Machine Learning in High Energy Physics

Lectures 1 & 2

Alex Rogozhnikov

Lund, MLHEP 2016

Intro notes

- two tracks:
  - introductory course (this one)
  - advanced track: Mon, Tue, Wed, then two tracks are merged
- Introductory track:
  - two lectures and two practice seminars on each day
- Kaggle challenges
  - 'Triggers' — only for advanced track, lasts for 3 days
  - 'Higgs' — for both tracks, lasts for 7 days

Intro notes — 2

- chat rooms
  - gitter
  - if you want to share something between teams — please do it publicly (via chat)
- cluster access
- repository
- glossary is in the repository

What is Machine Learning about?

- a method of teaching computers to make and improve predictions or behaviors based on some data?
- a field of computer science, probability theory, and optimization theory which allows complex tasks to be solved for which a logical/procedural approach would not be possible or feasible?
- a type of AI that provides computers with the ability to learn without being explicitly programmed?
- somewhat in between of statistics, AI, optimization theory, signal processing and pattern matching?

What is Machine Learning about

Inference of statistical dependencies which give us ability to predict

What is Machine Learning about

Inference of statistical dependencies which give us ability to predict

Data is cheap, knowledge is precious

Machine Learning is used in

- search engines
- spam detection
- security: virus detection, DDOS defense
- computer vision and speech recognition
- market basket analysis, customer relationship management (CRM), churn prediction
- credit scoring / insurance scoring, fraud detection
- health monitoring
- traffic jam prediction, self-driving cars
- advertisement systems / recommendation systems / news clustering

Machine Learning is used in

- search engines
- spam detection
- security: virus detection, DDOS defense
- computer vision and speech recognition
- market basket analysis, customer relationship management (CRM), churn prediction
- credit scoring / insurance scoring, fraud detection
- health monitoring
- traffic jam prediction, self-driving cars
- advertisement systems / recommendation systems / news clustering
- and hundreds more

Machine Learning in High Energy Physics

- Triggers (LHCb, CMS to join soon)
- Particle identification
- Calibration
- Tagging
- Stripping line
- Analysis

Machine Learning in High Energy Physics

- Triggers (LHCb, CMS to join soon)
- Particle identification
- Calibration
- Tagging
- Stripping line
- Analysis

On each stage different data is used and different information is inferred, but the ideas beyond are quite similar.

General notion

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

$$x_i, y_i$$

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

features = observables = variables

Classification problem

$y_i \in Y$, where Y is finite set of labels.

Examples

- particle identification based on information about track:

$$\begin{aligned} x_i = & \ (p, \eta, E, charge, PV\chi^2, FlightTime) \\ Y = & \ \{\text{electron, muon, pion, ... }\} \end{aligned}$$

- binary classification:

Regression problem

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

Examples:

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

Regression problem

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

Examples:

- predicting price of a 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

Classification based on nearest neighbours

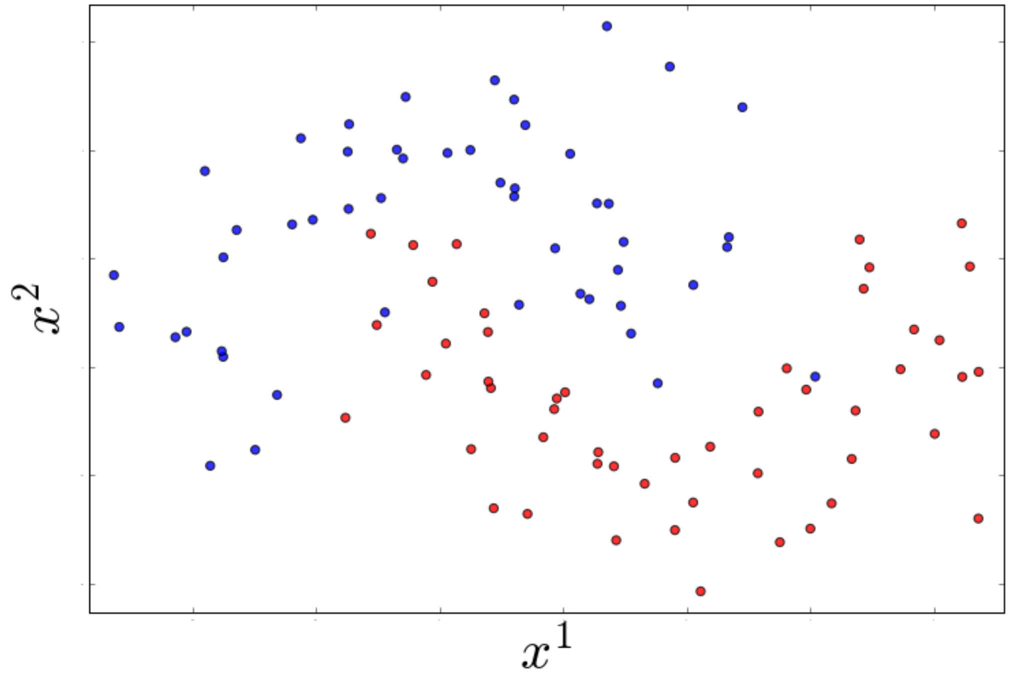
Given training set of objects and their labels $\{x_i, y_i\}$ we predict the label for the new observation x:

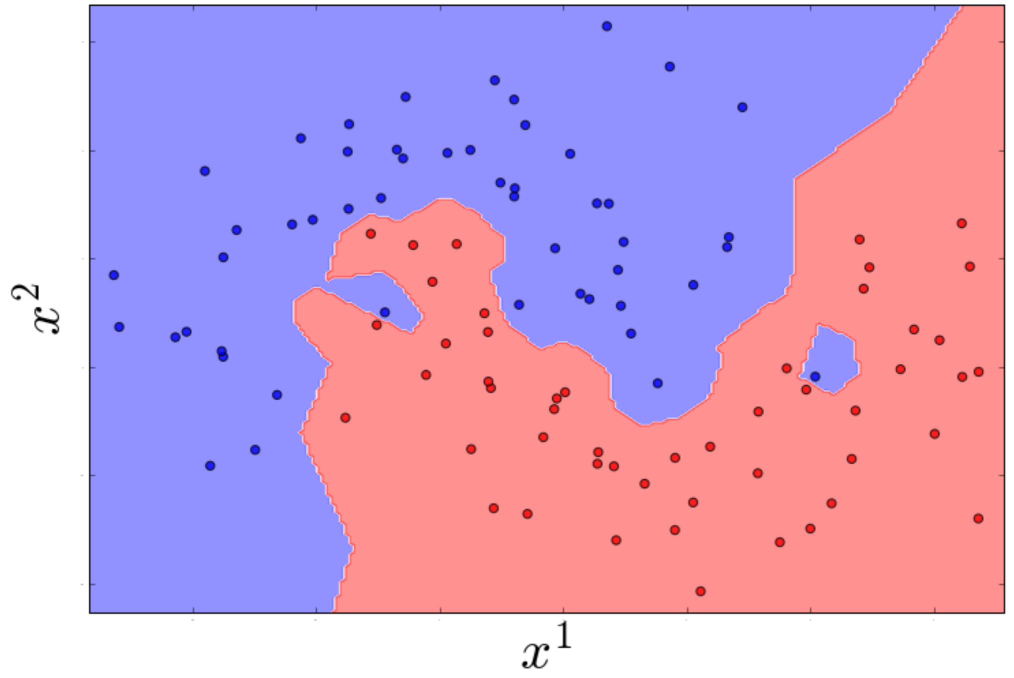
$$\hat{y} = y_j, \qquad j = \arg\min_i \rho(x, x_i)$$

Here and after $\rho(x, \tilde{x})$ is the distance in the space of features.

Visualization of decision rule

Consider a classification problem with 2 features:

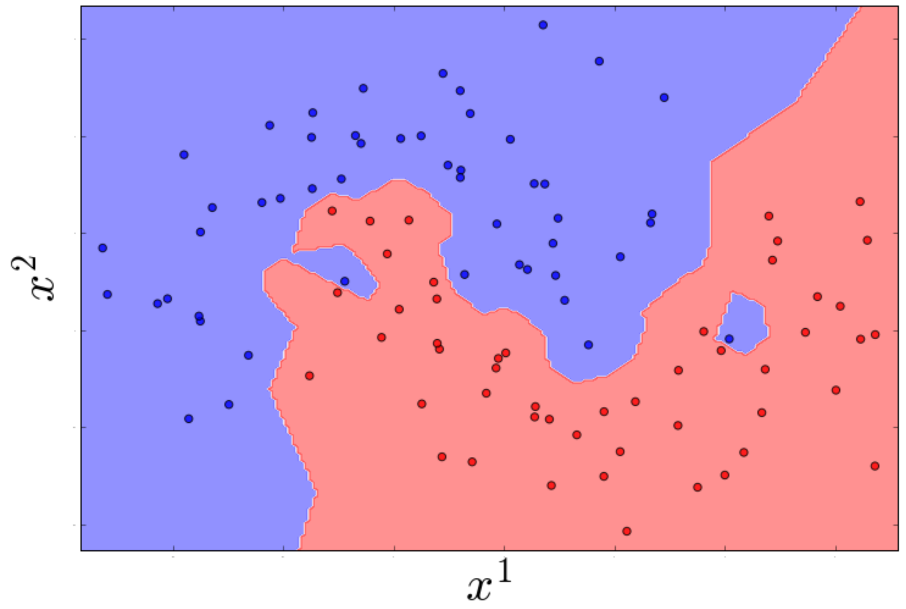


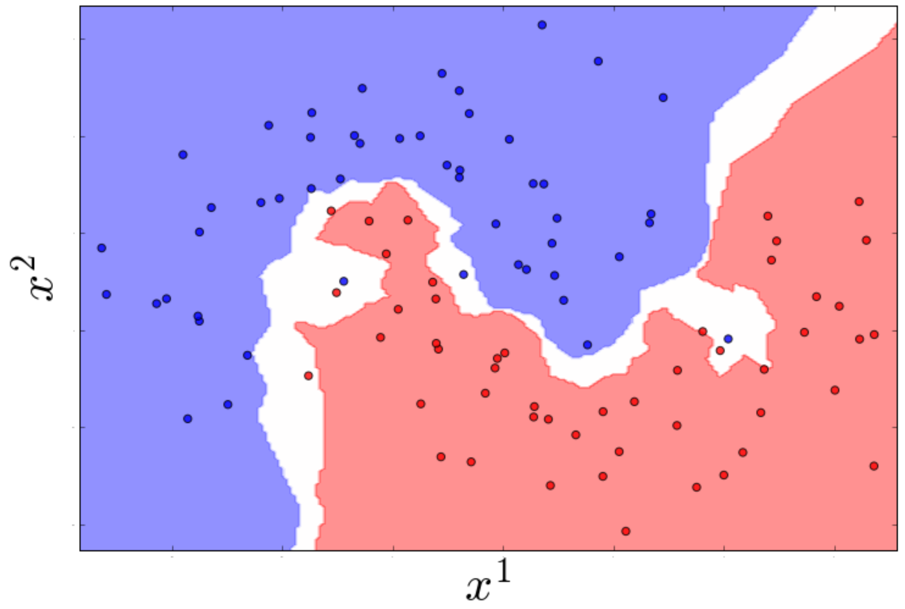


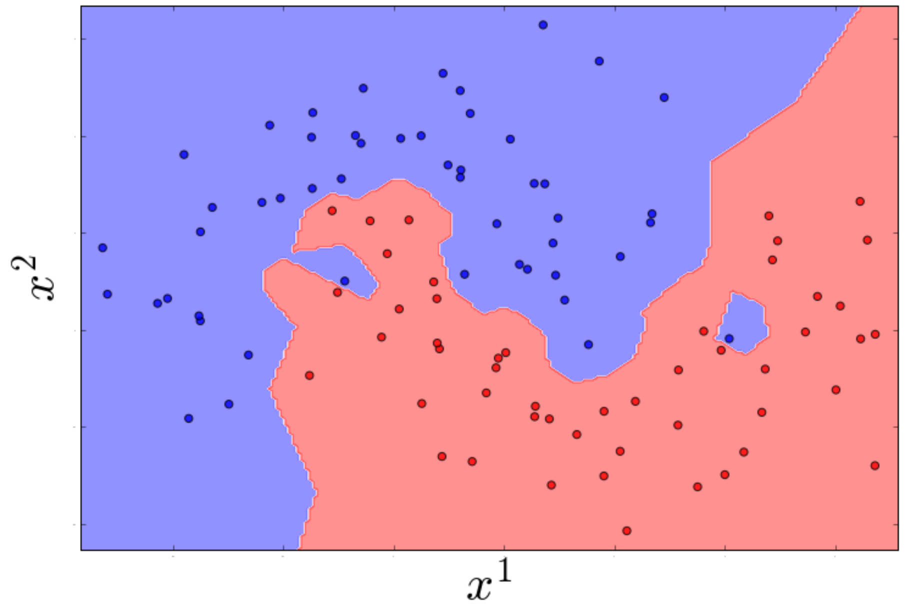
k Nearest Neighbours (kNN)

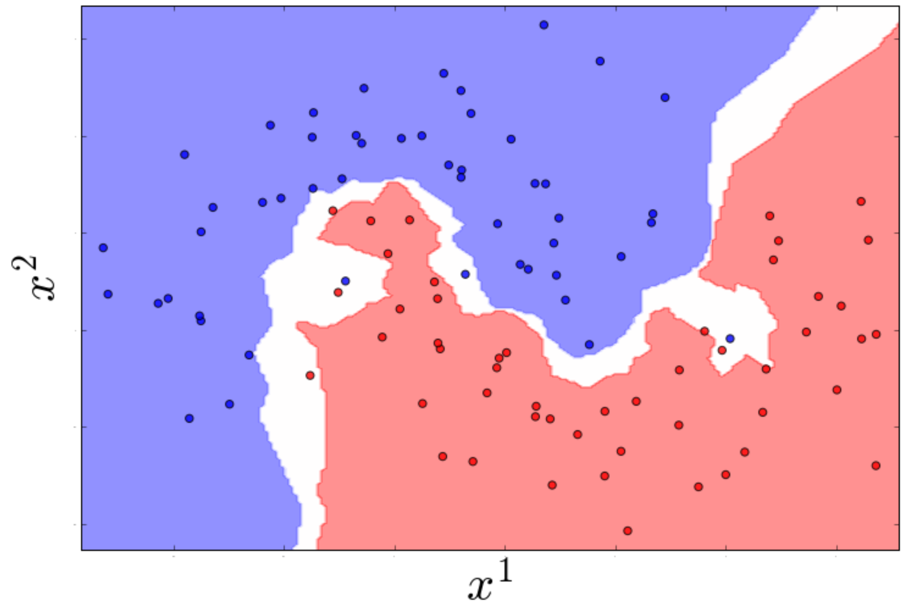
A better way is to use k neighbors:

$$p_{\tilde{y}}(x) = \frac{\text{\# of knn events of } x \text{ in class } \tilde{y}}{k}$$


x^2
x^1


x^2
x^1


x^2
x^1


x^2
x^1

Overfitting

What is the quality of classification on training dataset when $k = 1$?

Overfitting

What is the quality of classification on training dataset when $k = 1$?

- answer: it is ideal (closest neighbor is event itself)

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 $k > 1$

Overfitting

What is the quality of classification on training dataset when $k = 1$?

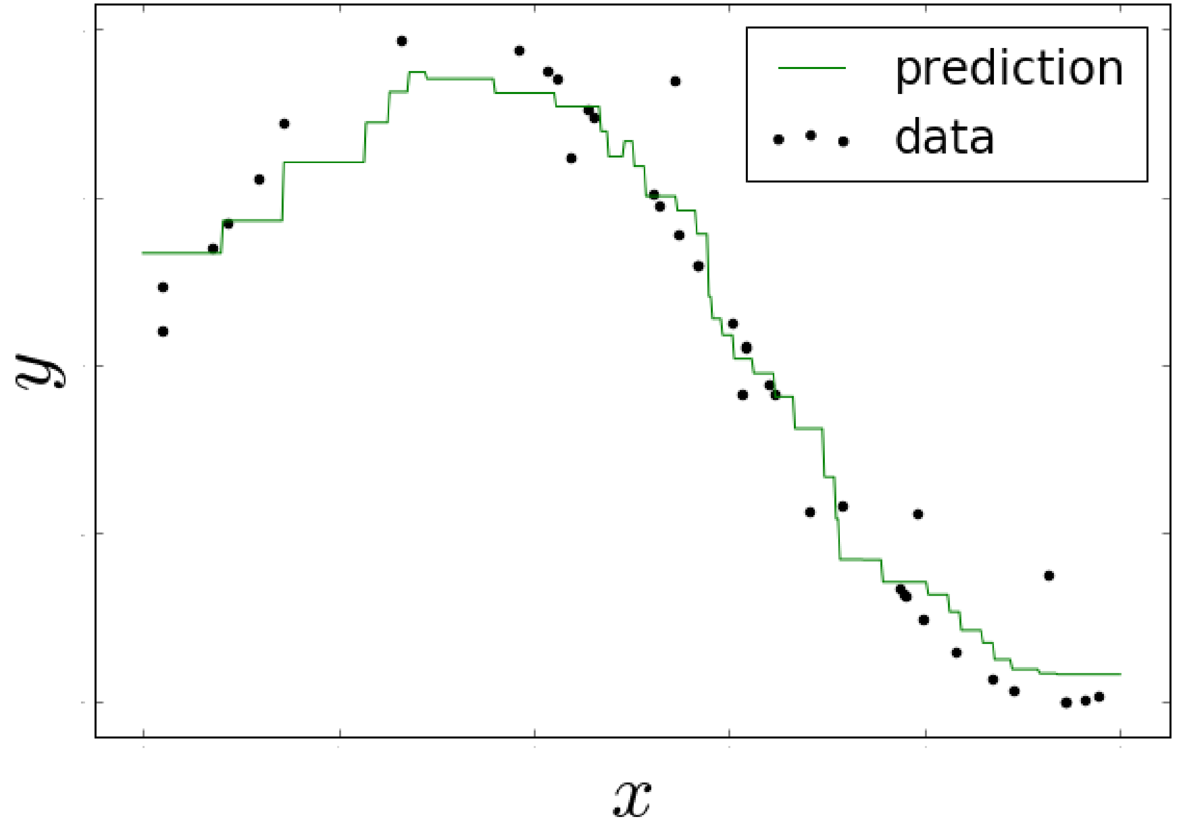
- answer: it is ideal (closest neighbor is event itself)
- quality is lower when $k > 1$

- 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 *an overfitting* (more details tomorrow)

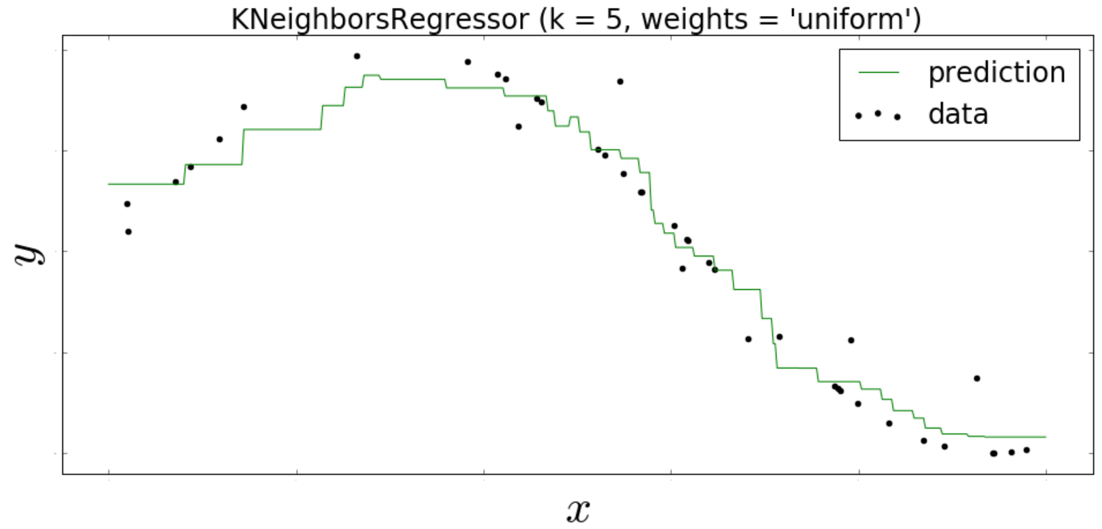
Regression using kNN

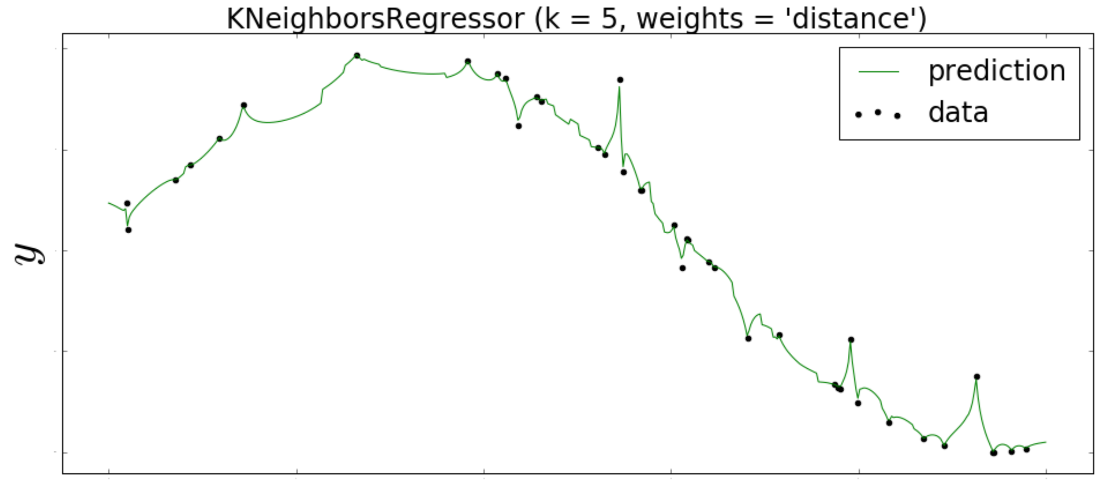
Regression with nearest neighbours is done by averaging of output



