.

Machine Learning

Boosting

Prof. Dr. Martin Riedmiller
AG Maschinelles Lernen und Natürlichsprachliche Systeme
Institut für Informatik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

riedmiller@informatik.uni-freiburg.de

Acknowledgment Slides are based on tutorials from Robert Schapire and Gunnar Raetsch

Overview of Today's Lecture: Boosting

- Motivation
- AdaBoost
- examples

Motivation

- often, simple rules already yield a reasonable classification performance
- e.g. email spam filter: 'buy now' would already be a good indicator for spam, doing better than random
- idea: use several of these simple rules ('decisions stumps') and combine them to do classification ($\Rightarrow$ committee appraoch)

Boosting

'Boosting' is a general method to combine simple, rough rules into a highly accurate decision model.

- assumes 'base learners' (or 'weak learners') that do at least slightly better than random, e.g. $\geq 53\%$ for a two-class classification. In other words the error is strictly smaller than 0.5.
- given sufficient data, a boosting algorithm can provably construct a single classifier with very high accuracy (say, e.g. 99%)

Algorithm: AdaBoost (Freund and Schapire, 1995)

General Framework for Boosting

- training set $(x_1, y_1), \ldots, (x_p, y_p)$, where $y_i \in \{-1, 1\}$ and $x_i \in \mathcal{X}$
- For $t = 1 \ldots T$
  - construct a distribution $D_t = (d_1^t, \ldots, d_p^t)$ on the training set, where d_i^t is the probability of occurence of training pattern i (also called 'weight' of i)
  - train a base learner $h_t : \mathcal{X} \mapsto \{-1, +1\}$ according to the training set distribution D_t with a small error ϵ_t, $\epsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i]$
- output final classifier H_{final} as a combination of the T hypothesis $h_1, \ldots, h_T$

AdaBoost - Details (Freund and Schapire, 1995)

- the final hypothesis is a weighted sum of the individual hypothesis:

$$H_{final}(x) = sign(\sum_t \alpha_t \, h_t(x))$$

- hypothesis are weighted depending on the error they produce:

$$\alpha_t := \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

  e.g: $\epsilon_t = 0.1 \rightarrow \alpha_t = 2.197$; $\epsilon_t = 0.4 \rightarrow \alpha_t = 0.41$;
  $\epsilon_t = 0.5 \rightarrow \alpha_t = 0$;

- the above choice of α can be shown to minimize an upper bound of the final hypothesis error (see e.g. (Schapire, 2003) for details)

AdaBoost - Details (cont'd) (Freund and Schapire, 1995)

- training pattern probabilities are recomputed, such that wrongly classified patterns gets a higher probability in the next round:

$$d_i^{t+1} := \frac{d_i^t}{Z_t} * \begin{cases} \exp(\alpha_t) & , \quad \text{if } y_i \neq h_t(x_i) \\ \exp(-\alpha_t) & , \quad \text{else} \end{cases}$$

$$= \frac{d_i^t}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

- Z_t is a normalisation factor, such that $\sum_i d_i^t = 1$ (probability distribution), i.e.

$$Z_t = \sum_i d_i^t \exp(-\alpha_t \, y_i \, h_t(x_i))$$

AdaBoost (Freund and Schapire, 1995)

- Initialize $D^1 = (1/p, \dots, 1/p)$, p is the number of training patterns
- For $t = 1 \dots T$:
  - train base learner h_t according to distribution D^t
  - determine expected error:

    $$\epsilon_t := \sum_{i=1}^{p} d_i^t \, \mathbf{I}(h(x_i) \neq y_i),$$

    where $I(A) = 1$, if A is true, 0, else.
  - compute hypothesis weight and new pattern distribution:

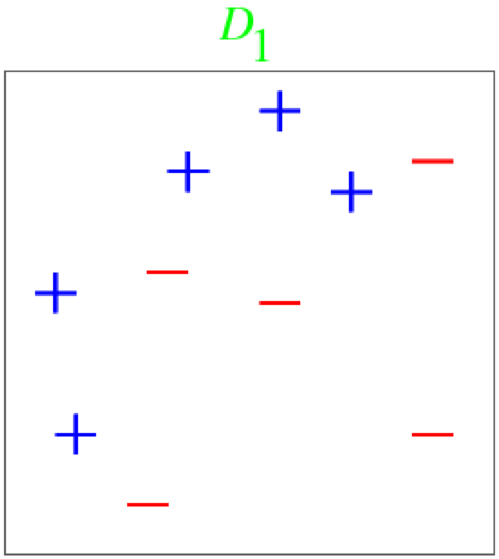
    $$\alpha_t := \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

    $$d_i^{t+1} := \frac{d_i^t}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i)),$$ (Z_t normalisation factor)
- Final hypothesis: $H_{final}(x) = sign(\sum_t \alpha_t \, h_t(x))$

Example (1)

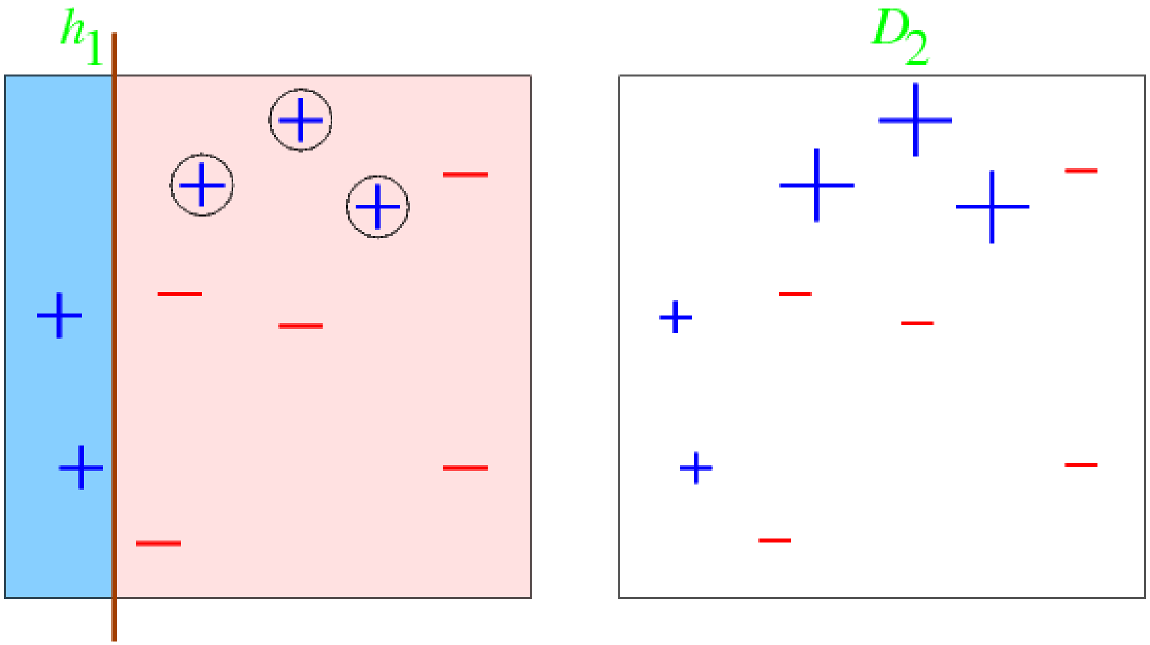
Example taken from (R. Schapire)

Hypothesis: simple splits parallel to one axis



Example (2)

Example taken from (R. Schapire)



$error\, \epsilon_1 =?$

$\epsilon_1 = 0.3, \alpha_1 = 0.42$, wrong pattern: $d_i^2 = 0.166$, correct: $d_i^2 = 0.07$

computation see next slide

$$\epsilon_1 = \sum_{i=1}^{p} d_i^1 \, \mathbf{I}(h(x_i) \neq y_i) = \frac{1}{10} 3 = 0.3$$

$$\alpha_1 = \frac{1}{2} \ln \frac{1-\epsilon_1}{\epsilon_1} = \frac{1}{2} \ln \frac{1-0.3}{0.3} = 0.42$$

