Ensemble Methods

Principle of ensemble classifier

- Ask experts for their opinion and choose the option with majority vote
- Let's say we have a set of m experts:

$$H = \{f1, f2, \dots, fm\} \quad fi(x) \in \{P, N\}$$

- The majority vote decision will be:

$$F(x) = sign\left(\frac{1}{m}\sum_{i=1}^{m} fi(x))\right)$$

- Diversity of the expert a key point of this approach

Why to use the ensemble methods

Better performance

Assume that $\forall i, p(f_i(x) \neq y) \leq \mu < 0.5$

Classifiers are independent then the probability of wrong classification by the ensemble

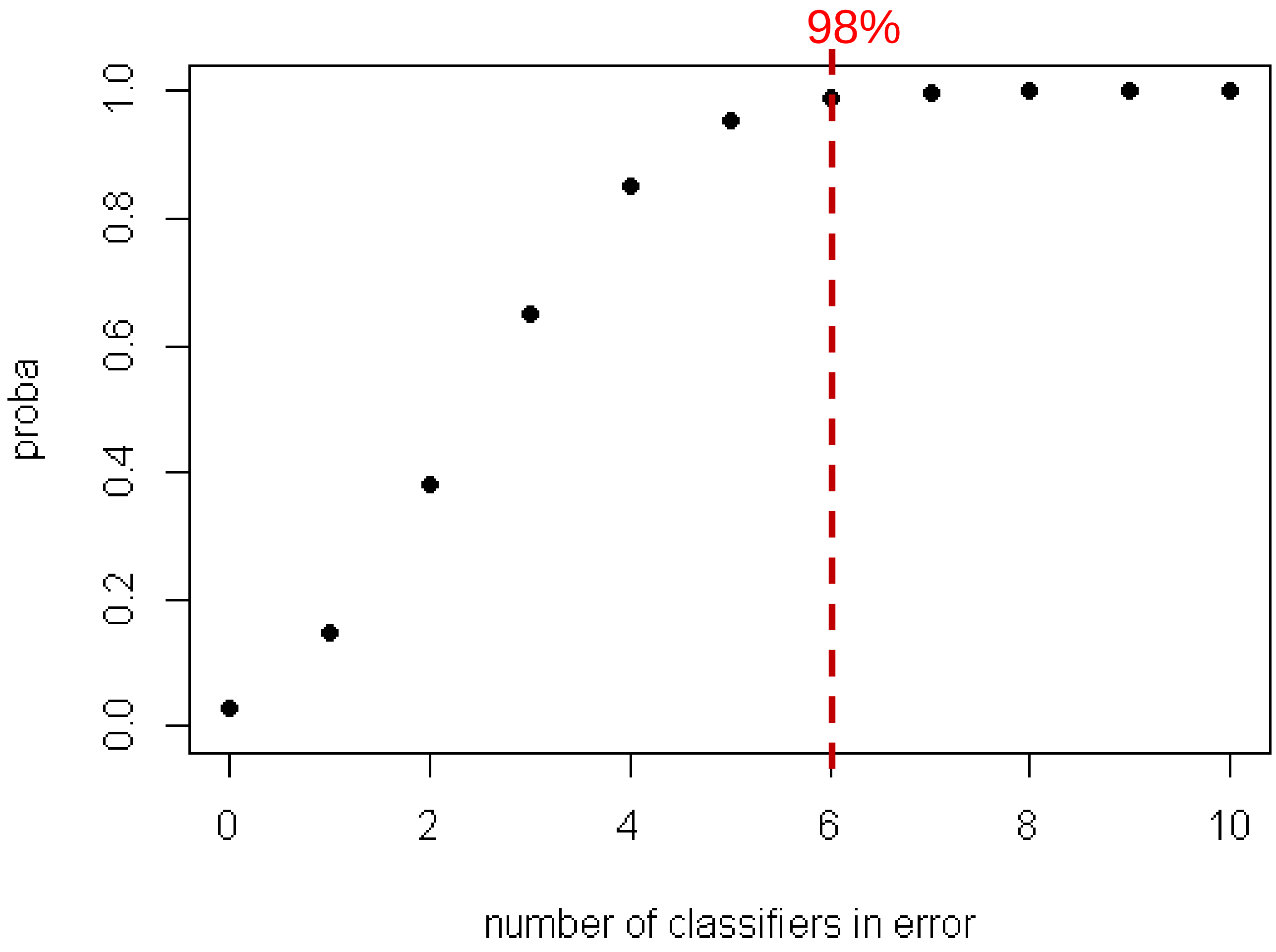
$$p(F(x) \neq y) = 1 - pr\left(k \leq \frac{M}{2}\right)$$

