

Machine learning Lecture 2



- Simple non-linear methods like naive Bayes classifier, k-nearest neighbours, Parzen kernel methods.
- Boosted Decision Trees (BDT)

Overtraining





Correct



Overtraining

- **Overtraining** algorithm "learns" the particular events, not the rules.
 - This effect appears for all ML algorithms.
 - Remedy checking with another, independent dataset.



9.03.2017

M. Wolter, Uczenie maszynowe



Naive Bayes classifier



Frequently called "projected likelihood" (TMVA).

Based on the assumption, that variables are independent (so "naive"):

$$P(y \mid x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n \mid y)}{P(x_1, ..., x_n)}$$
 "Naive" assumption: $P(x_i \mid y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i \mid y),$

$$P(y \mid x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^n P(x_i \mid y)}{P(x_1, ..., x_n)}$$

- Output probability is a product of probabilities for all variables.
- Fast and stable, not optimal, but in many cases sufficient.

Tau identification in ATLAS experiment

 Many identification variables, none of them gives sufficient identification: => use information from all variables.



M. Wolter, Uczenie maszynowe



Tau identification - results



- Cuts well known.
- Likelihood (Naive Bayes) just described.
- Boosted Decision Trees (BDT) we will describe soon.

Kernel density estimators

Approximation of the unknown probability density as a sum of kernel functions centered in the points x_n of the training sample (Parzen, years 1960-ties).

$$D(x) = \frac{P(S)}{P(S) + P(B)}$$

• Typical kernel functions: Gauss, $1/x^n$ itp.

 Simple idea, but using this method requires a lot of memory and CPU.





Approximated probability density (blue) comapred to the true probability density (green) as a function of the width of the Gaussian kernel function. We can see, that the width is a smoothing parameter.



QUAERO package from the D0 experiment



FIG. 1. The background density (a), signal density (b), and selected region (shaded) (c) determined by QUAERO for the standard model processes discussed in the text. From top to bottom the signals are: $WW \rightarrow e\mu \not\!\!\!E_T, ZZ \rightarrow ee\,2j,$ $t\bar{t} \rightarrow e\not\!\!\!E_T 4j$, and $t\bar{t} \rightarrow e\mu \not\!\!\!E_T 2j$. The dots in the plots in the rightmost column represent events observed in the data.



PDE_RS – extension of the Parzen methods

*published by T. Carli, B. Koblitz, NIM A 501 (2003) 576-588

Counts signal (n_s) and background (n_b) events in N-dimensional cube around the classified event – only few events from the training sample are needed. Size of the cube – a free parameter..

Discriminator D(x) :



• Simple analysis.

- Events stored in the binary tree the neighbor events are found quickly.
- It's a special case of Parzen method the kernel function:

Tau lepton identification in ATLAS $\pi^{+} \pi^{0} \pi^{+}$ experiment



3-prong events

None of the variables separates tau's from background sufficiently well – we need to use all of them.







KNN - k nearest neighbors

- Proposed already in 1951
- An event is qualified to the class, to which belongs the majority of it's k nearest neighbors,
- or we calculate the probability of belonging to the class "signal" as:





Boosting, bagging, BDT... Ensemble learning



What does BDT mean???

BDT – Boosted Decision Tree:

- Decision Tree an algorithm know for a long time, used in most of the expert systems. For example, the first aid manual is frequently written in the form of a decision tree: if (condition) do something, else do something different, then check another condition...
- Boosted a method of joining many weak classifiers in an ensemble to get one strong classifiers. It is not limited to decision trees, however with them it is most commonly used.

Decision trees



Decision tree – a series of cuts, each "leaf" (A,B,C,D,E) has a label, for example "signal" and "background".





- Easy in visualization and interpretation
- Resistant for *outliers*.
- Weak variables are ignored.
- Fast training and classification.
- Unfortunately: sensitive for fluctuations, unstable.

Building the tree





- We divide the traing sample by the best separating cut on the best variable.
- We repeat the procedure until the stopping conditions are fulfilled, for example: number of leafs, number of events in a leaf etc.
- The ratio S/B in a leaf defines the classification (binary signal or background, or a real number giving the probability, that a given event is a signal).