wrong patterns: $d_i^1 \exp(\alpha_1) = 0.1 \exp(0.42) = 0.152$

correct patterns: $d_i^1 \exp(-\alpha_1) = 0.1 \exp(-0.42) = 0.066$

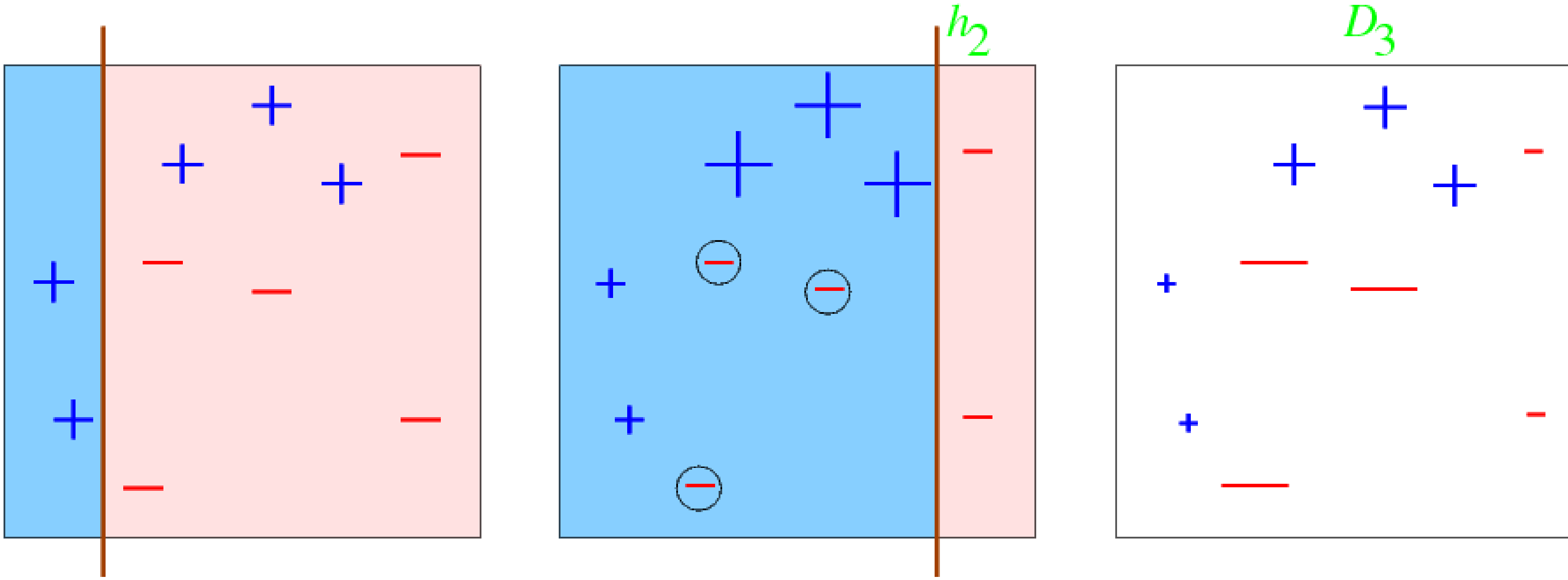
$Z_1 = 7\,0.066 + 3\,0.152 = 0.918$

wrong patterns: $d_i^2 = 0.152/0.918 = 0.166$

correct patterns: $d_i^2 = 0.066/0.918 = 0.07$

Example (3)

Example taken from (R. Schapire)