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

kNN with weights





Average neighbours' output with weights:

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

the closer the neighbour, the higher weights of its contribution:

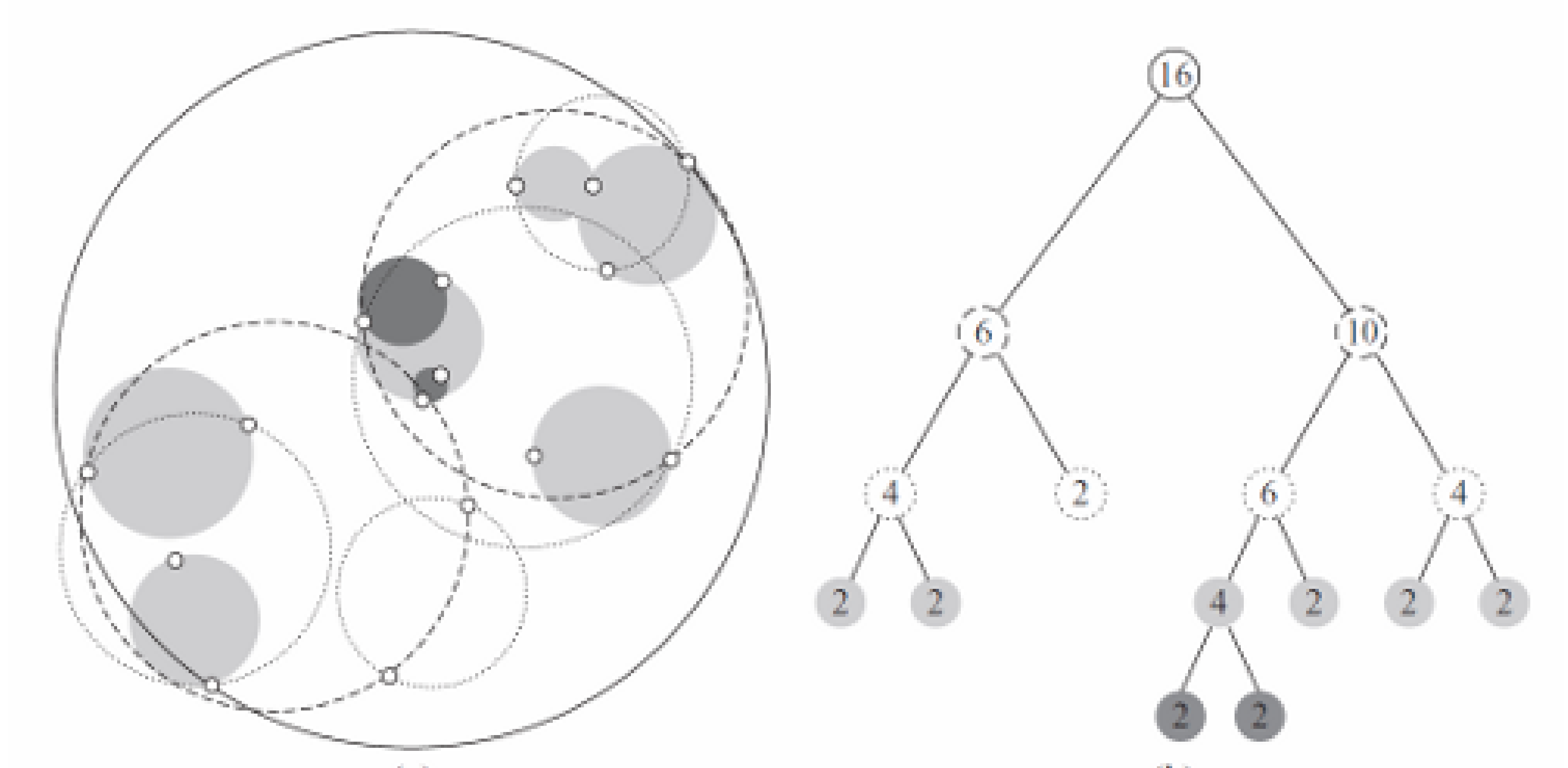
$$w_j = 1/\rho(x, x_j)$$

Computational complexity

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

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

Spacial index: ball tree



(a) (b)

Ball tree

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

Other option exists: KD-tree.

Overview of kNN

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

Sensitivity to scale of features

Euclidean distance:

$$\rho(x, \tilde{x})^2 = (x_1 - \tilde{x}_1)^2 + (x_2 - \tilde{x}_2)^2 + \cdots + (x_d - \tilde{x}_d)^2$$

Sensitivity to scale of features

Euclidean distance:

$$\rho(x, \tilde{x})^2 = (x_1 - \tilde{x}_1)^2 + (x_2 - \tilde{x}_2)^2 + \cdots + (x_d - \tilde{x}_d)^2$$

Change scale of first feature:

$$\begin{aligned}\rho(x, \tilde{x})^2 &= (10x_1 - 10\tilde{x}_1)^2 + (x_2 - \tilde{x}_2)^2 + \cdots + (x_d - \tilde{x}_d)^2 \\ \rho(x, \tilde{x})^2 &\sim 100\,(x_1 - \tilde{x}_1)^2\end{aligned}$$

Sensitivity to scale of features

Euclidean distance:

$$\rho(x, \tilde{x})^2 = (x_1 - \tilde{x}_1)^2 + (x_2 - \tilde{x}_2)^2 + \dots + (x_d - \tilde{x}_d)^2$$

Change scale of first feature:

$$\begin{aligned} \rho(x, \tilde{x})^2 &= (10x_1 - 10\tilde{x}_1)^2 + (x_2 - \tilde{x}_2)^2 + \dots + (x_d - \tilde{x}_d)^2 \\ \rho(x, \tilde{x})^2 &\sim 100\,(x_1 - \tilde{x}_1)^2 \end{aligned}$$

Scaling of features frequently increases quality.

Distance function matters

- Minkowski distance $\rho(x, \tilde{x})^p = \sum_l (x_l - \tilde{x}_l)^p$
- Canberra

$$\rho(x, \tilde{x}) = \sum_l \frac{|x_l - \tilde{x}_l|}{|x_l| + |\tilde{x}_l|}$$

- Cosine metric

$$\rho(x, \tilde{x}) = \frac{< x, \tilde{x} >}{|x| \ |\tilde{x}|}$$

Problems with high dimensions

With higher dimensions $d >> 1$ the neighboring points are further.

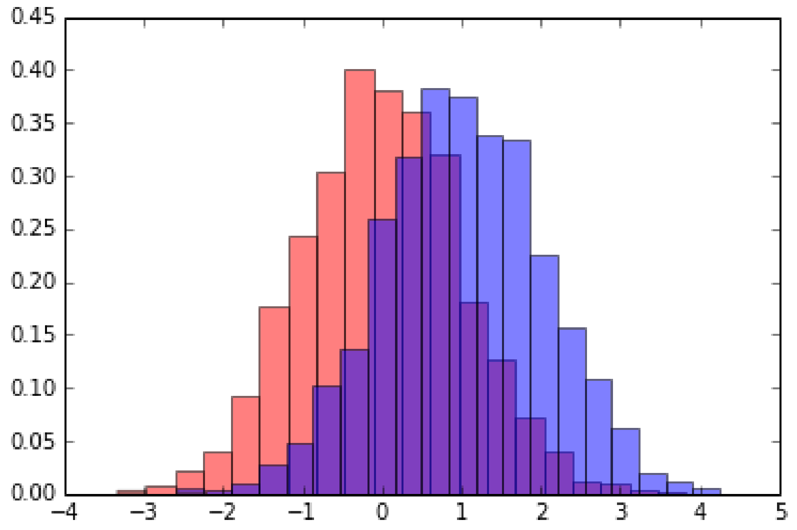
Example: consider n training data points being distributed unformly in the unit cube:

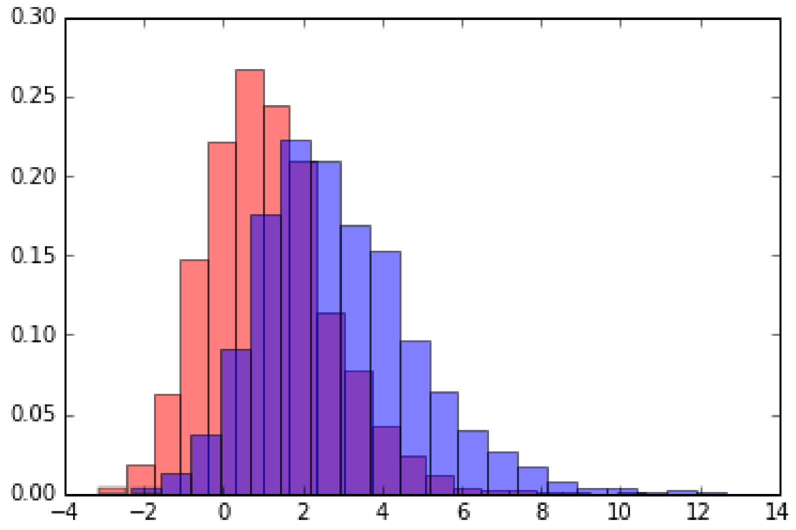
- expected number of point in the ball of size r is proportional to the r^d
- to collect the same amount on neighbors, we need to put $r = \text{const}^{1/d} \to 1$

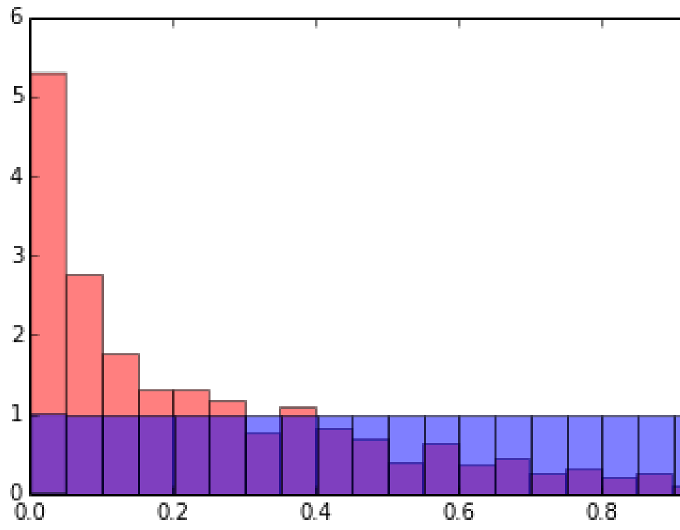
kNN suffers from *curse of dimensionality*.