 Gini inpurity: (Corrado Gini 1912, invented the Gini index used to measure the inequality of incomes)

p (1-p) : p= P(signal), *purity*

- Cross-entropy:
 -(p ln p + (1-p)ln(1-p))
- Missidentification:
 - 1-max(p,1-p)

AdaBoost – ensemble of classifiers



16

Problem: could we improve a weak classifier?

Answer: yes, by applying it many times.

 An algorithm used most frequently: AdaBoost (Freund & Schapire 1996 – Gödel prize)

- Build a decision tree.
- Increase the weights of wrongly classified events.
- Repeat many times (typically 100-1000)
- Classify the events by "voting" by all the trees.







AdaBoost for 2-dimensional data – results after 1st, 2nd, 3rd, 5th, 10th and 100th iteration. The solid green line shows the results of the combined classifier, the dashed line the borderes between classes obtained after each step. For the last two plots the dotted line marks the class borders obtained from the bagging algorithm (we will talk about bagging soon).

AdaBoost



 Five years after publication of AdaBoost Friedman proved, that the algorithm in fact minimizes the exponential loss function:

$$E = \sum_{n=1}^{\infty} \exp\left(-t_n f_m(x_n)\right)$$

n=1 where *f(x)* is the answer of the algorithm

t = 1 signal, t = -1 background

 $t_n \cdot f_m(x_n) > 0$ – correctly classified

 $t_n \cdot f_m(x_n) < 0$ – incorrectly classified

- Exponential function goes up quickly => huge punishment for wrongly classified events, so the algorithm is sensitive for single, outstanding points. Classification becomes worse, when data are hard going of the sensitive for single of the sensitive for sensitive for single of the sensitive for sensitive for single of the sensitive for sensitive for single of the sensitive for sensitive fo
 - Another loss function?
 - Friedman in year 2000 proposed few other loss functions, but AdaBoost is still the most popular (we will talk about).





AdaBoost in action

AdaBoost in Action

Kai O. Arras Social Robotics Lab, University of Freiburg

Nov 2009 000 Social Robotics Laboratory



AdaBoost algorithm in detail

1. Give all vectors from the training set a weight $w_i = 1/N$.

2. For m = 1,...,M:

a) Train classifiers $y_m(x)$ on the training sample minimizing:

$$J_m = \sum_{n=1}^N w_n^m I\left(y_m(x_n) \neq t_n\right)$$

b) Calculate the quantities:

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^m I\left(y_m(x_n) \neq t_n\right)}{\sum_{n=1}^N w_n^m}$$
$$\alpha_m = \frac{1}{2} \ln \frac{1 - \epsilon_m}{\epsilon_m}$$

c) Change the weights of the vectors in the training sample:

$$w_n^{m+1} = \begin{cases} \frac{w_n^m}{Z_m} e^{\alpha_m} & \text{dla} \quad y_m(x_n) \neq t_n \\ \\ \frac{w_n^m}{Z_m} e^{-\alpha_m} & \text{dla} \quad y_m(x_n) = t_n \end{cases}$$

3. The result of voting is given as:

$$Y_M(\mathbf{x}) = sign\left[\sum_{m=1}^M \alpha_m y_m(\mathbf{x})\right]$$

All the classifiers are trained on the same training sample, but with different weights. These weights depend on the results of the preceding training, so it's difficult to train many classifiers in parallel.



AdaBoost

- 1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for n = 1, ..., N.
- 2. For m = 1, ..., M:
 - (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$
(14.15)

where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

(b) Evaluate the quantities

$$\epsilon_{m} = \frac{\sum_{n=1}^{N} w_{n}^{(m)} I(y_{m}(\mathbf{x}_{n}) \neq t_{n})}{\sum_{n=1}^{N} w_{n}^{(m)}}$$
(14.16)

and then use these to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}.$$
(14.17)

(c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$
(14.18)

Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x})\right). \tag{14.19}$$

M. Wolter, Uczenie maszynowe





Classifier boosting

- Boosting, bagging in a "magical" way we get a strong classifier out of weak ones
- Typically boosting used on decision trees – Boosted Decision Trees BDT.
- Good results without time consuming tuning of parameters:

"the best out-of-box classification algorithm".