$error\, \epsilon_2 = ?$

$\epsilon_2 = 0.21, \alpha_2 = 0.65$

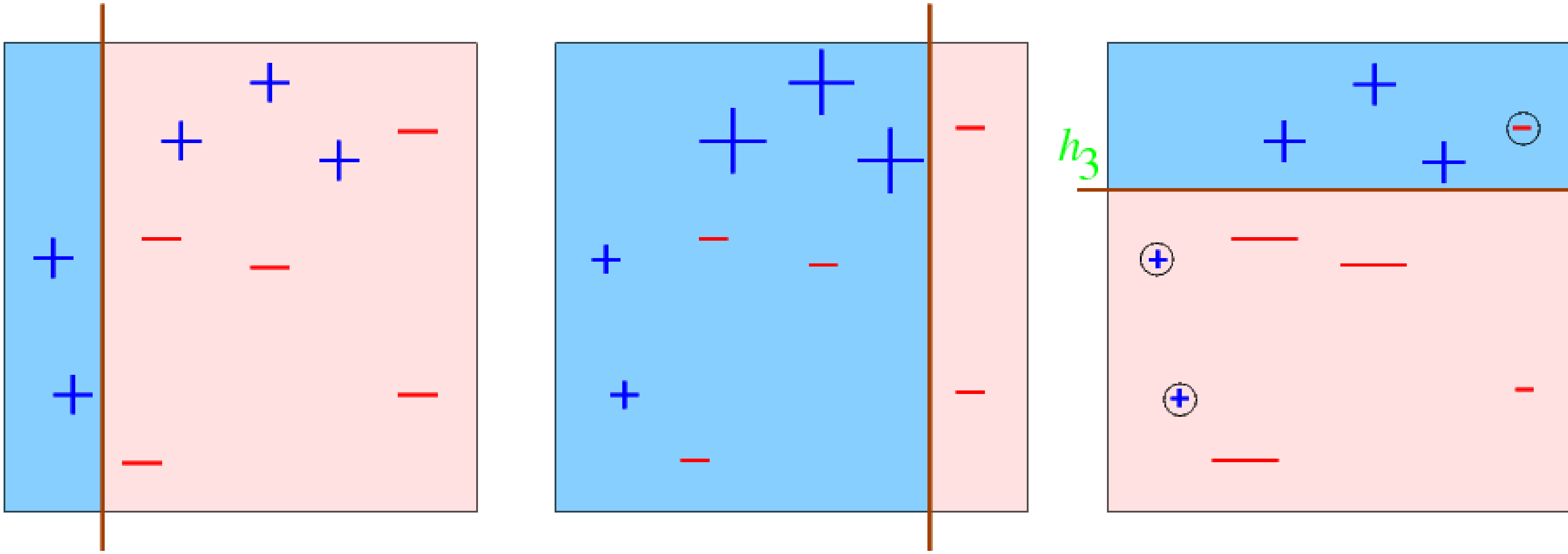
computation see next slide

$$\epsilon_2 = \sum_{i=1}^{p} d_i^2 \, \mathbf{I}(h(x_i) \neq y_i) = 0.07 + 0.07 + 0.07 = 0.21$$

$$\alpha_1 = \frac{1}{2} \, \ln \frac{1 - \epsilon_2}{\epsilon_2} = \frac{1}{2} \, \ln \frac{1 - 0.21}{0.21} = 0.65$$

Example (4)

Example taken from (R. Schapire)