Measuring quality of binary classification

The classifier's output in binary classification is real variable (say, signal is blue and background is red)



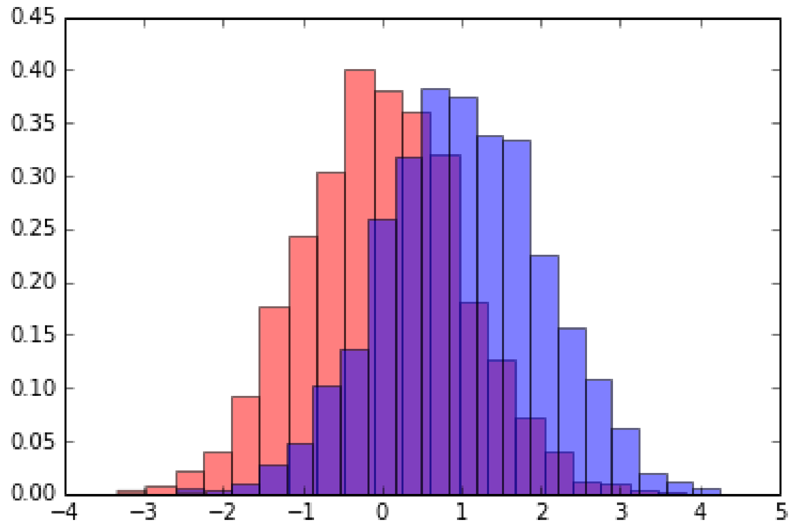


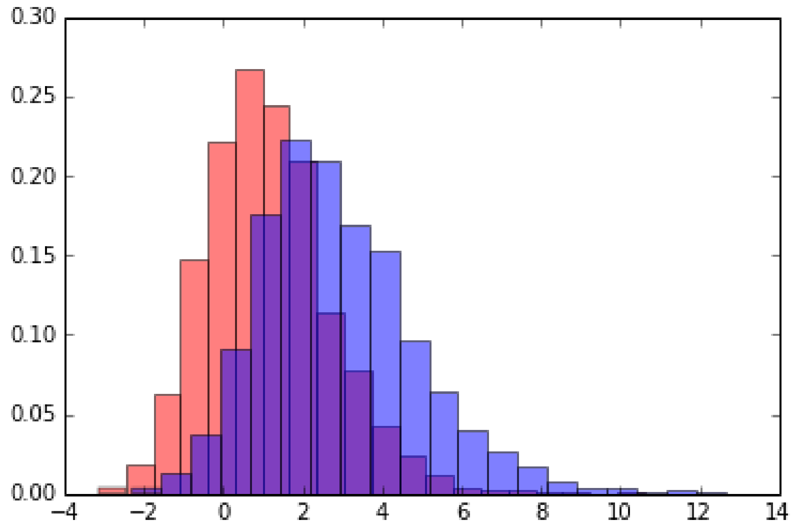


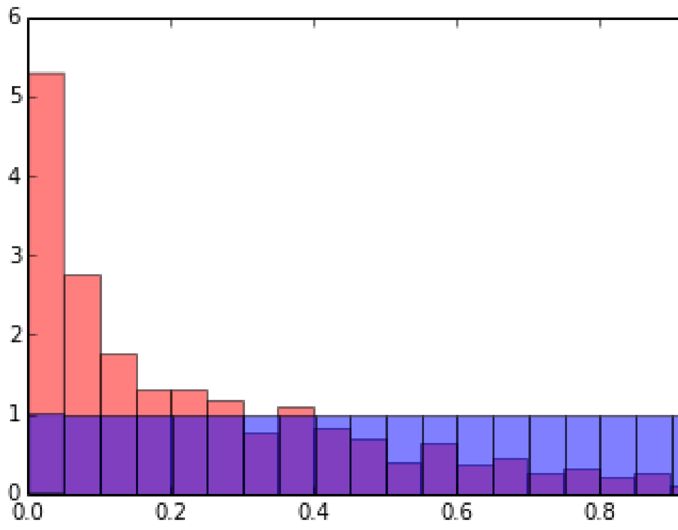
- Which classifier provides better discrimination?

Measuring quality of binary classification

The classifier's output in binary classification is real variable (say, signal is blue and background is red)



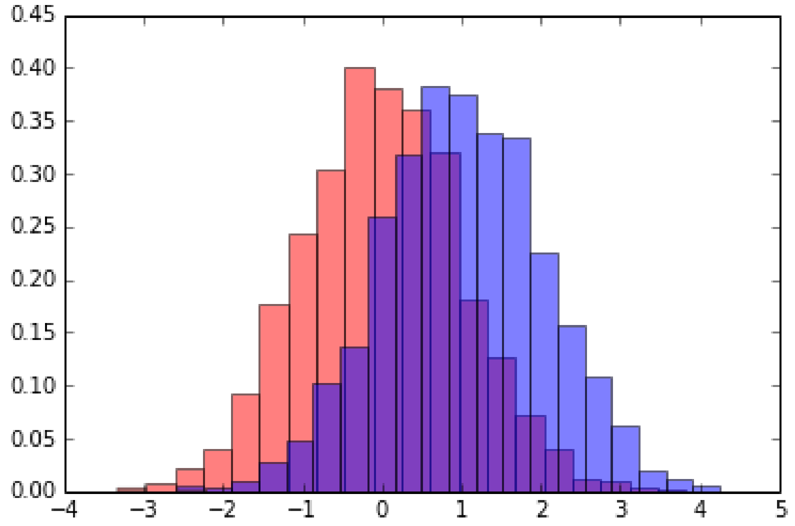


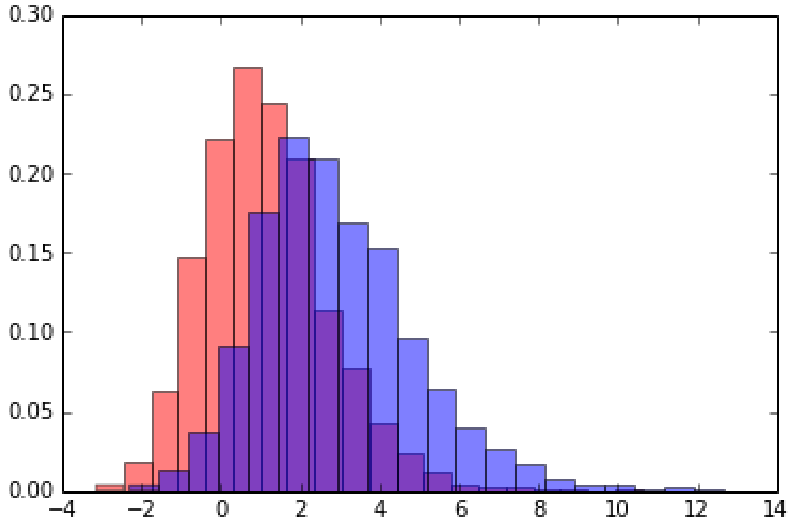


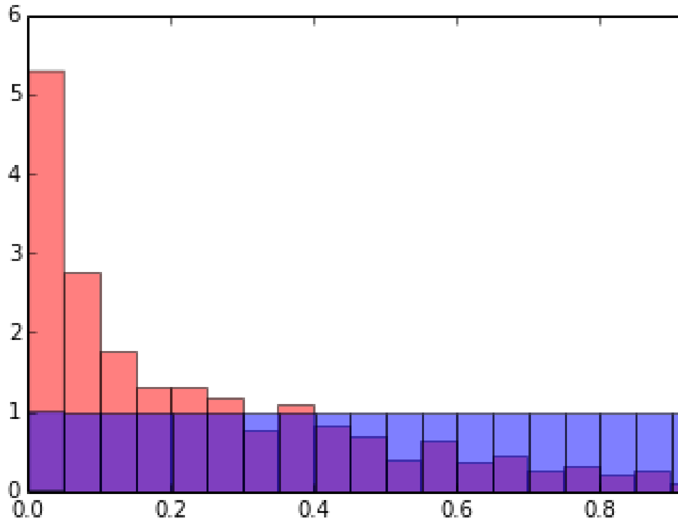
- Which classifier provides better discrimination?