- Relatively resistant on overtraining.
- Quite frequently used nowadays. And with good results!



Naive Bayes classifier Boosted Naive Bayes classifier

Application of a boosting algorithm (5 iterations) for Naive Bayes classifier.

Boosting – different loss functions

• Another loss functions:

-	Nazwa funkcji	Równanie	Nazwa algorytmu					
-	wykładnicza	$\exp\left(-t_n f_m(x_n)\right)$	A da Boost					
-	prawdopodobieństwa	$\log\left(1 + \exp(-2t_n f_m(x_n))\right)$	LogitBoost	1				
-	błąd kwadratowy	$(t_n - f_m(x_n))^2$	SquareBoost	1				
-	błąd absolutny	$ t_n - f_m(x_n) $		1				
-	zero-jedynkowa	$\begin{cases} 0 & sign(f_m(x_n)) \neq t_n \\ 1 & sign(f_m(x_n)) = t_n \end{cases}$			 			
Pro: Not so sensitive for outliers. The properly chosen "sensitivity" is important, when there are regions with mixed signal and background. Contra: There is no fast and simple minimization algorithm like AdaBoost => minimization using general algorithms, like for function fit.			$ \begin{array}{c} 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \\ 2 \\ 1 \\ 0 \\ 2 \\ -1.5 \\ \end{array} $	-1 -0.5	logit square absolut misclas	e sifica	tion	



Boosting – different loss functions



- Discrete AdaBoost the basic AdaBoost described on previous slides
- Real AdaBoost, LogitBoost, Gentle AdaBoost- modifications (different loss functions and minimization algorithms) proposed by Friedman.
- For bigger trees the results are identical.



Stumps - 2 Classes

Eight Node Trees - 2 Classes

Number of Terms

Number of Terms





Ovetraing of the Boosted Decision Tree (BDT)

Algorithm uses stumps – decision trees with two branches.

At the end the space is divided into very small areas overtraining.

9.03.2017

M. Wolter, Uczenie maszynowe

Bagging (Bootstrap AGGregatING)

1.5

0.5

-0.5

-1.5

Algorithm proposed by Leo Breiman in 1994:

- Take N events from N-element training set, but with repetitions.
- Train the classifier on this set.
- Repeat many times
- Classify new events by voting of all the classifiers.



For this data the red line (mean of 100 classifiers) is smoother, more stable and more resistant for overtraining the any of the single classifiers.

Analogy: mean of many poorly fitting functions gives a well fitting function.

funkcja sinus

dopasowane wielomiany

srednia z wielomianow

M. Wolter, Uczenie maszynowe

Х



Bagging

- **Random Forest** further development of Bagging:
 - K times pick N elements from the N element sample.
 - For each sample train a decision tree.
 - Building the tree in every decision point use only m randomly chosen variables

- Typically boosting gives better results, but bagging has a tendency to be more stable at high noise or when there are many "outliers".
- Bagging is a parallel algorithm, boosting a sequential one.

Bauer and Kohavi, "An empirical comparison of voting classification algorithms", Machine Learning 36 (1999)

Bagging vs. boosting



300

Stumps

Test error for Bagging, Discrete AdaBoost and Real AdaBoost on a simulated two-class nested spheres problem. There are 2000 training data points in ten

400

Number of Terms

10 Node Trees



Discrete AdaBoost [Freund and Schapire (1996b)]

- 1. Start with weights $w_i = 1/N, i = 1, \dots, N$.
- 2. Repeat for m = 1, 2, ..., M:
 - (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
 - (b) Compute $\operatorname{err}_m = E_w[1_{(y \neq f_m(x))}], c_m = \log((1 \operatorname{err}_m)/\operatorname{err}_m).$
 - (c) Set $w_i \leftarrow w_i \exp[c_m \mathbb{1}_{(y_i \neq f_m(x_i))}]$, i = 1, 2, ..., N, and renormalize so that $\sum_i w_i = 1$.
- 3. Output the classifier sign[$\sum_{m=1}^{M} c_m f_m(x)$].

