

# Machine Learning

## Boosting

Prof. Matthias Hein

Machine Learning Group  
Department of Mathematics and Computer Science  
Saarland University, Saarbrücken, Germany

Lecture 17, 10.01.2014

Classification methods:

- **Boosting,**
- Decision Trees,
- Neural Networks,
- **Nearest Neighbor Methods, Parzen-Window,**

## Boosting

- **Given:** a set of (bad) classifiers.
- Question:** Is there a way to combine them so that they yield a reasonable classifier ?
- Boosting “boosts” existing bad/weak classifiers.
- Developed by Schapire and Freund (1996) - but concept is older.
- Final classification can often be implemented quite efficiently - interesting for real time application e.g. face detection on a video sequence.

**Boosting belongs to the ensemble methods**

## History of Boosting

- Freund and Schapire propose Adaboost in 1996,  
Folklore result: Adaboost does not overfit !
- Friedman, Hastie and Tibshirani (2000) provide interpretation of  
Adaboost in terms of minimization of empirical exponential loss  
⇒ provides a general scheme for boosting which leads to emergence  
of new variants.
- Lugosi and Vayatis (2004) prove for a regularized boosting variant  
that it is Bayes consistent.
- Bartlett and Traskin (2007) prove that Adaboost is Bayes consistent.
- still ongoing discussion about interpretation after more than 10 years  
of boosting (recent controversial JMLR article).

## General scheme for boosting methods

- ① For each training point  $(X_i, Y_i)$  one has a weight  $\gamma_i$ .
- ② A step of boosting method involves the following steps
  - ① one trains a classifier  $f_k$  using a base method (weak learner) with the weighted training data  $(X_i, Y_i, \gamma_i)$ ,
  - ② one re-computes the weights  $\gamma$ , where usually the weights of wrongly classified training points are increasing and the weights of correctly classified points are decreasing.
- ③ One aggregates the classifiers  $f_k$  to the final classifier  
 $F(x) = \text{sign}(\sum_{k=1} \alpha_k f_k)$ , where the coefficients  $\alpha_k$  are either one or depend on the error of the classifier  $f_k$ .

## Caution

- Several boosting methods have been proposed (huge literature),
- Differences are often very subtle,
- Different requirements on the properties of the weak learner.

**Note:** Boosting heavily depends on the weak learner

⇒ two boosting methods with different weak learner cannot be compared !

**Today:** **Adaboost** and **GentleBoost**.

# What is Adaboost ?

## Properties

- Adaboost stands for “Adaptive Boosting”
- Many variants - we discuss Adaboost.M1 proposed by Freund and Schapire in 1996.
- Weak learner has to be **binary-valued** !
- Depending on the weak learner - final classifier allows interpretation (boosted decision stumps).

# Adaboost

Input: Training data  $(X_i, Y_i)_{i=1}^n$ , binary-valued Weak Learner,  
Number of iterations  $T$ .

Initialize weights:  $\gamma_i^1 = \frac{1}{n}$ ,  $i = 1, \dots, n$

For  $t = 1, \dots, T$ ,

- 1) Fit the weak learner,  $f_t : \mathcal{X} \rightarrow \{-1, 1\}$ , with weights  $\gamma_i$ .

The weak learner uses the weighted zero-one loss

$$L(f) = \sum_{i=1}^n \gamma_i \mathbb{1}_{f(X_i) \neq Y_i}$$

- 2) compute the weighted error of  $f_t$ ,  $\text{err}_t = \sum_{i=1}^n \gamma_i \mathbb{1}_{f(X_i) \neq Y_i}$ ,

- 3) define  $c_t = \log \left( \frac{1 - \text{err}_t}{\text{err}_t} \right)$ ,

- 4) update the weights  $\gamma_i^t$ ,

$$\gamma_i^{t+1} = \gamma_i^t \exp(c_t \mathbb{1}_{Y_i \neq f_t(X_i)}),$$

- 5) renormalize so that  $\sum_{i=1}^n \gamma_i^{t+1} = 1$ .

Output: final classifier  $f(x) = \text{sign}(\sum_{t=1}^T c_t f_t(x))$ .

## Idea of Adaboost:

- in each step: fit weak learner to data,
- data which is misclassified gets higher weight  
     $\implies$  next weak learner will try harder to fit misclassified points,
- contribution  $c_t$  of each weak classifier  $f_t$  to the final hypothesis,

$$c_t = \log \left( \frac{1 - \text{err}_t}{\text{err}_t} \right)$$

- $\implies$  strictly monotonically decreasing with  $\text{err}_t$ .
- final classifier:  $f(x) = \text{sign} \left( \sum_{t=1}^T c_t f_t(x) \right)$ ,  
 $\implies$  Adaboost learns a point in the vector space of functions generated by the weak learner.

## Decision stump as weak learner:

$$f_t(x) = 2\mathbb{1}_{x_i+b>0} - 1 \quad \text{and} \quad f_t(x) = 2\mathbb{1}_{-x_i+b>0} - 1.$$

or more generally,

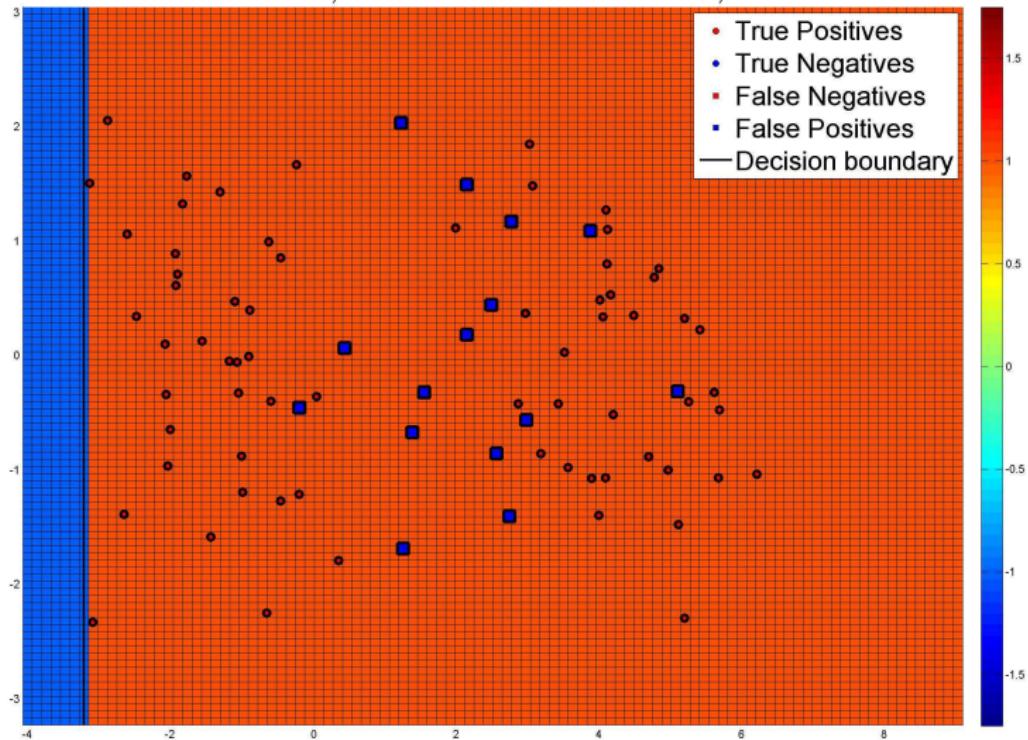
$$f_t(x) = 2\mathbb{1}_{\langle w, x \rangle + b > 0} - 1.$$

## Selection of the weak learner:

- random (e.g. randomly select coordinate or direction  $w$ )
- compute for all choices weak learner (if possible) and take the one with the smallest weighted error.