ROC curve demonstration

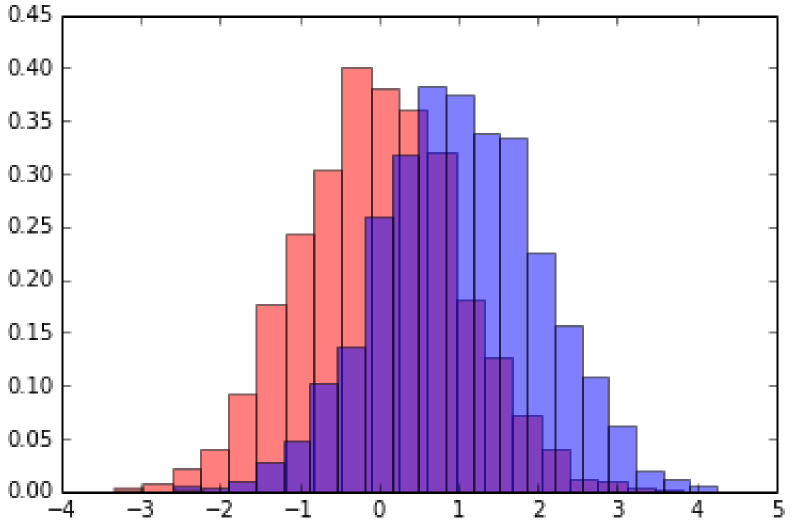
ROC curve

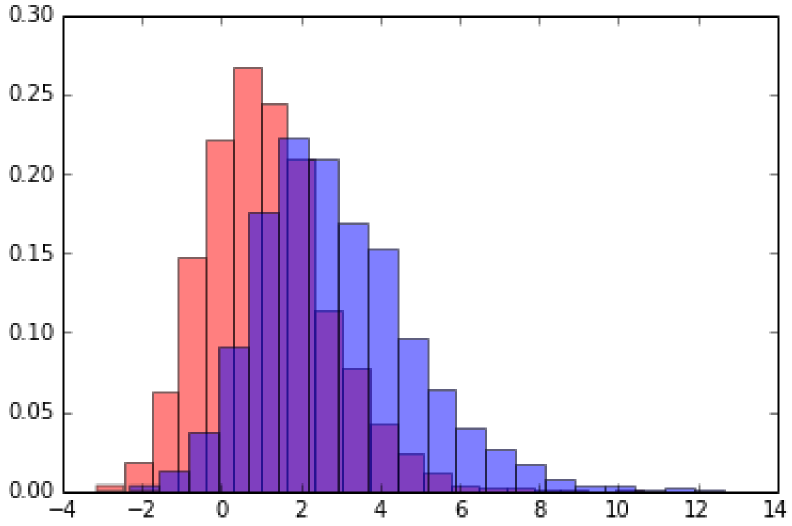


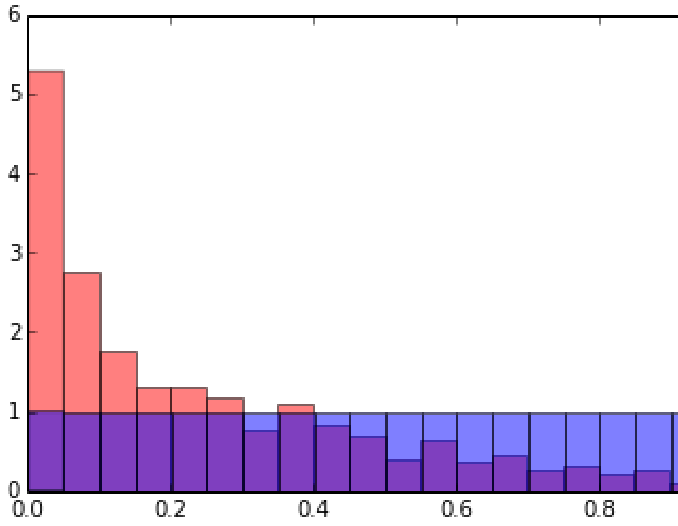




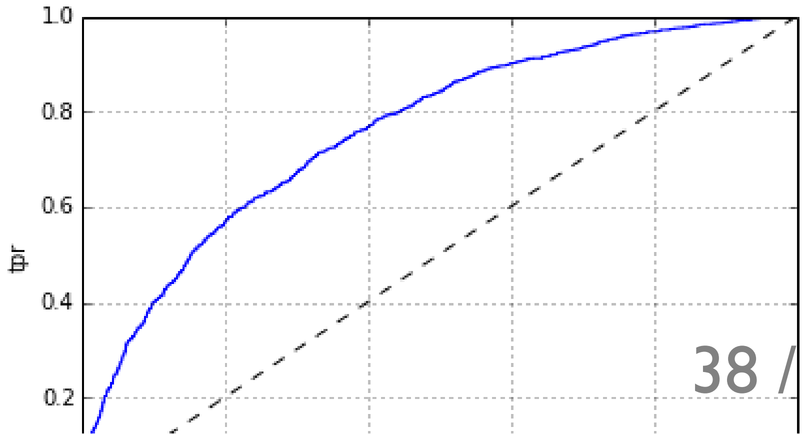
ROC curve







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



ROC curve

- Defined only for binary classification
- Contains important information:
  all possible combinations of signal and background efficiencies you may achieve by setting threshold

ROC curve

- Defined only for binary classification
- 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

- Defined only for binary classification
- 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:

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

ROC curve

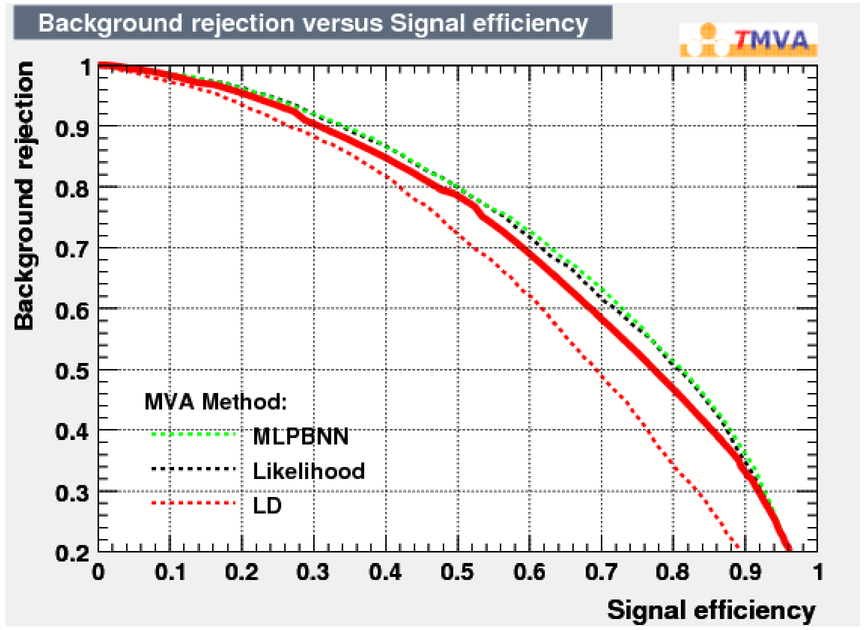
- Defined only for binary classification
- Contains important information:
  all possible combinations of signal and background efficiencies you may achieve by setting threshold
- Particular values of thresholds (and initial pdfs) don't matter, ROC curve doesn't contain this information
- ROC curve =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

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

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

Terminology and Conventions

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



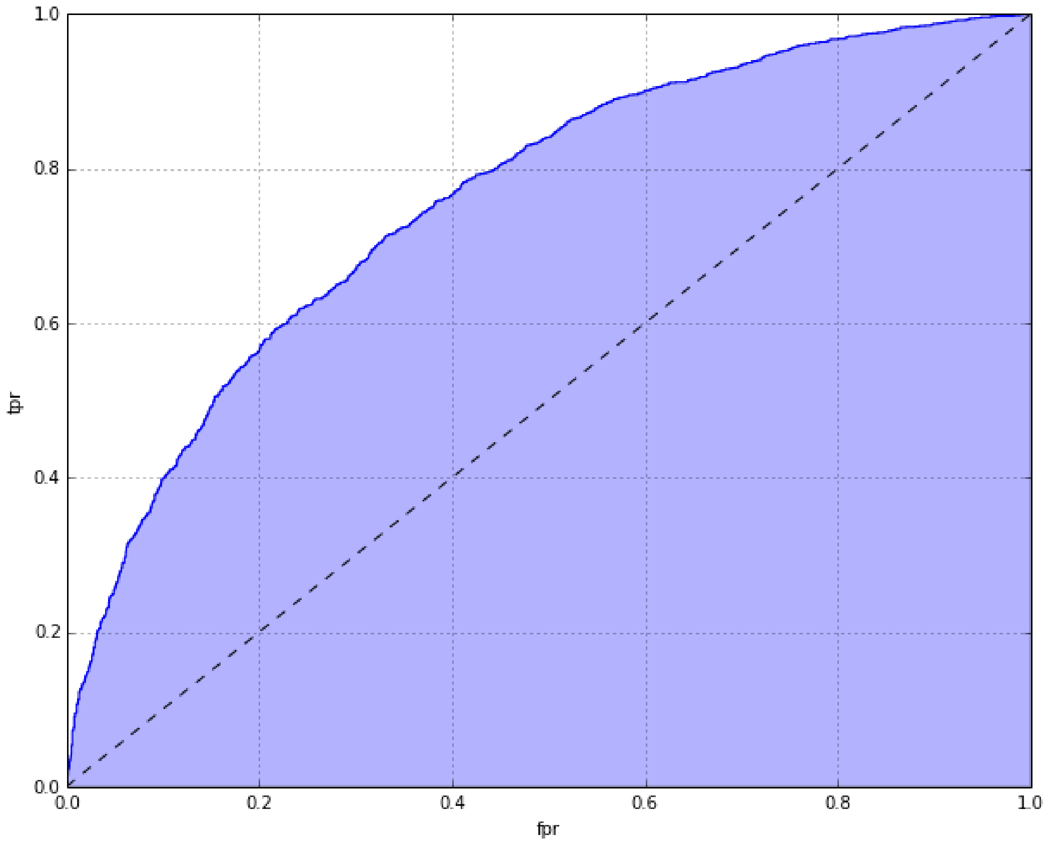
Terminology and Conventions

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

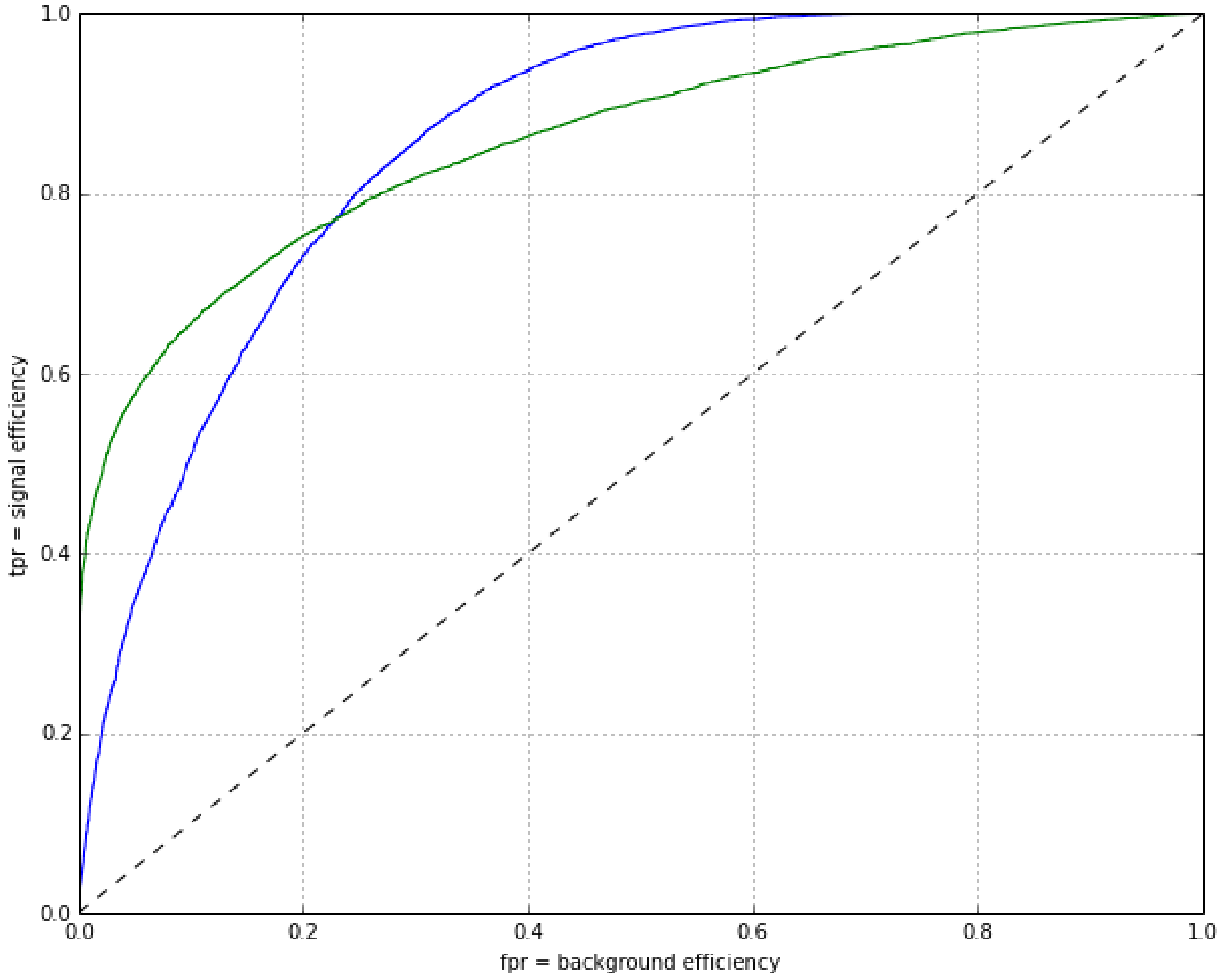
ROC AUC (area under the ROC curve)