Real AdaBoost

- 1. Start with weights $w_i = 1/N$, i = 1, 2, ..., N.
- 2. Repeat for m = 1, 2, ..., M:
 - (a) Fit the classifier to obtain a class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, using weights w_i on the training data.
 - (b) Set $f_m(x) \leftarrow \frac{1}{2} \log p_m(x) / (1 p_m(x)) \in R$.
 - (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, i = 1, 2, ..., N, and renormalize so that $\sum_i w_i = 1$.
- 3. Output the classifier sign[$\sum_{m=1}^{M} f_m(x)$].

Exercise using TMVA



Boosting and Bagging – separations of two Gaussians in 4 dimensions (linear Fisher Classifier is always the best).



Trees with a depth 10

Again tau lepton identification in ATLAS



BDT better than Naive Bayes and cuts

Searches of decays $B \rightarrow \gamma l \nu$ in BaBar experiment





11-D data: 10 continuous and 1 discrete variable.

Some variables are strongly correlated: $\rho(Fisher,acthrust)=0.8$ and $\rho(costheblg,nuEP)=-0.87$.

Monte Carlo:

- training = 500K
- validation = 250K
- test = 250K



Signal significance obtained by different methods:

Method		$B \to \gamma e \nu$				$B \to \gamma \mu \nu$				
	$\mathcal{S}_{ ext{train}}$	$\mathcal{S}_{\mathrm{valid}}$	$\mathcal{S}_{\text{test}}$	W_1	W_0	$\mathcal{S}_{ ext{train}}$	$\mathcal{S}_{\mathrm{valid}}$	$\mathcal{S}_{ ext{test}}$	W_1	W_0
Original method	2.66	-	2.42	37.5	202.2	1.75	-	1.62	25.8	227.4
Decision tree	3.28	2.72	2.16	20.3	68.1	1.74	1.63	1.54	29.0	325.9
Bump hunter with one bump	2.72	2.54	2.31	47.5	376.6	1.76	1.54	1.54	31.7	393.8
AdaBoost with binary splits	2.54	2.65	2.27	84.2	1288.5	1.68	1.74	1.47	49.7	1087.7
AdaBoost with decision trees	13.63	2.99	2.62	58.0	432.8	11.87	1.97	1.75	41.6	523.0
Combiner of background subclassifiers	3.03	2.88	2.49	83.2	1037.2	1.84	1.90	1.66	55.2	1057.1
Bagging with decision trees	9.20	3.25	2.99	69.1	465.8	8.09	2.07	1.98	49.4	571.1

Training of 50 "boosted decision trees" or 100 "bagged decision trees" took few hours. W_1 i W_0 are the expected numbers of signal and background events at luminosity 210 fb⁻¹.

Cleanest signal "bagged decision trees".

"Bagged decision trees" improve the signal purity by 14% comparing to "boosted decision trees":

• 8% due to "bagging" instead of "boosting".

• 9% due to optimalization S/(S+B) instead of Gini.

Experiment Mini-BooNE – neutrino beam





Przykładowe przypadki

Pattern of hit tubes (with charge and time information) allows reconstruction of track location and direction and separation of different event types.

e.g. candidate events:



muon from ν_{μ} interaction



Michel electron from stopped μ decay after ν_{μ} interaction

size = charge, color = time



 $\pi^0 \rightarrow$ two photons from ν_{μ} interaction





- Boosting, bagging in a magic way we can build a strong classifier out of weak ones.
- Commonly used for decision trees, because they are simple and fast.
- Gives good results:
 - "the best out-of-box classification algorithm".
- Very popular in high energy physics
- Or maybe just fancy?...

Should we use BDT only:

It follows, therefore, that if a method is already close to the Bayes limit, then *no* other method, however sophisticated, can be expected to yield dramatic improvements.

Harrison B. Prosper

Exercise: apply BDT to the "chessboard" data

- Create ntuple TMVAcreateNtuple.py https://indico.ifj.edu.pl/event/196/
- Using examples from the previous lecture apply: Likelihood (Naive Bayes), kNN, BDT with boosting and bagging
- Note how many trees and how big are needed for Boosted and Bagged trees
- Plot the ROC curve and the output map (TMVAreader.py)





Chessboard analysis



9.03.2017

M. Wolter, Uczenie maszynowe