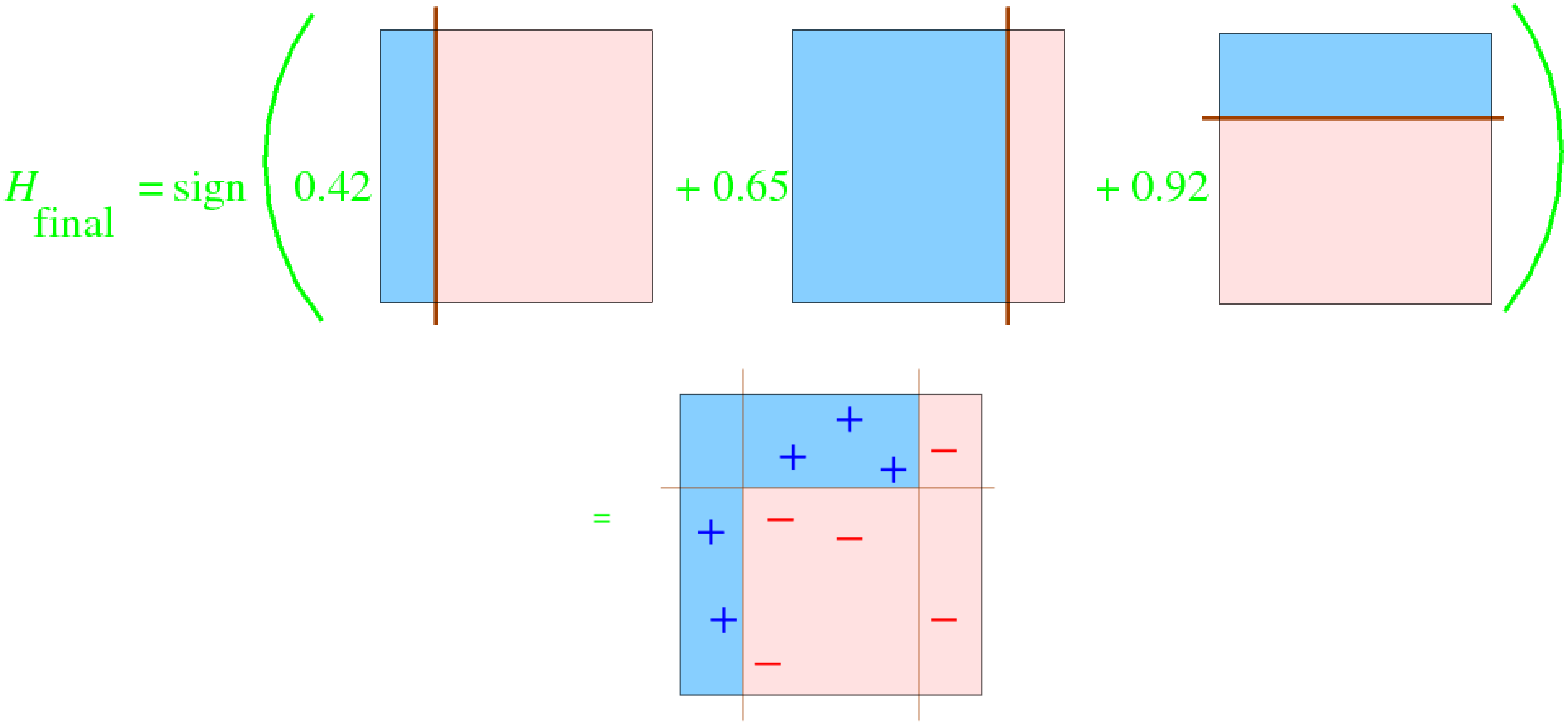


$\epsilon_2 = 0.14, \alpha_2 = 0.92$

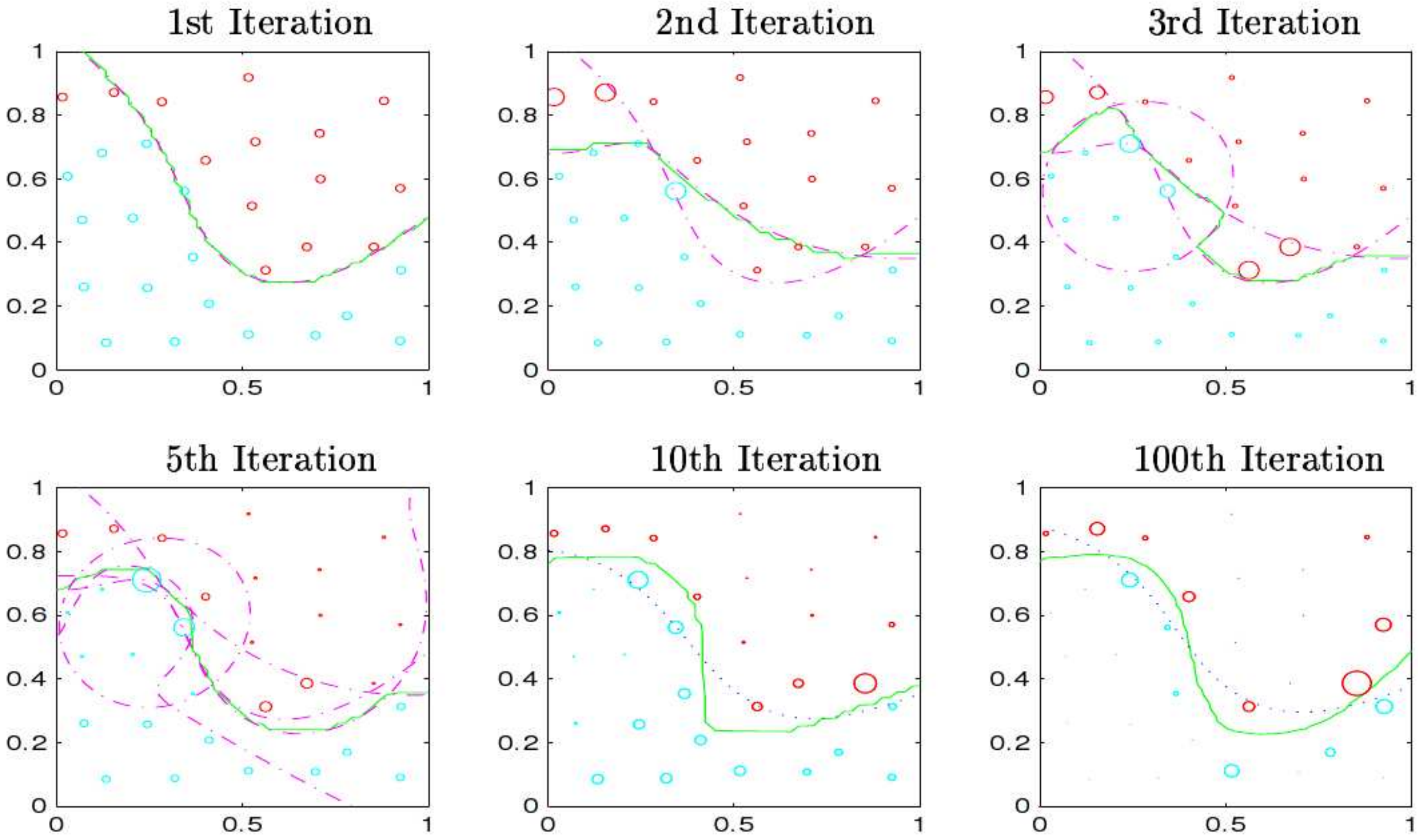
Example (5)