$ROC\ AUC = P(r_b < r_s)$

where r_b, r_s are predictions of random background and signal events.

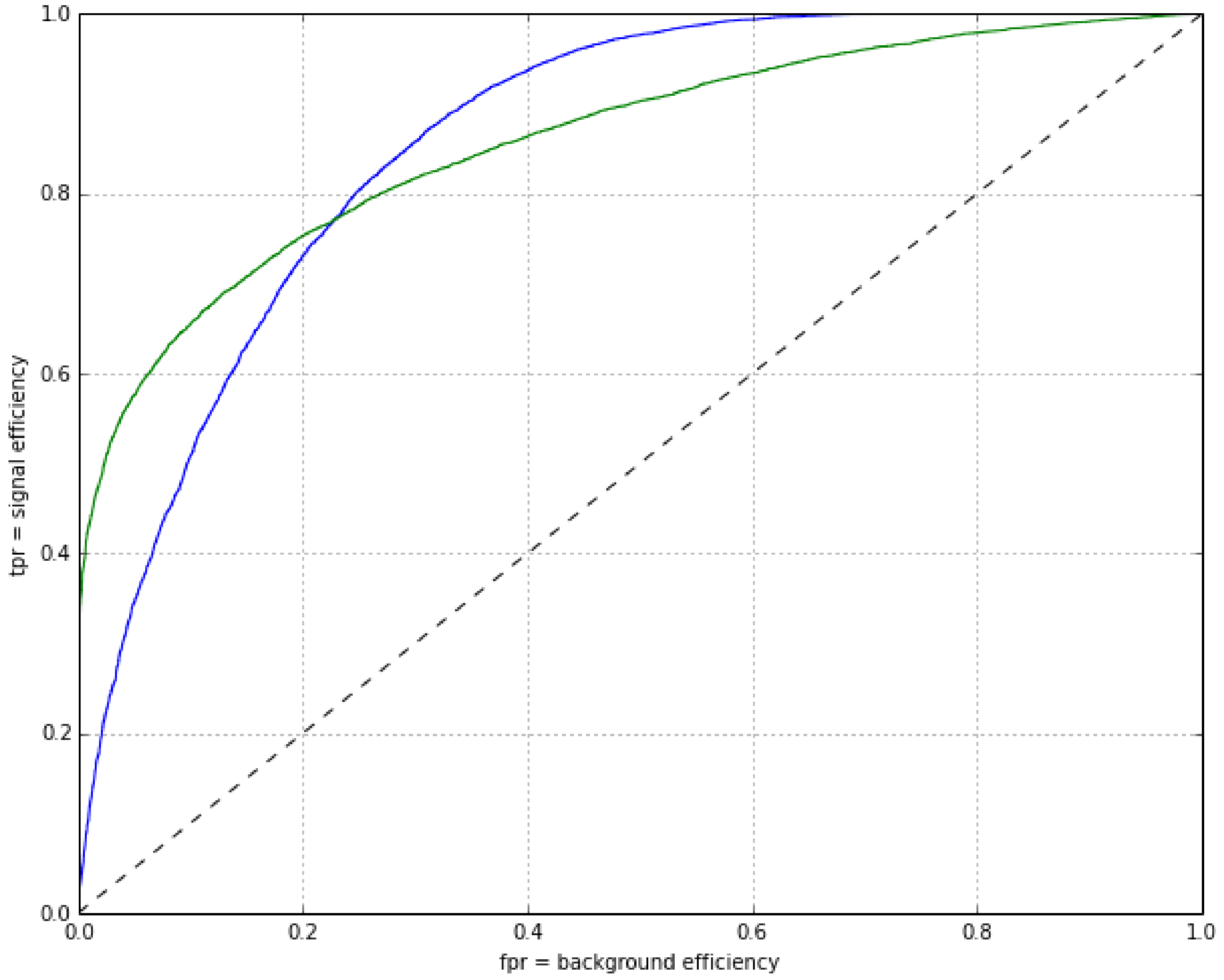


Classifier have the same ROC AUC, but which is better for triggers at the LHC? (we need to pass very few background)



Classifier have the same ROC AUC, but which is better for triggers at the LHC? (we need to pass very few background)

Applications frequently demand different metric.



n-minutes break

Recapitulation

1. Statistical ML: applications and problems
2. ML in HEP
3. k nearest neighbours classifier and regressor.
4. ROC curve, ROC AUC

Statistical Machine Learning

Machine learning we use in practice is based on statistics

- Main assumption: the data is generated from probabilistic distribution:

$$p(x, y)$$

- Does there really exist the distribution of people / pages / texts?

Statistical Machine Learning

Machine learning we use in practice is based on statistics

- Main assumption: the data is generated from probabilistic distribution:

$$p(x, y)$$

- Does there really exist the distribution of people / pages / texts?
- In HEP these distributions do exist

Optimal classification. Bayes optimal classifier

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

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

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

Optimal classification. Bayes optimal 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 $\frac{p(y=1 \mid x)}{p(y=0 \mid x)}$ (*Bayes optimal*

Optimal Binary Classification

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

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

Optimal Binary Classification

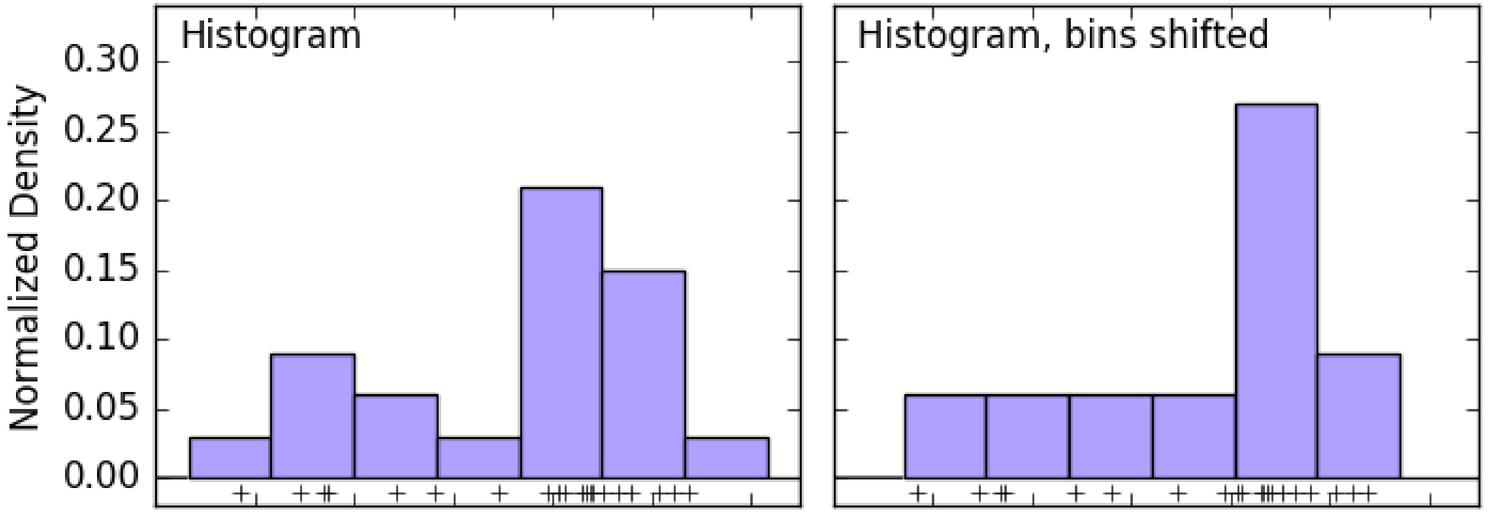
- Bayes optimal classifier has highest possible ROC curve.
- Since the classification quality depends only on order, $p(y = 1 \mid x)$ gives optimal classification quality too!

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

How can we estimate terms from this expression?

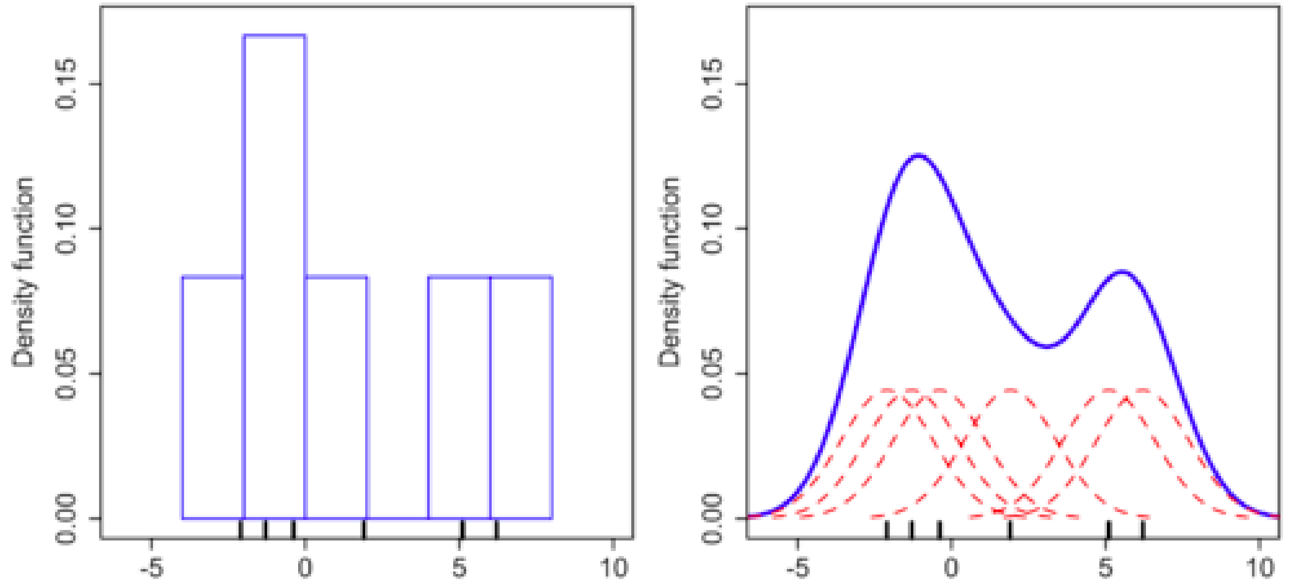
Histograms density estimation

Counting number of samples in each bin and normalizing.