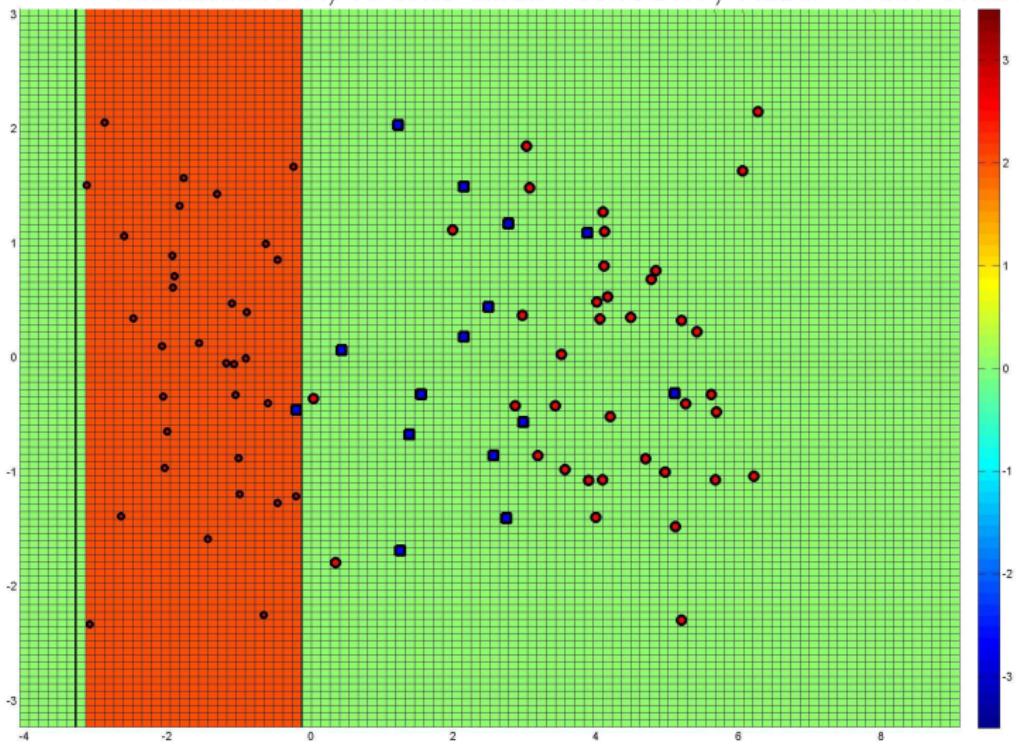
# Adaboost in action

Iteration: 1, TrainError: 0.17647, TestError: 0.19765



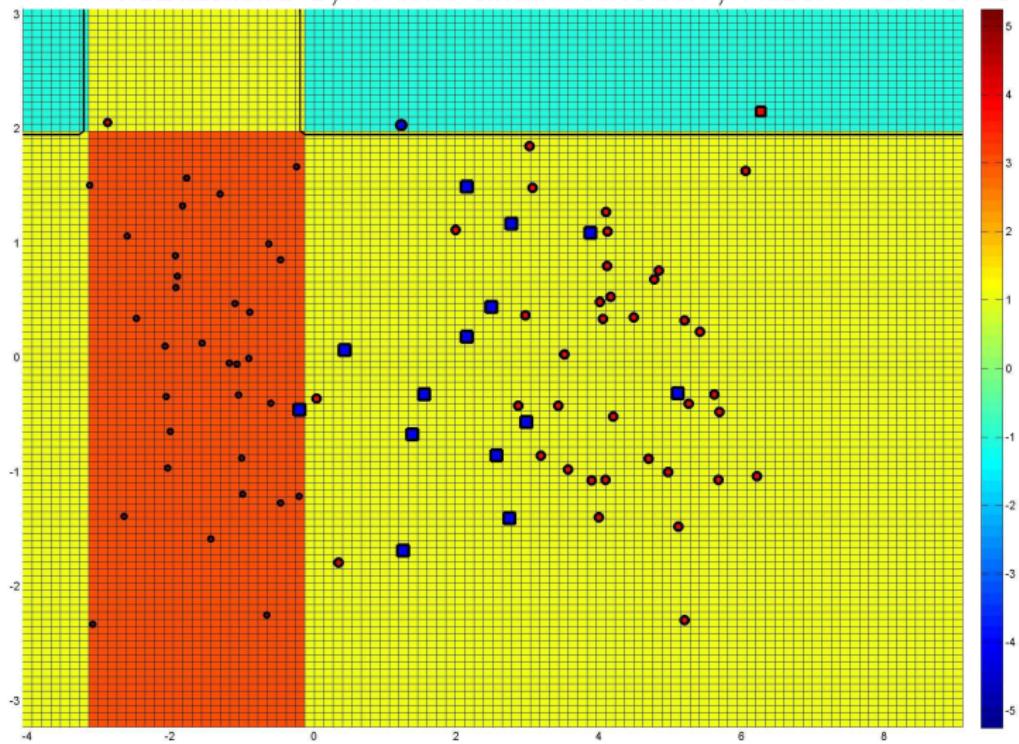
# Adaboost in action

Iteration: 2, TrainError: 0.17647, TestError: 0.19765



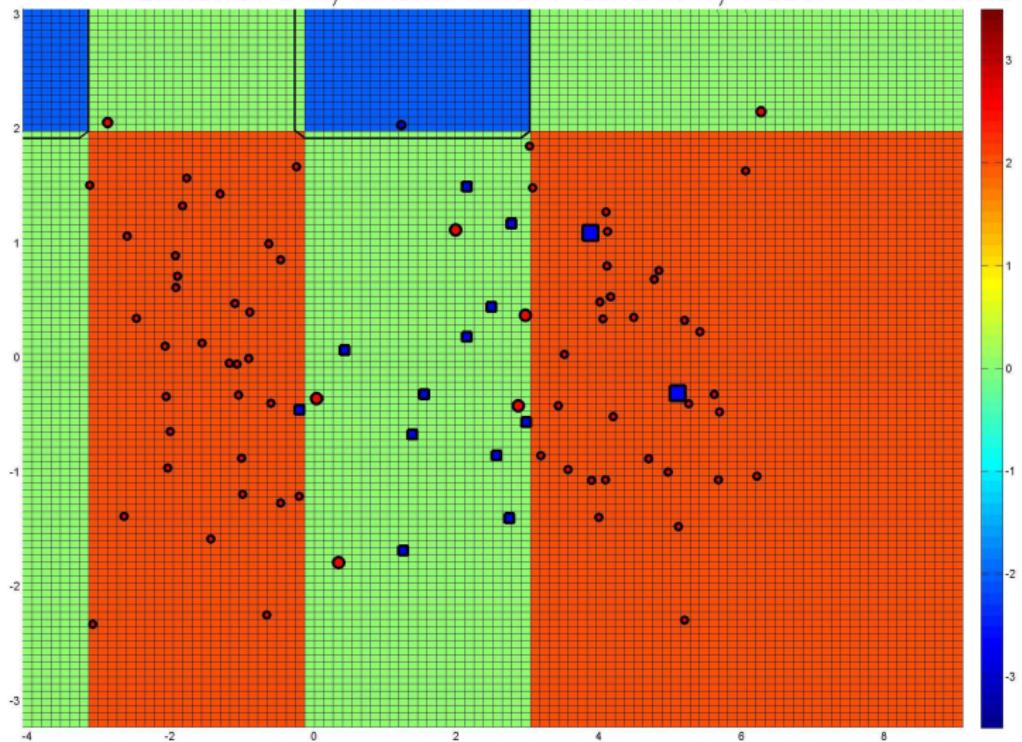
# Adaboost in action

Iteration: 3, TrainError: 0.17647, TestError: 0.18353



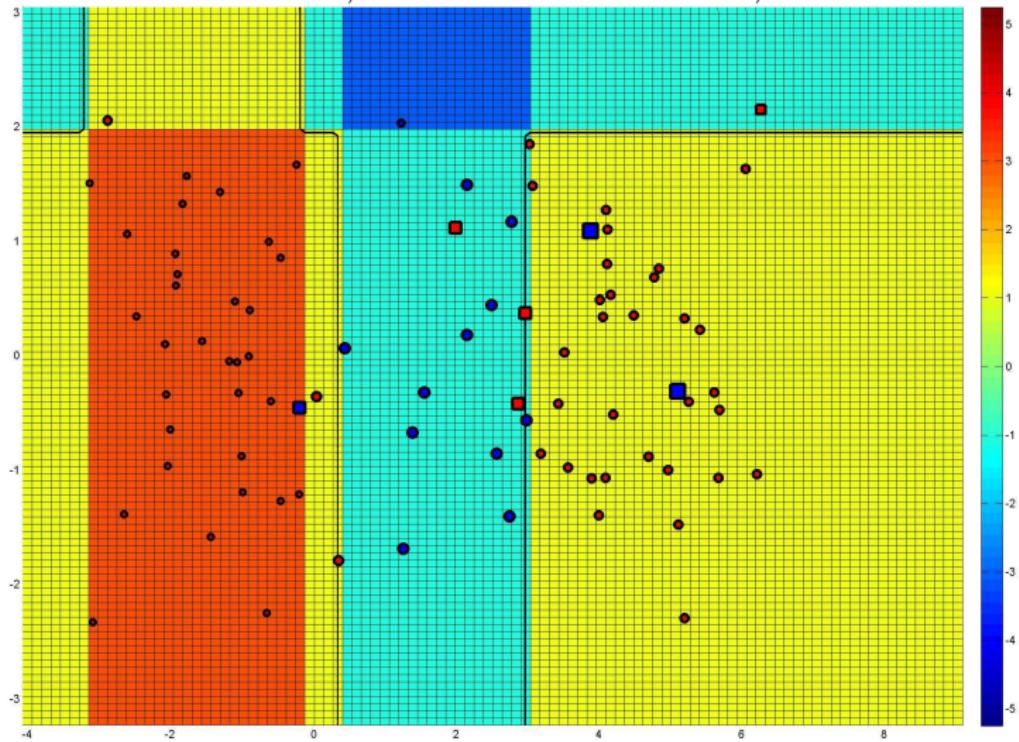
# Adaboost in action

Iteration: 4, TrainError: 0.16471, TestError: 0.19412



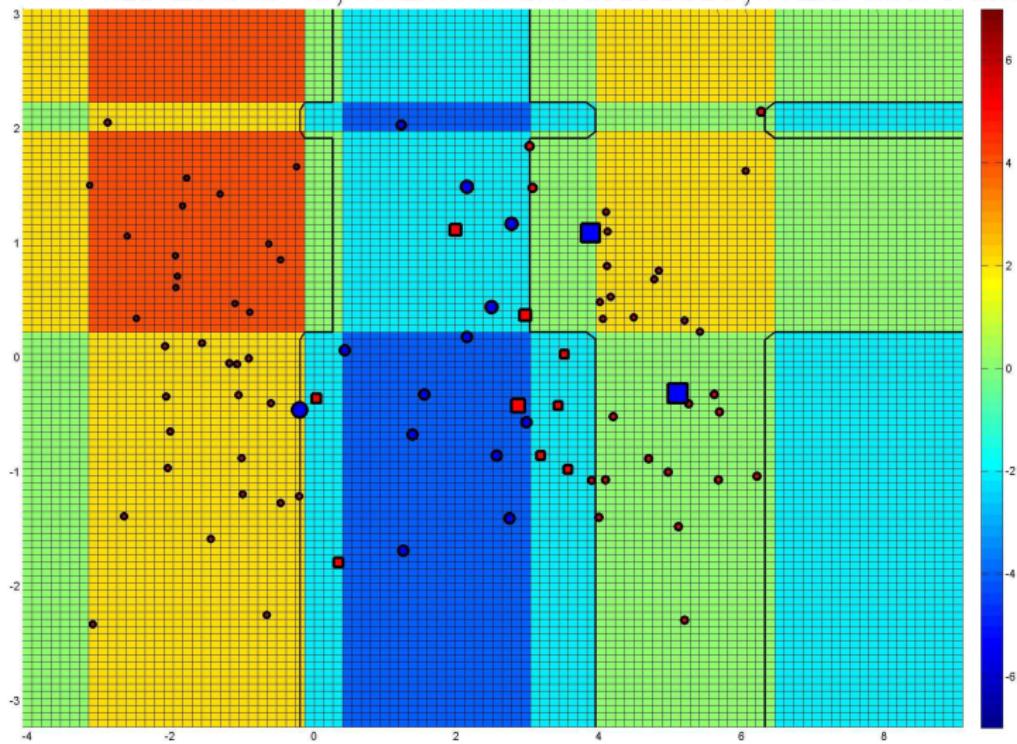