Example taken from (R. Schapire)

Final classifier:



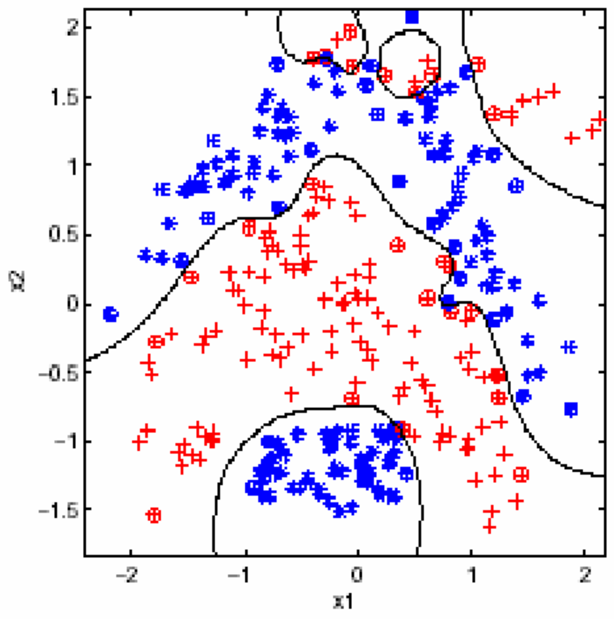
Further examples (1)



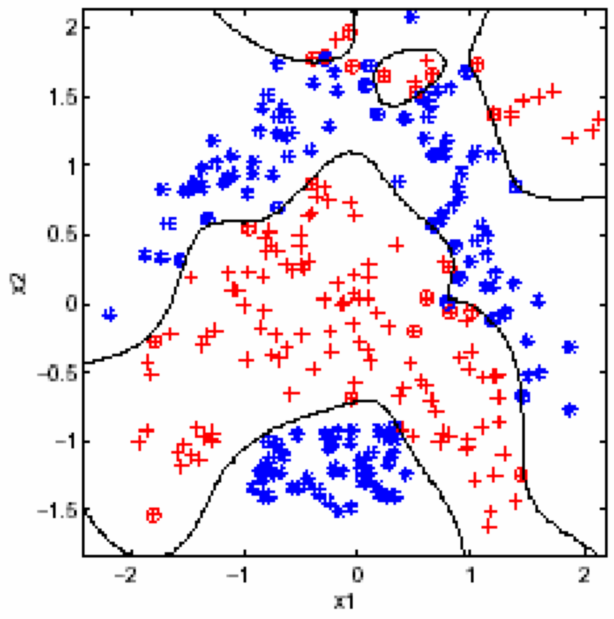
Taken from (Meir, Raetsch, 2003): AdaBoost on a 2D toy data set: color indicates the label and the diameter is proportional to the weight of the example. Dashed lines show decision boundaries of the single classifiers (up to 5th iteration). Solid line shows the decision line of the combined classifier. In the last two plots the decision line of Bagging is plotted for a comparison.