- fast
- choice of binning is crucial
- number of bins grows exponentially $\rightarrow$ curse of dimensionality

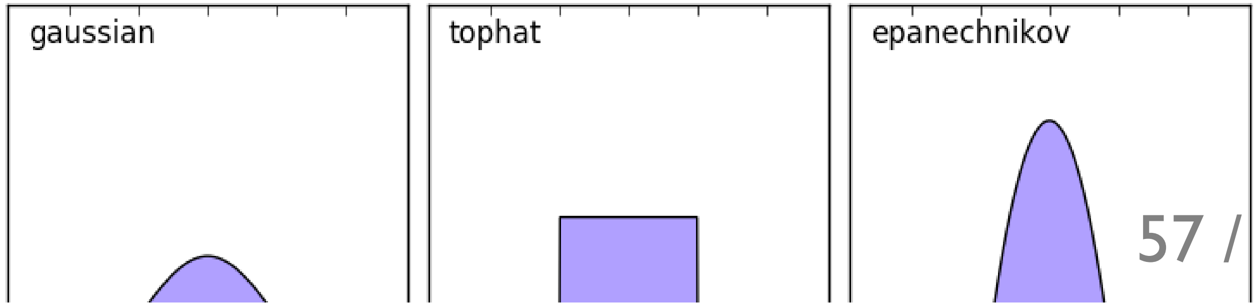
Kernel density estimation



$$f(x) = \frac{1}{nh}\sum_i K\left(\frac{x - x_i}{h}\right)$$

$K(x)$ is kernel, h is bandwidth

Typically, gaussian kernel is used,
but there are many others.



Kernel density estimation

- bandwidth selection
- Silverman's rule of thumb:

$$h = \hat{\sigma}\left(\frac{4}{3n}\right)^{\frac{1}{5}}$$

σ hat is the RMS of data as if it were ‘gaus’-like

Kernel density estimation

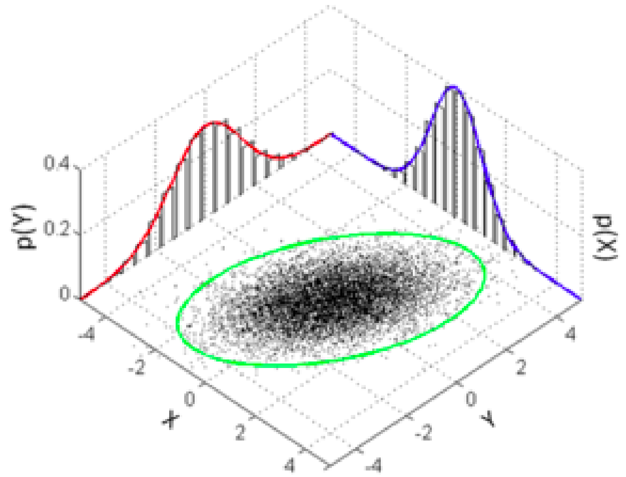
- bandwidth selection
- Silverman's rule of thumb:

$$h = \hat{\sigma}\left(\frac{4}{3n}\right)^{\frac{1}{5}}$$

- may be irrelevant if the data is far from being gaussian

Parametric density estimation

Family of density functions: $f(x;\theta)$.



Problem: estimate parameters of a Gaussian distribution.

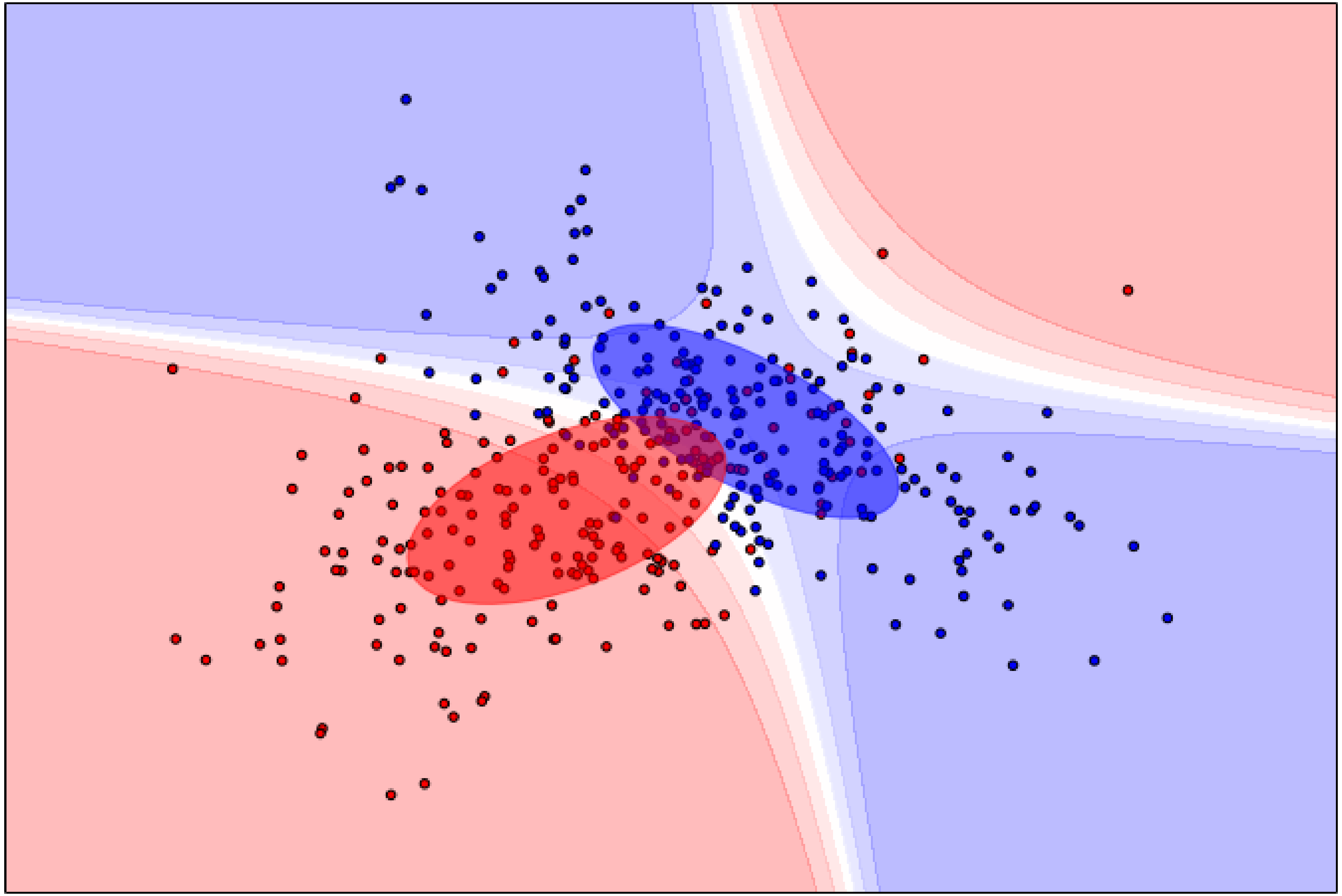
$$f(x;\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

QDA (Quadratic discriminant analysis)

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

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

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



QDA complexity

n samples, d dimensions

- training consists of fitting $p(x \mid y = 0)$ and $p(x \mid y = 1)$ takes $O(nd^2 + d^3)$
  - computing covariance matrix $O(nd^2)$
  - inverting covariance matrix $O(d^3)$
- prediction takes $O(d^2)$ for each sample spent on computing dot product

QDA overview

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

Gaussian mixtures for density estimation

Mixture of distributions:

$$f(x) = \sum_{c-\text{components}} \pi_c f_c(x, \theta_c) \qquad \sum_{c-\text{components}} \pi_c = 1$$

Mixture of Gaussian distributions:

$$f(x) = \sum_{c-\text{components}} \pi_c f(x; \mu_c, \Sigma_c)$$

Parameters to be found: $\pi_1, \ldots, \pi_C, \mu_1, \ldots, \mu_C, \Sigma_1, \ldots, \Sigma_C$

Gaussian mixtures: finding parameters

Criterion is maximizing likelihood (using MLE to find optimal parameters)

$$\sum_i \log f(x_i; \theta) \rightarrow \max_\theta$$

- no analytic solution
- we can use general-purpose optimization methods

Gaussian mixtures: finding parameters

Criterion is maximizing likelihood (using MLE to find optimal parameters)

$$\sum_i \log f(x_i; \theta) \to \max_\theta$$

- no analytic solution
- we can use general-purpose optimization methods

In mixtures parameters are split in two groups:

- $\theta_1, \ldots, \theta_C$ — parameters of components
- $\pi_1, \ldots, \pi_C$ — contributions of components

Expectation-Maximization algorithm [Dempster et al., 1977]

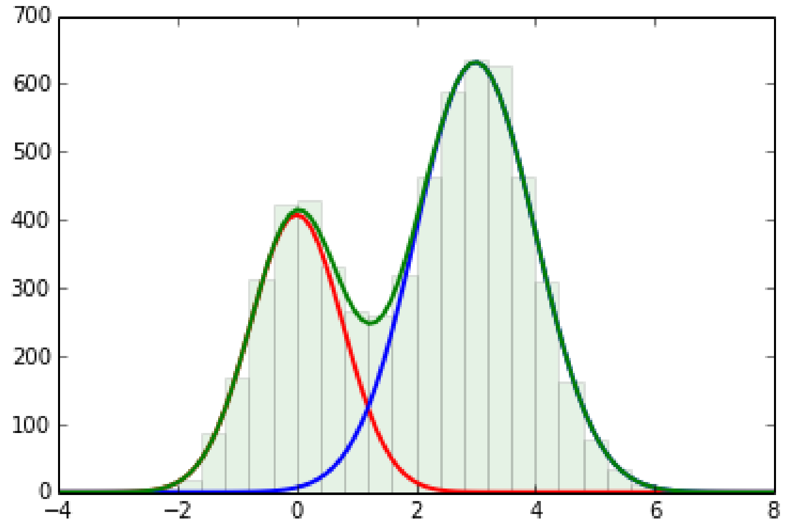
Idea: introduce set of hidden variables $\pi_c(x)$