where pr is the cumulative binomial distribution

The upper bound is much better than the original error rate

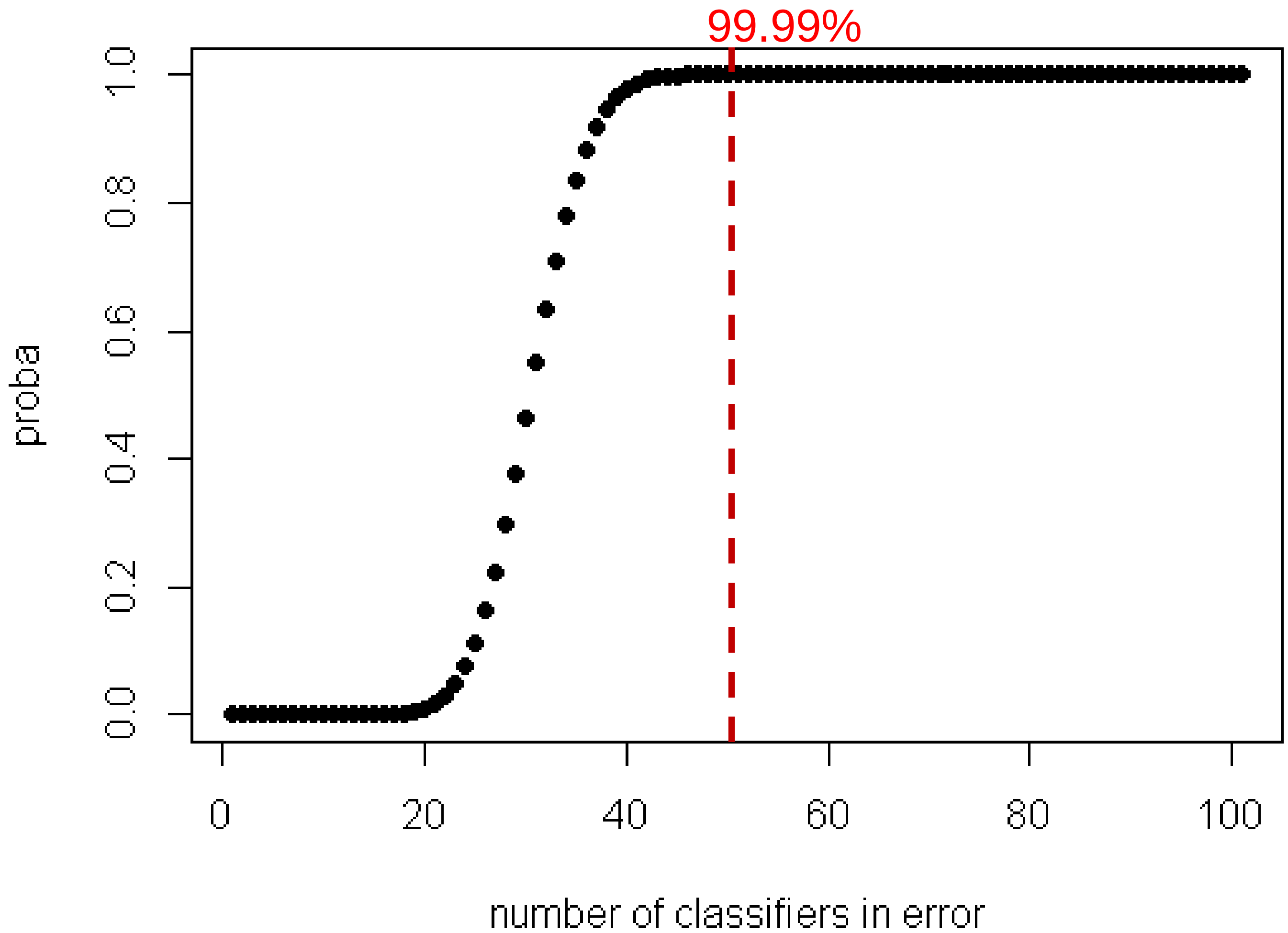
Why to use the ensemble methods

- For 10 classifiers with 30% of error



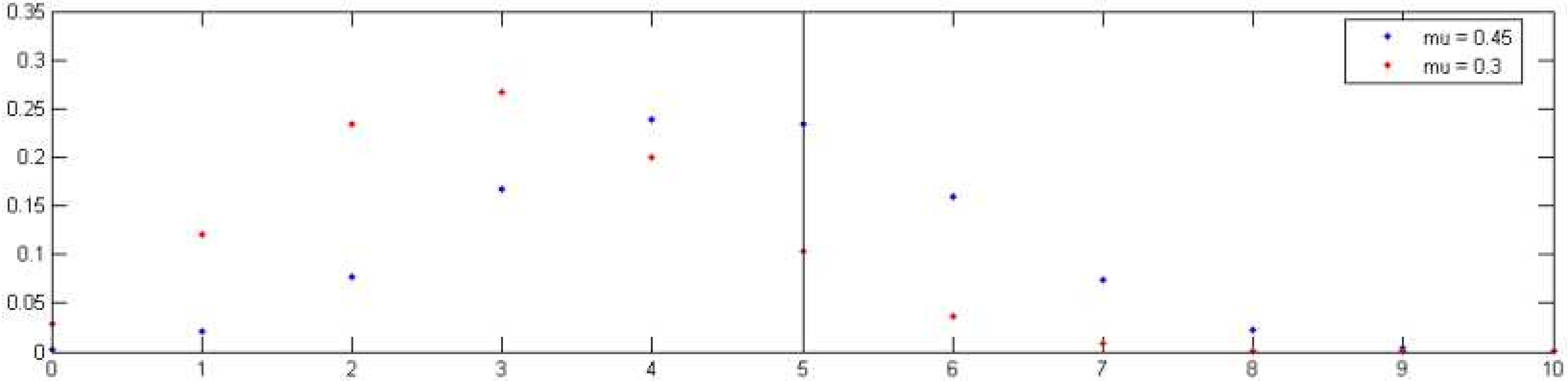