# Adaboost in action

Iteration: 5, TrainError: 0.070588, TestError: 0.12



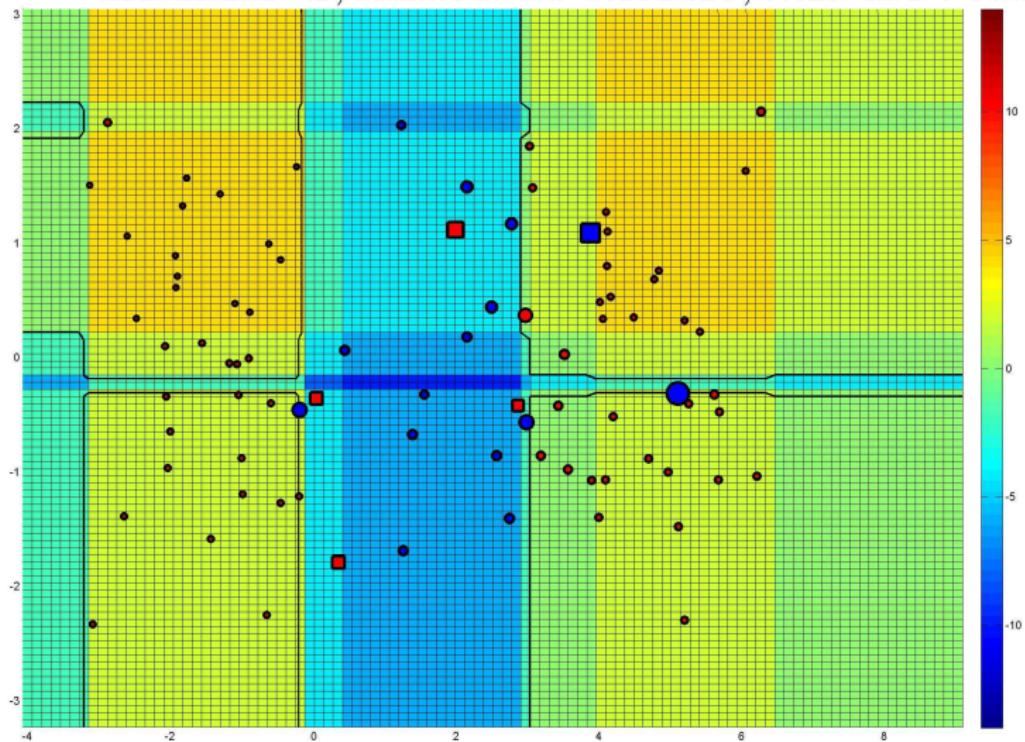
# Adaboost in action

Iteration: 10, TrainError: 0.070588, TestError: 0.10235



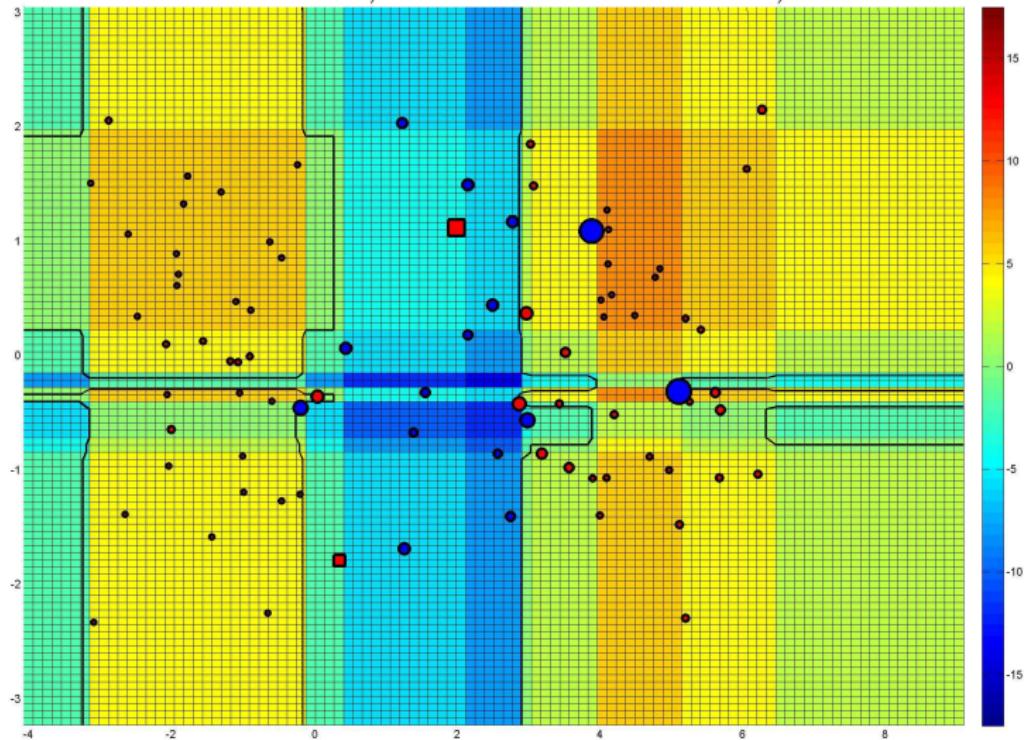
# Adaboost in action

Iteration: 20, TrainError: 0.070588, TestError: 0.11529



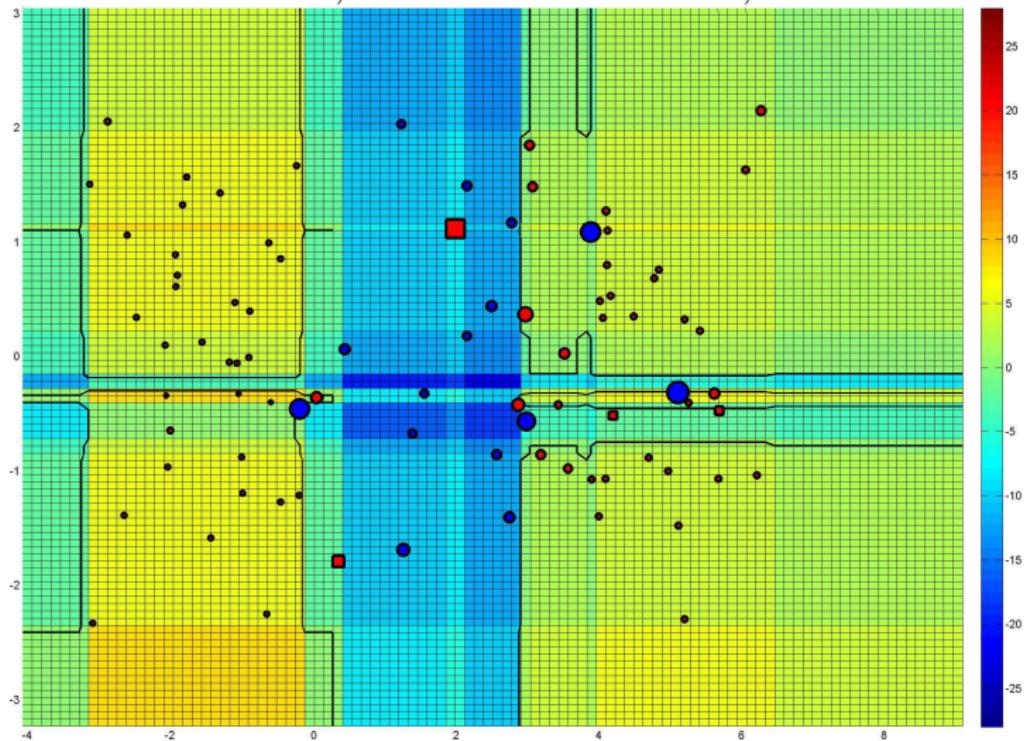
# Adaboost in action

Iteration: 40, TrainError: 0.035294, TestError: 0.1



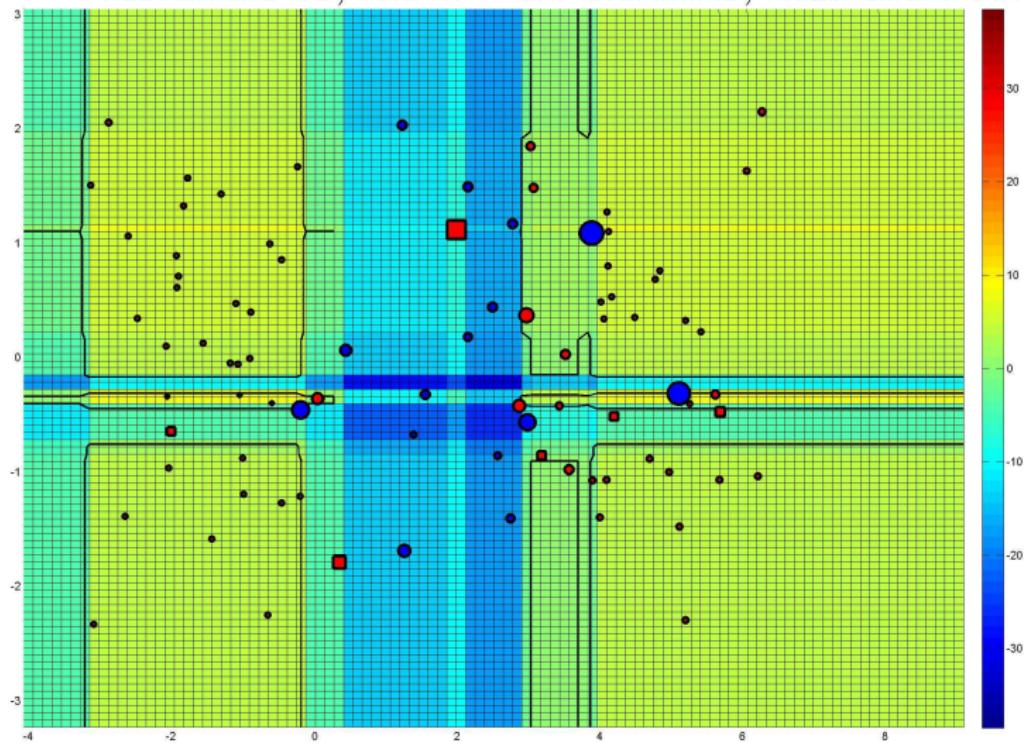
# Adaboost in action

Iteration: 60, TrainError: 0.023529, TestError: 0.10353



