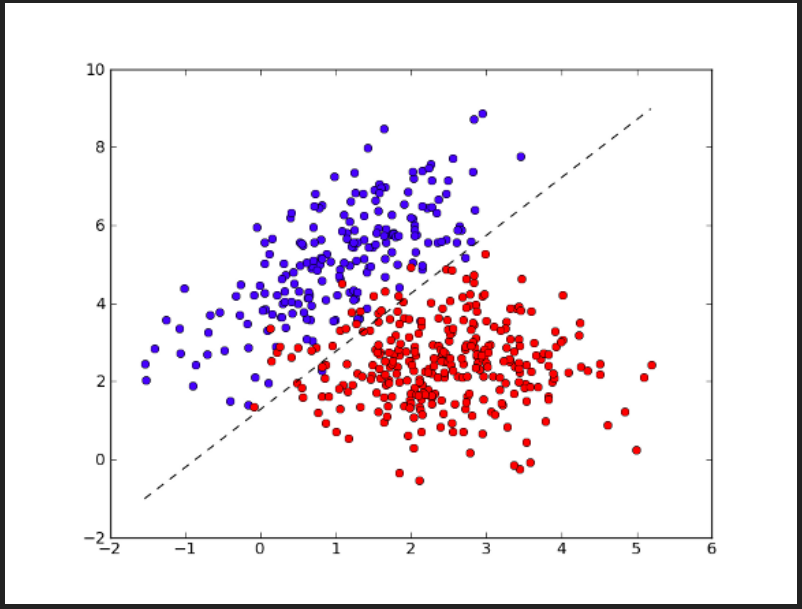
- simple decision rule
- fast prediction
- many parameters to reconstruct in high dimensions
- data almost never has gaussian distribution

WHAT ARE THE PROBLEMS WITH GENERATIVE APPROACH?

Generative approach: trying to reconstruct $p(x, y)$, then use it to predict.

- Real life distributions hardly can be reconstructed
- Especially in high-dimensional spaces
- So, we switch to discriminative approach: guessing $p(y|x)$

LINEAR DECISION RULE



Decision function is linear:

$$d(x) =< w, x > +w_0$$

$$\begin{cases} d(x) > 0, & \text{class } +1 \\ d(x) < 0, & \text{class } -1 \end{cases}$$

This is parametric model (finding parameters w, w_0).

FINDING OPTIMAL PARAMETERS

- A good initial guess: get such w, w_0, that error of classification is minimal ([true] = 1, [false] = 0):

$$\mathcal{L} = \sum_{i \in \text{events}} [y_i \neq \text{sgn}(d(x_i))]$$

- Discontinuous optimization (arrrrgh!)
- Let's make decision rule **smooth**

$$p_{+1}(x) = f(d(x))$$
$$p_{-1}(x) = 1 - p_{+1}(x)$$

$$\begin{cases} f(0) = 0.5 \\ f(x) > 0.5 & if\ x > 0 \\ f(x) < 0.5 & if\ x < 0 \end{cases}$$

LOGISTIC FUNCTION

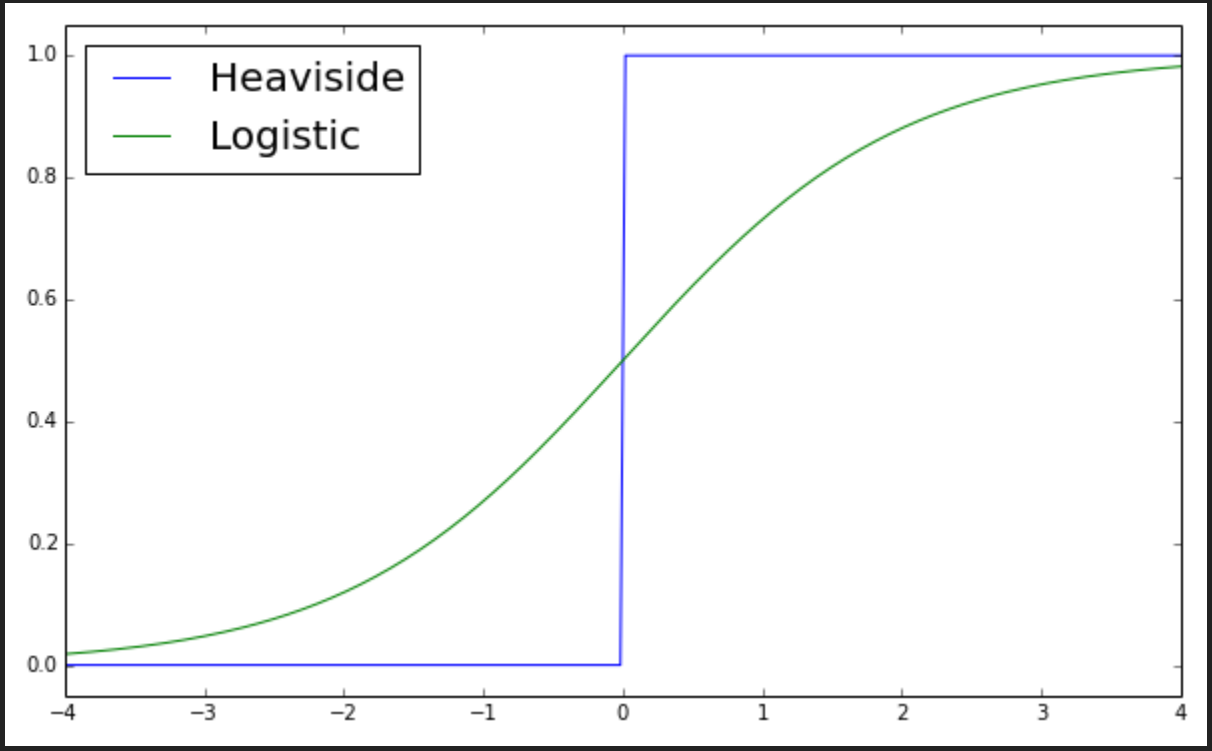
a smooth step rule.