Why to use the ensemble methods

- For 100 classifiers with 30% of error

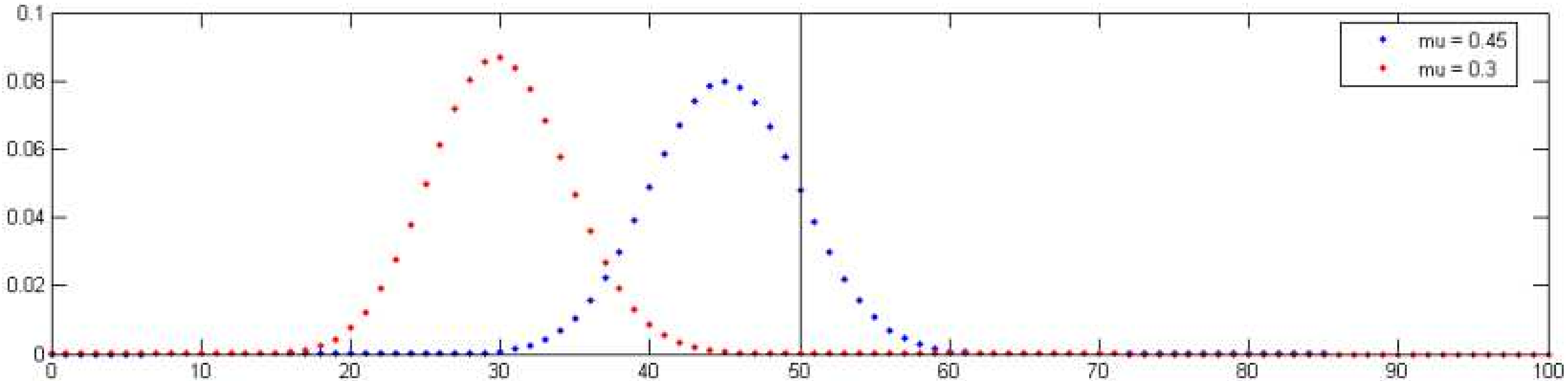


Why to use the ensemble methods

10 classifiers



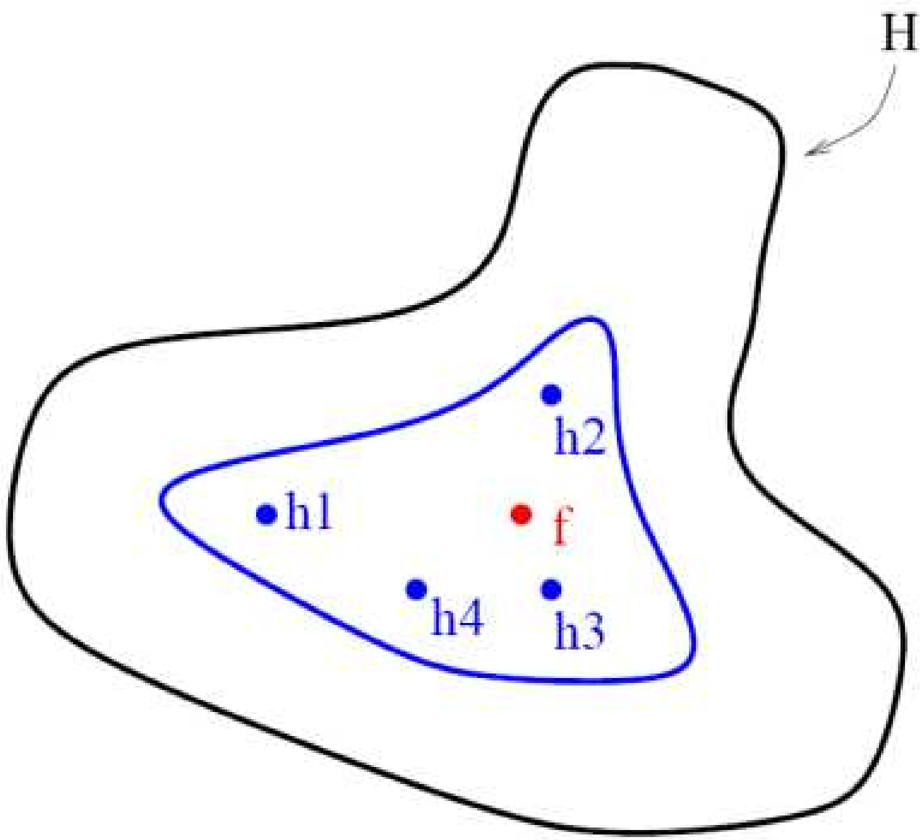
100 classifiers