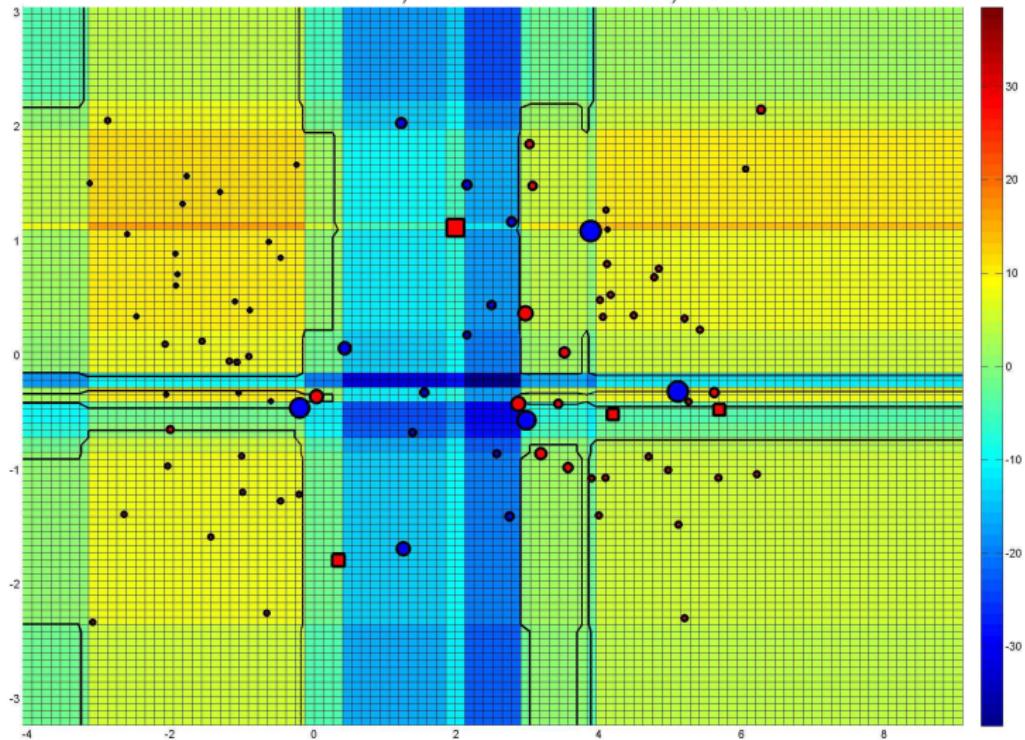
# Adaboost in action

Iteration: 80, TrainError: 0.023529, TestError: 0.10353



# Adaboost in action

Iteration: 100, TrainError: 0, TestError: 0.12118



# Adaboost in action



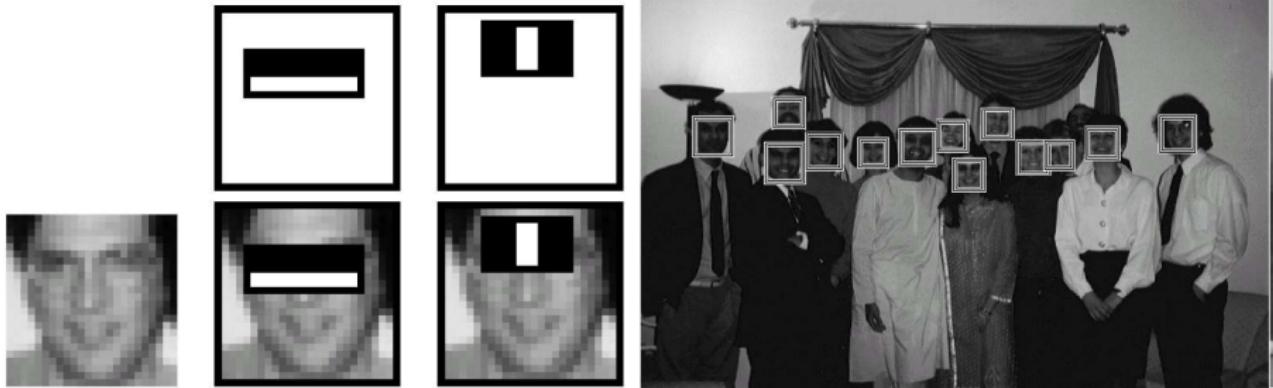
- final classification using decision stump can be done extremely fast  
     $\implies$  very interesting for applications where real-time performance is required.
- Most successful application of Adaboost in face recognition by Viola and Jones (2001).