$$\sigma(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

PROPERTIES

1. monotonic, $\sigma(x) \in (0, 1)$
2. $\sigma(x) + \sigma(-x) = 1$
3. $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
4. $2\,\sigma(x) = 1 + \tanh(x/2)$

LOGISTIC FUNCTION



LOGISTIC REGRESSION

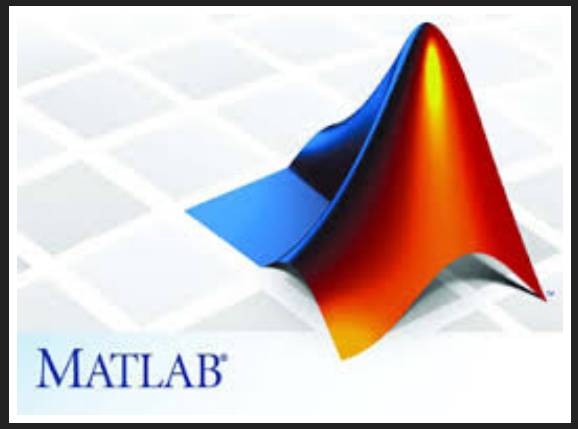
Optimizing log-likelihood (with probabilities obtained with logistic function)

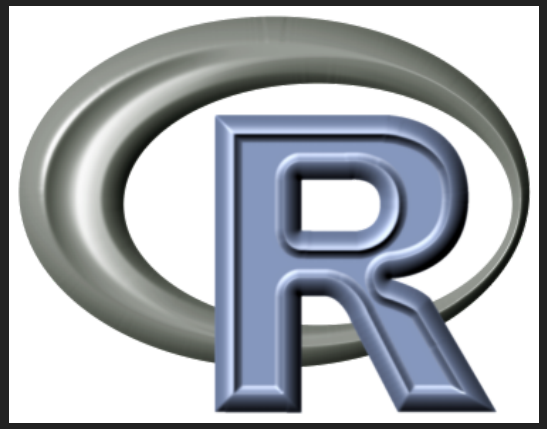
$$
\begin{aligned}
d(x) &= < w, x > + w_0 \\
p_{+1}(x) &= \sigma(d(x)) \\
p_{-1}(x) &= \sigma(-d(x))
\end{aligned}
$$

$$
\mathcal{L} = \frac{1}{N} \sum_{i \in \text{events}} -\ln(p_{y_i}(x_i)) = \frac{1}{N} \sum_i L(x_i, y_i) \to \min
$$

Exercise: find expression and build plot for $L(x_i, y_i)$

DATA SCIENTIST PIPELINE





1. Experiments in appropriate high-level language or environment
2. After experiments are over — implement final algorithm in low-level language (C++, CUDA, FPGA)

Second point is not always needed.

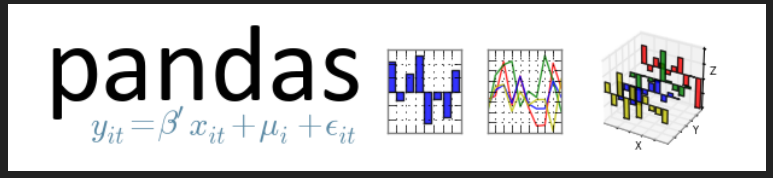
SCIENTIFIC PYTHON

NumPy

vectorized computations in python

Matplotlib

for drawing



Pandas

for data manipulation and analysis (based on NumPy)

SCIENTIFIC PYTHON

Scikit-learn

most popular library for machine learning

Scipy

libraries for science and engineering

Root_numpy

convenient way to work with ROOT files

THE END