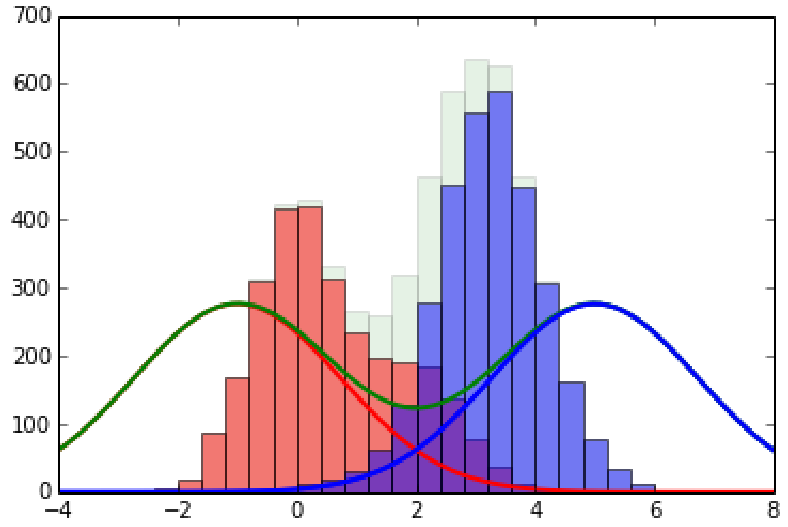
- Expectation: $\pi_c(x) = p(x \in c) = \dfrac{\pi_c f_c(x; \theta_c)}{\sum_{\tilde{c}} \pi_{\tilde{c}} f_{\tilde{c}}(x; \theta_{\tilde{c}})}$
- Maximization:

$$\pi_c = \sum_i p(x_i \in c)$$

$$\theta_c = \arg\max_\theta \sum_i p(x \in c) \log f_c(x, \theta_c)$$

EM algorithm





EM algorithm

Classification model based on mixtures density estimation is called *MDA* (mixture discriminant analysis)

Generative approach

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

QDA, MDA are generative classifiers.

Classification model based on mixtures density estimation is called *MDA* (mixture discriminant analysis)

Generative approach

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

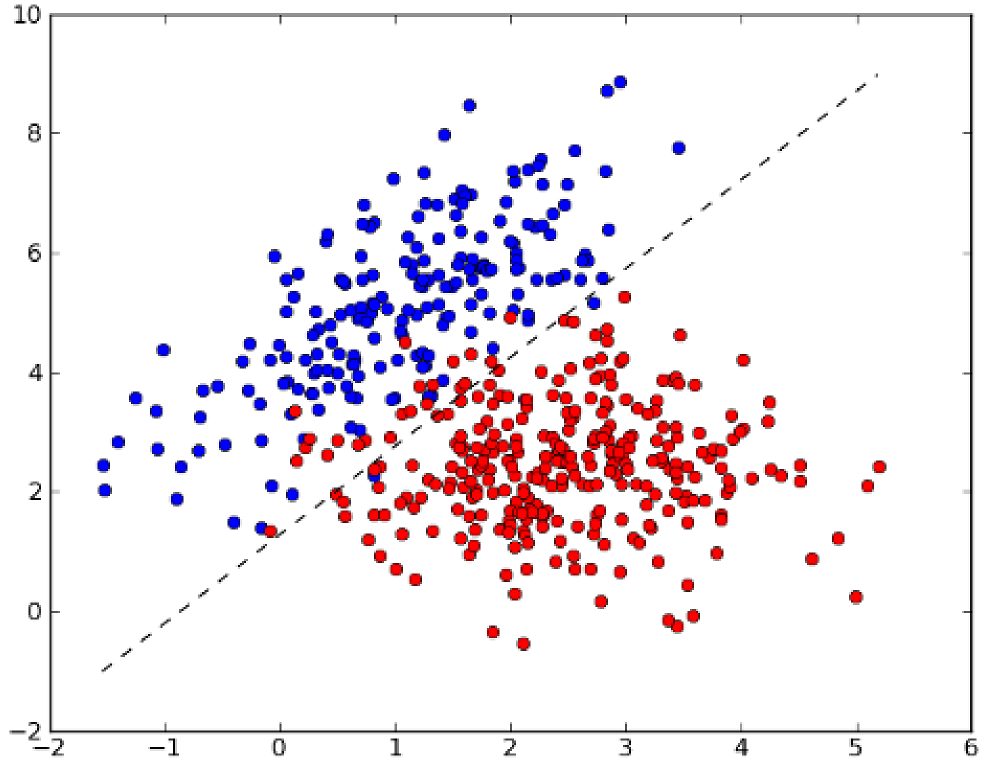
QDA, MDA are generative classifiers.

Problems of generative approach

- Real life distributions hardly can be reconstructed
- Especially in high-dimensional spaces

Classification: truck vs car

Linear decision rule



Decision function is linear:

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

$$\begin{cases} d(x) > 0 \rightarrow & \hat{y} = +1 \\ d(x) < 0 \rightarrow & \hat{y} = -1 \end{cases}$$

This is a 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:

$$\mathrm{L} = \sum_{i \in \text{events}} \mathbf{1}_{y_i \neq \hat{y}_i} \qquad \hat{y}_i = \operatorname{sgn}(d(x_i))$$

  Notion: $\mathbf{1}_{true} = 1,\ \mathbf{1}_{false} = 0.$

- Discontinuous optimization (arrrrgh!)

Finding Optimal Parameters - 2

- Discontinuous optimization
    - solution: 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

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

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 regression

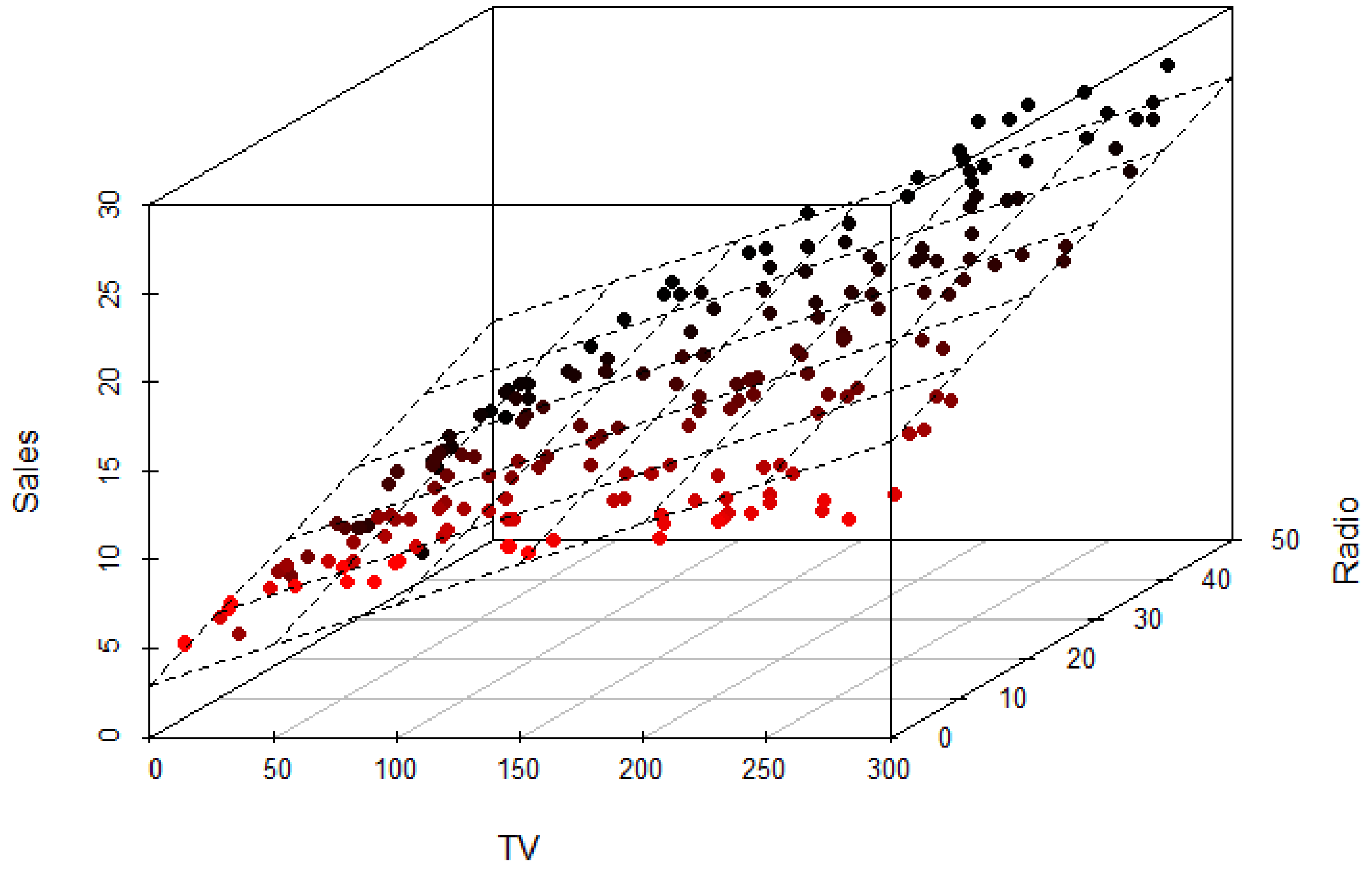
Define probabilities obtained with logistic function

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

and optimize log-likelihood:

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

Linear model for regression



How to use linear function for regression?

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

Simplification of notion: $x_0 = 1, x = (1, x_1, \ldots, x_d)$

.

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

Linear regression (ordinary least squares)

We can use linear function for regression:

$$d(x_i) = y_i \qquad d(x) =< w, x >$$

This is a linear system with $d + 1$ variables and n equations.

Minimize *OLS* aka *MSE* (mean squared error):

$$\mathrm{L} = \sum_i (d(x_i) - y_i)^2 \rightarrow \ \min$$

Linear regression

- can use some other error
  - but no explicit solution in other cases
- demonstrates properties of linear models
  - reliable estimates when $n >> d$
  - able to completely fit to the data if $n = d$
  - undefined when $d < n$

Data Scientist Pipeline

- Experiments in appropriate high-level language or environment
- 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