They use more generally weak learner of the form,

$$f_t(x) = 2\mathbb{1}_{\langle w, x \rangle + b > 0} - 1$$

but coefficients  $w$  are restricted to be binary valued or zero (most coefficients of the weight vector  $w$  are zero).

# Face Recognition



## Interpretation of boosting

- Iterative updates,

$$F(x) \mapsto F(x) + c_t f_t(x),$$

can be understood as a descent in function space, where

- ➊  $f_t$  is a descent direction based on the current  $F$ ,
  - ➋  $c_t$  is the stepsize of the descent step.
- The objective which is minimized is the empirical exponential loss of the final classifier  $F$ ,

$$L(F) = \frac{1}{n} \sum_{i=1}^n e^{-Y_i F(X_i)}.$$

## Interpretation of boosting

- initiated simultaneously in the statistics and machine learning community,
- following Proposition is a variation of the result of Friedman, Hastie and Tibshirani (2000).

### Proposition

Suppose the weak learner  $f_t$  is binary-valued,  $f_t : \mathcal{X} \rightarrow \{-1, 1\}$ . The update step  $F_{t+1} = F_t + c_t f_t$  of the Adaboost algorithm, where  $c_t = \log\left(\frac{1-\text{err}_t}{\text{err}_t}\right)$  and  $\text{err}_t$  is the weighted zero-one loss, is a descent step in order to minimize the empirical exponential loss.

⇒ proof on blackboard !

# Result for Adaboost

## Corollary

*After each update of the weights, the weighted misclassification error of the most recent weak learner is 50%.*

**Proof:** This follows directly from the optimality condition for the parameter  $c$  derived in the last proof,

$$\mathbb{E}[Y f(X) e^{-Y F(X)} e^{-Y f(X)\alpha}] = 0.$$

Note, that the second factor corresponds to the new weights (up to normalization) and the weighted zero one loss of  $f$  is given as

$$\frac{\frac{1}{2} \mathbb{E}[(1 - Y f(X)) e^{-Y F(X)} e^{-Y f(X)\alpha}]}{\mathbb{E}[e^{-Y F(X)} e^{-Y f(X)\alpha}]} = \frac{1}{2}.$$

Theory has motivated many variants of boosting  $\implies$  **GentleBoost**

Input: Training data  $(X_i, Y_i)_{i=1}^n$ , **real-valued** Weak Learner,  
Number of iterations  $T$ .

Initialize weights:  $\gamma_i^1 = \frac{1}{n}$ ,  $i = 1, \dots, n$ .

For  $t = 1, \dots, T$ ,

- 1) Fit the weak learner,  $f_t : \mathcal{X} \rightarrow \mathbb{R}$ , with weights  $\gamma_i$ .

The weak learner uses weighted squared loss

$$L(f) = \sum_{i=1}^n \gamma_i (Y_i - f_t(X_i))^2,$$

- 2) update the weights  $\gamma_i^t$ ,

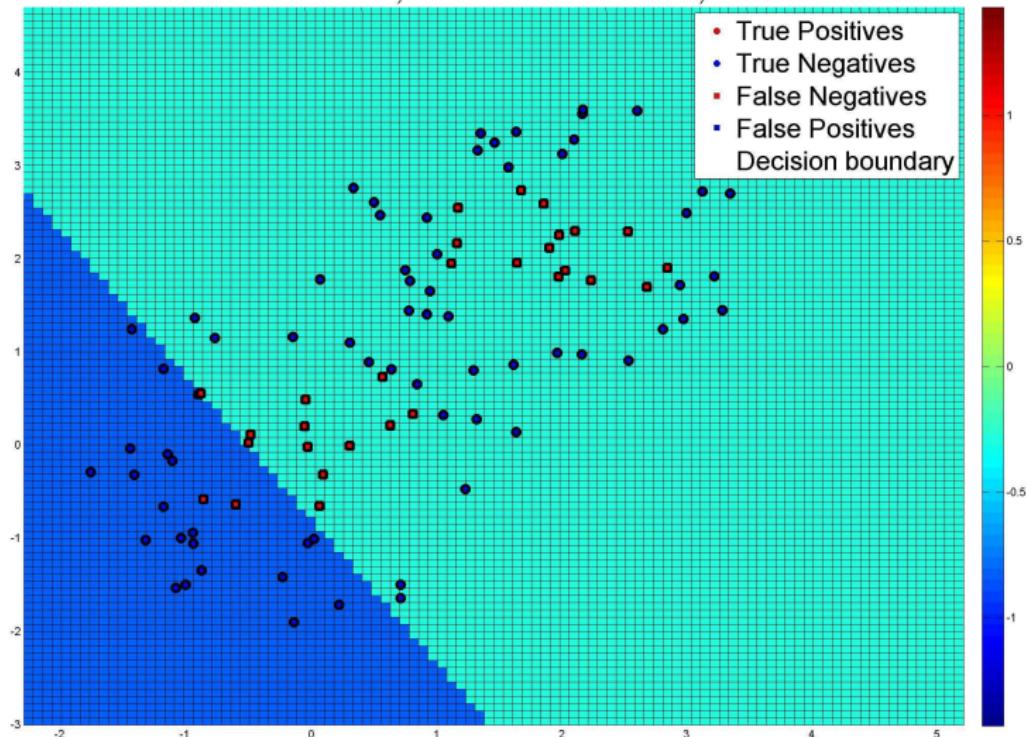
$$\gamma_i^{t+1} = \gamma_i^t \exp(-Y_i f_t(X_i)),$$

- 5) renormalize so that  $\sum_{i=1}^n \gamma_i^{t+1} = 1$ .

Output: final classifier  $f(x) = \text{sign}(\sum_{t=1}^T f_t(x))$ .