Number of classifiers with the wrong decision

Why to use the ensemble methods

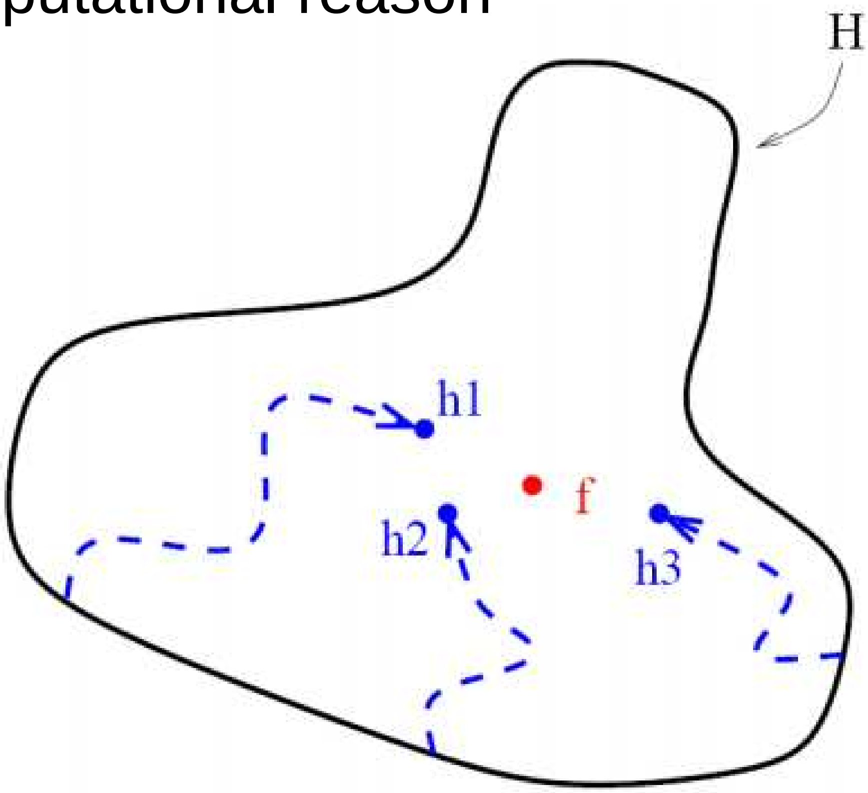
Statistical reason



From: T. G. Diettrich, Ensemble Methods in Machine Learning, Lecture Notes in Computer Science, Vol. 1857, pages: 1-15, 2000.