Performance of AdaBoost

In case of low noise in the training data, AdaBoost can reasonbly learn even complex decision boundaries without overfitting



AdaBoost's decision line

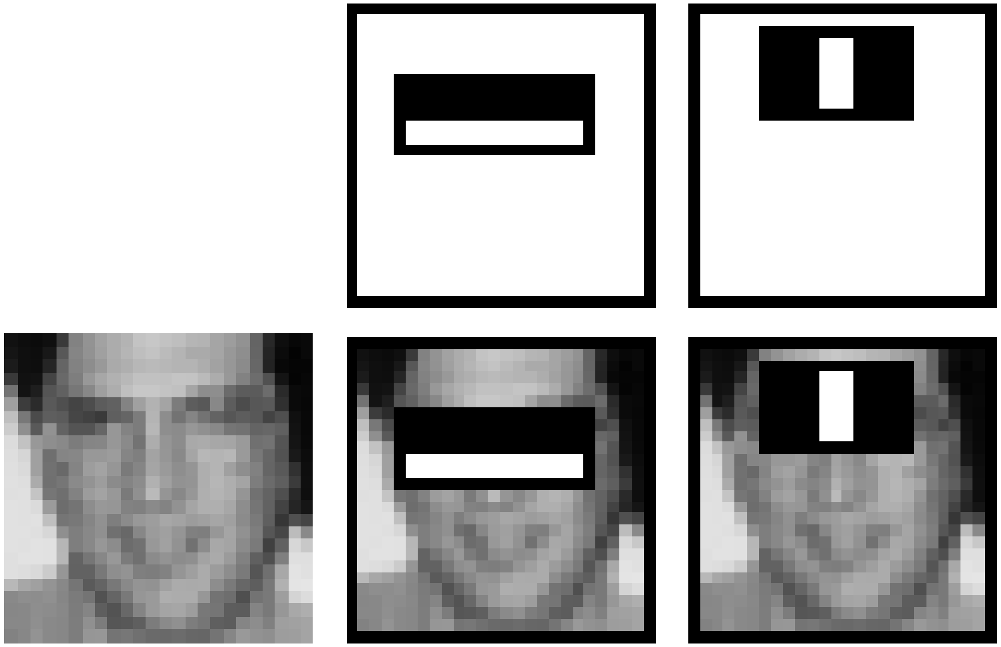


SVM's decision line

Figure taken from G. Raetsch, Tutorial MLSS '03

Practical Application: Face Recognition (Viola and Jones, 2004)

- problem: find faces in photographs and movies
- weak classifiers: detec light/ dark rectangles in images
- many tricks to make it fast and accurate



Further aspects

- Boosting can be applied to any ML classifier: MLPs, Decision Trees, . . .
- related approach: Bagging (Breiman): training sample distribution remains unchanged. For each training of a hypotheses, another set of training samples is drawn with putting back.
- boosting performs usually very well with respect to non overfitting the data, if the noise in the data is low. While this is astonishing at first glance, it can be theortically explained by analyzing the margins of the classifiers.
- if the data has a higher noise level, boosting runs into the problem of overfitting. Regularisation methods exist, that try to avoid this (e.g. by restricting the weight of notorious outliers).
- extension to multi-class problems and regression problems exist.

References

- R. Schapire: A boosting tutorial. www.cs.princeton.edu/s̃chapire
- Meir and G. Raetsch, 2003: An introduction to boosting and leveraging 2003
- G. Raetsch: Tutorial at MLSS 2003
- Viola and Jones, 2004: Fast Face Recognition