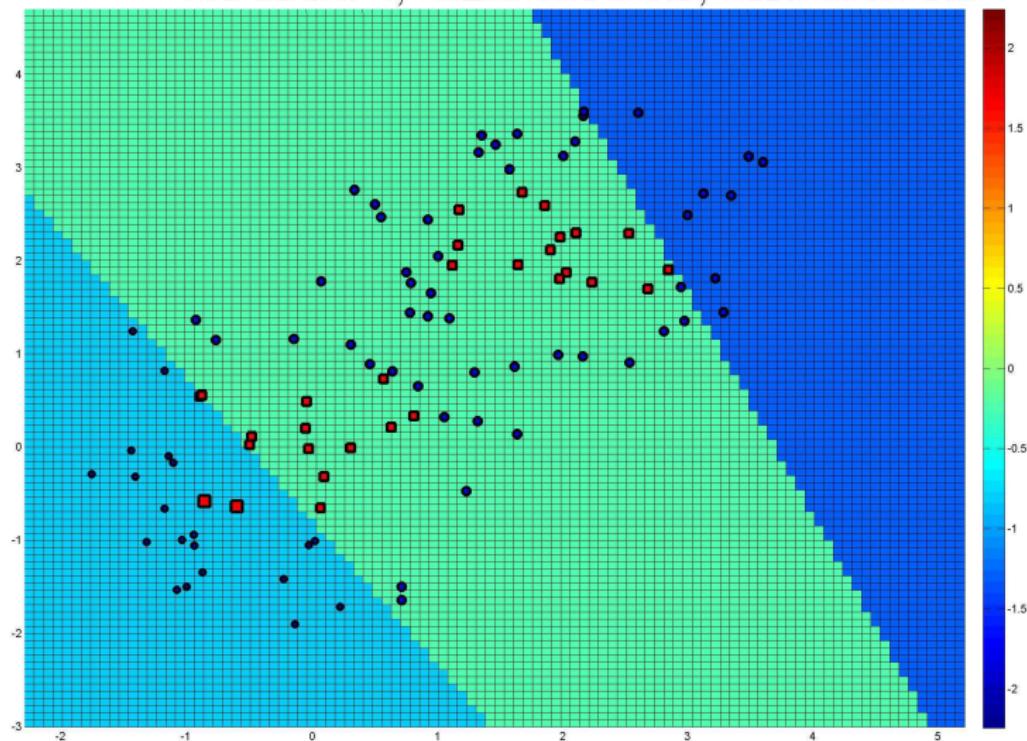
# GentleBoost in action

Iteration: 1, TrainError: 0.3, TestError: 0.3



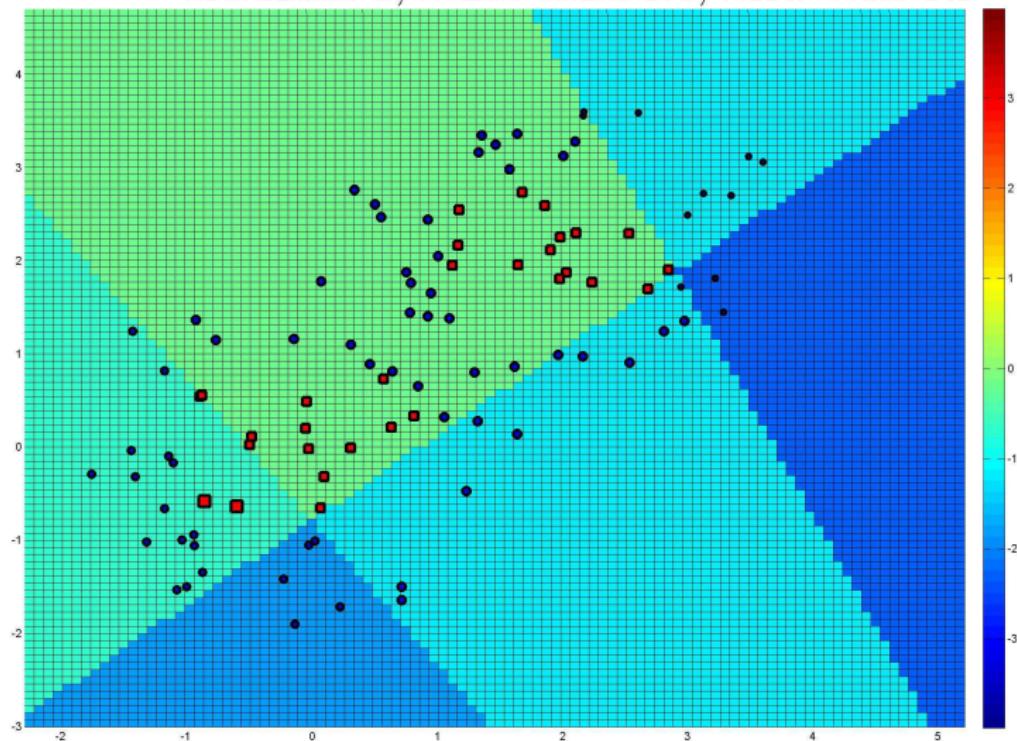
# GentleBoost in action

Iteration: 2, TrainError: 0.3, TestError: 0.3



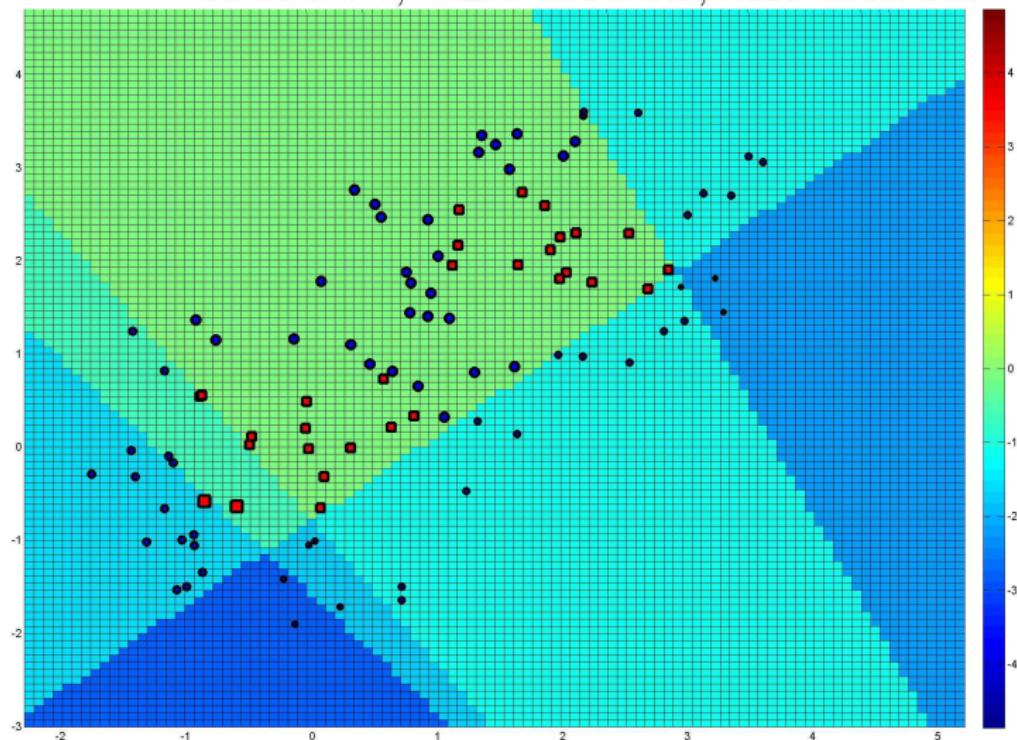
# GentleBoost in action

Iteration: 3, TrainError: 0.3, TestError: 0.3



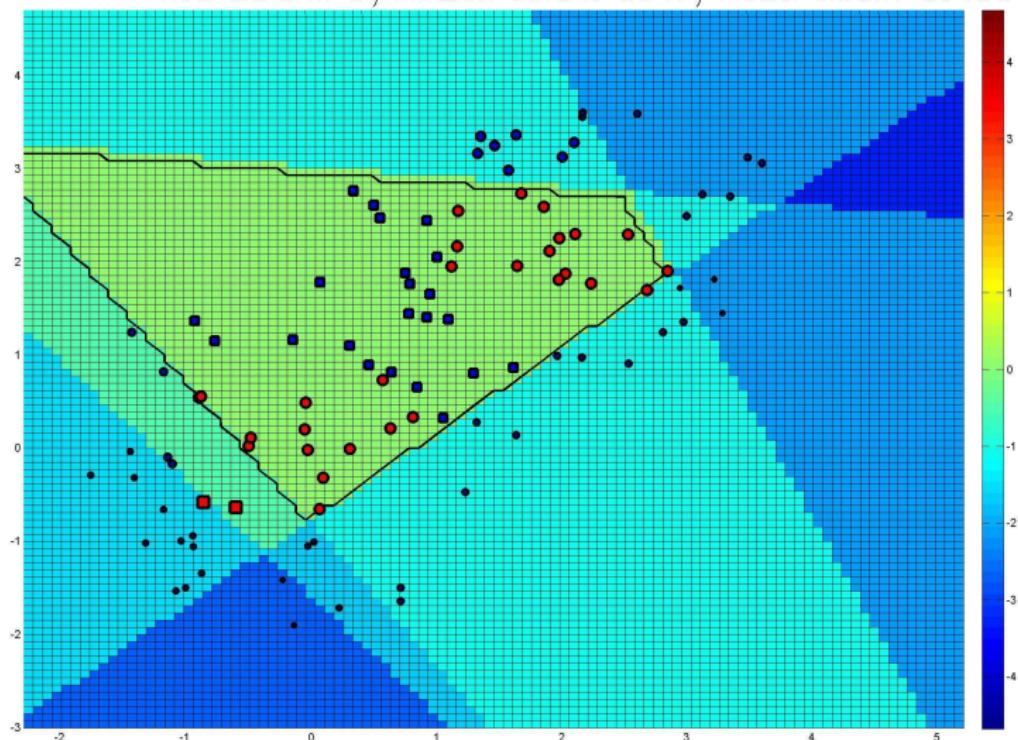
# GentleBoost in action

Iteration: 4, TrainError: 0.3, TestError: 0.3



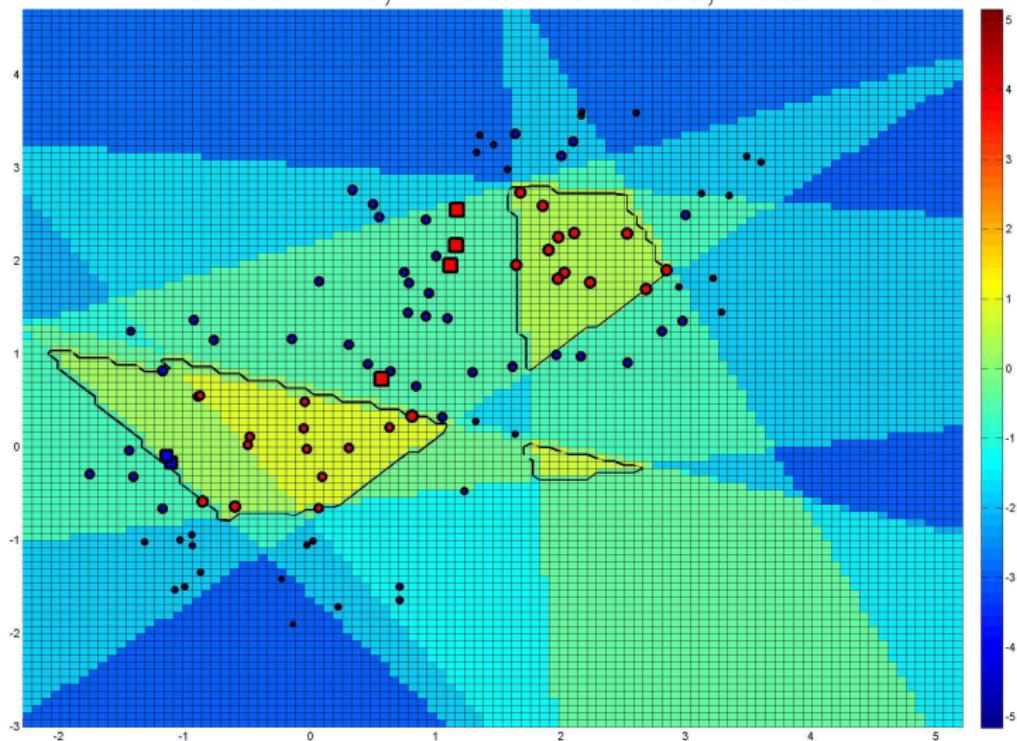
# GentleBoost in action

Iteration: 5, TrainError: 0.24, TestError: 0.266



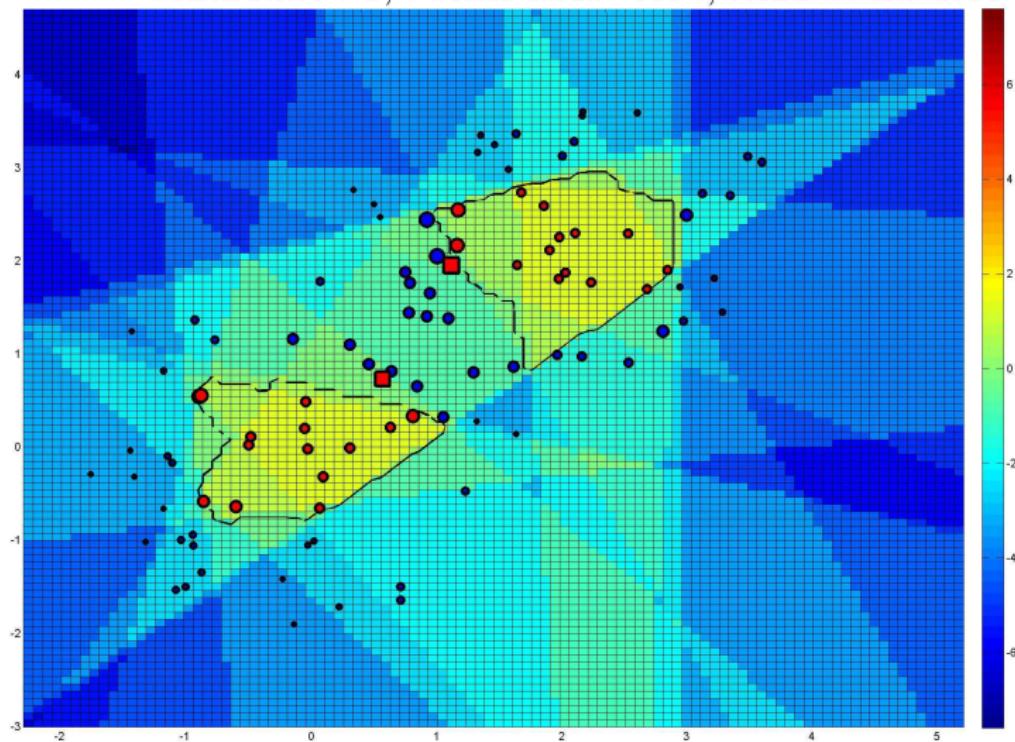
# GentleBoost in action

Iteration: 10, TrainError: 0.06, TestError: 0.122



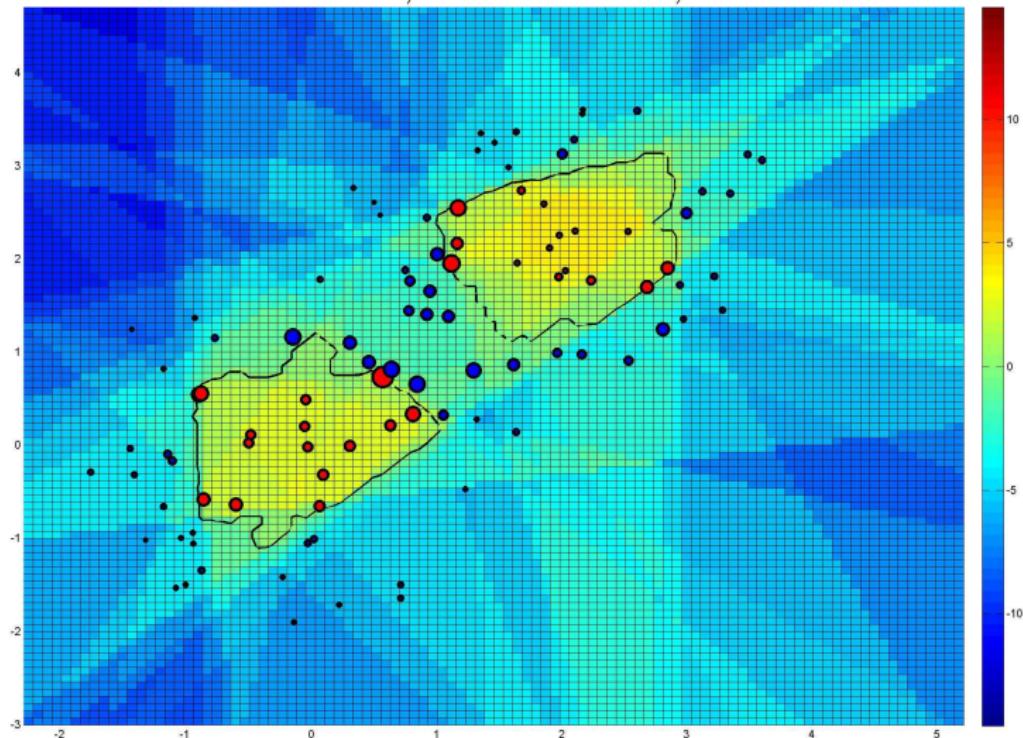
# GentleBoost in action

Iteration: 20, TrainError: 0.02, TestError: 0.071



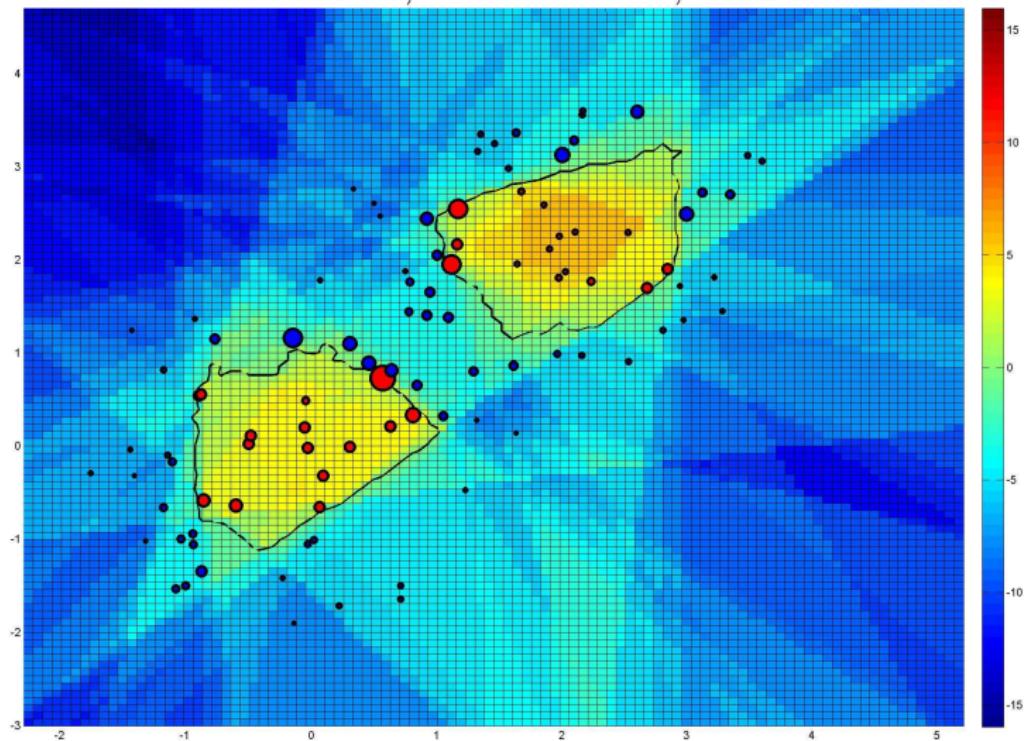
# GentleBoost in action

Iteration: 40, TrainError: 0, TestError: 0.056



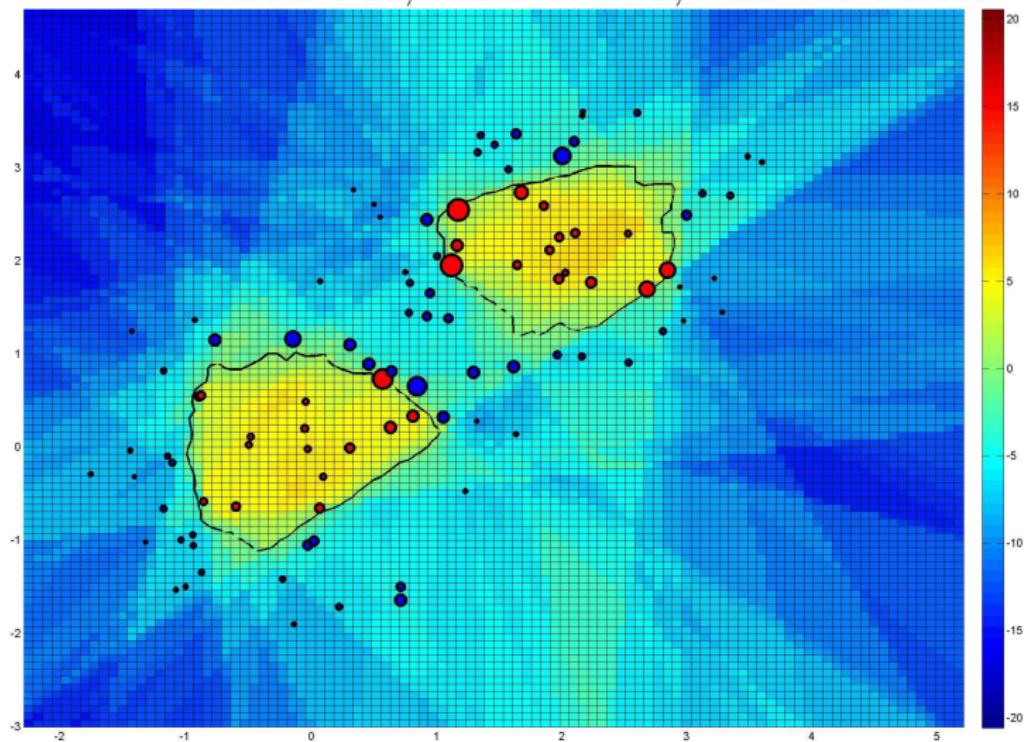
# GentleBoost in action

Iteration: 60, TrainError: 0, TestError: 0.059



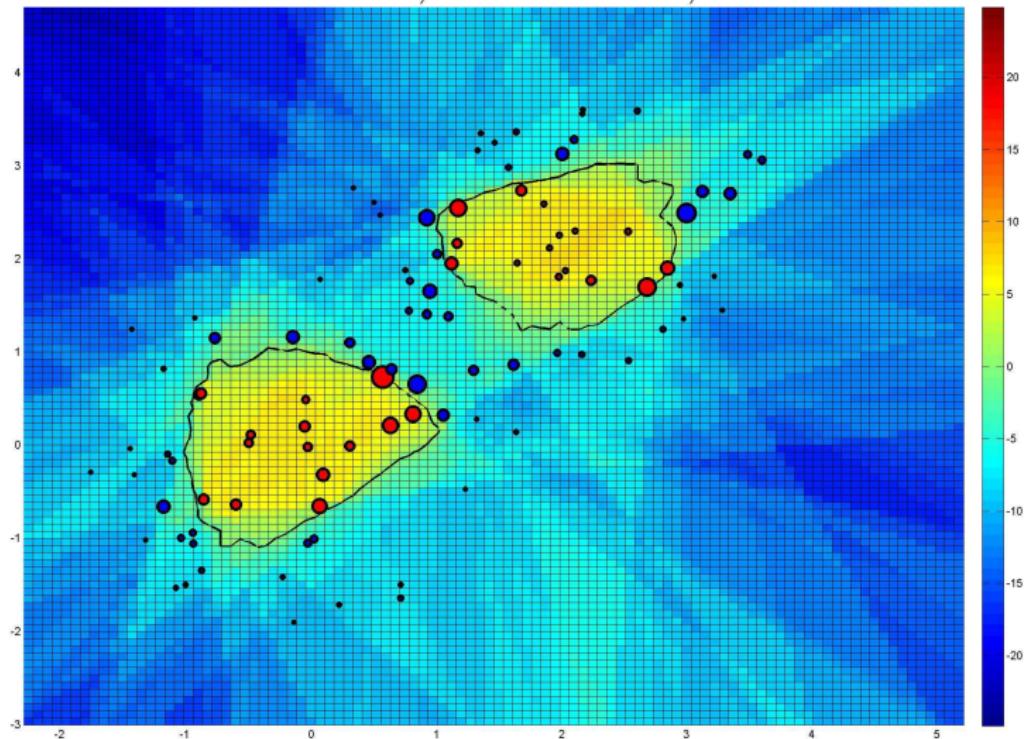
# GentleBoost in action

Iteration: 80, TrainError: 0, TestError: 0.057

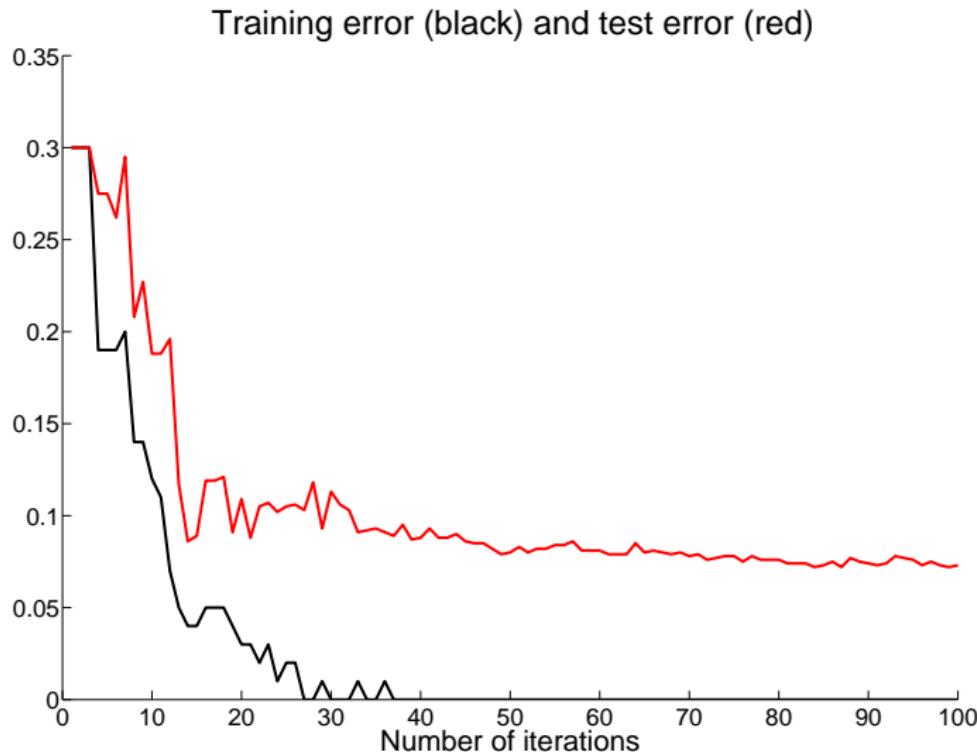


# GentleBoost in action

Iteration: 100, TrainError: 0, TestError: 0.062



# GentleBoost in action



## Proposition

Suppose the weak learner  $f_t$  is real-valued,  $f_t : \mathcal{X} \rightarrow \mathbb{R}$ . The update step  $F_{t+1} = F_t + f_t$  of the GentleBoost algorithm is an **approximate Newton step** in order to minimize the empirical exponential loss.

**Proof:** We can expand the risk of  $F + f$  up to second order,

$$R(F + f) = \mathbb{E}[e^{-Y(F(X) + f(X))}] \approx \mathbb{E}[e^{-YF(X)}(1 - Yf(X) + \frac{1}{2}f(X)^2)].$$

The first term does not depend on  $f$  so we can modify it without changing the minimizer,

$$\mathbb{E}[e^{-YF(X)}(\frac{1}{2} - Yf(X) + \frac{1}{2}f(X)^2)] = \frac{1}{2}\mathbb{E}[e^{-YF(X)}(Y - f(X))^2],$$

which is up to the normalization of the weights  $\gamma_i = e^{-Y_i F(X_i)}$  equal to the weighted squared loss minimized by the weak learner  $\Rightarrow$  weak learner minimizes in each step a second order approximation of the exponential loss.