Why to use the ensemble methods

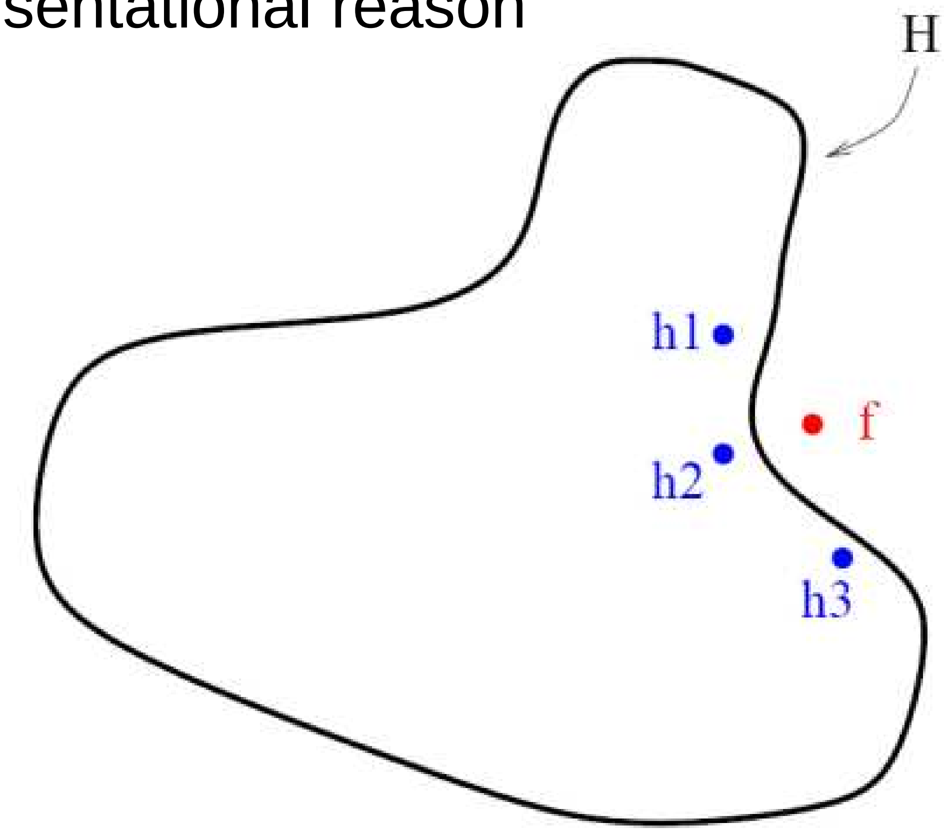
Computational reason



From: T. G. Diettrich, Ensemble Methods in Machine Learning, Lecture Notes in Computer Science, Vol. 1857, pages: 1-15, 2000.

Why to use the ensemble methods

Representational reason



From: T. G. Diettrich, Ensemble Methods in Machine Learning, Lecture Notes in Computer Science, Vol. 1857, pages: 1-15, 2000.

How to use ensemble methods

- Set of weak classifiers $p(f(x) \neq y) < 0.5$

  - Train diverse set of models on the same datasets: decision tree, KNN, linear discriminant (generally the simplest, the better).

  - Train different models by using diversity in datasets, parameters, initial conditions.

- Aggregation
  - Bagging
  - Boosting

Bagging

- Create bootstraped training sets, each containing examples drawn randomly with replacement from the original dataset.
- Train a classifier for each bootstrap dataset
- The final decision is a vote of all classifiers
- Originally developed to reduce the variance of the classifier

Results

Error rate

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	22%
soybean	14.5	10.6	27%

Bias-Variance decomposition

- The error rate can be decomposed into a bias and variance term
- Reduction of the variance
- Example : heart dataset

	unaggregated	aggregated
variance	47.64	4.66
bias	1.51	1.97

Adaboost - Adaptive Boosting

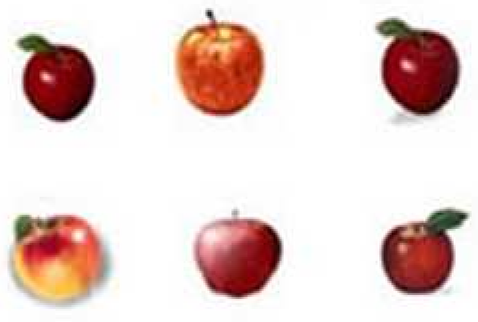
- The examples are weigthed
  - Each example has a weigth representing its importance in the classification problem.

- AdaBoost is an algorithm constructing complex classifier from a combination of simple and weak classifiers.

- The final decision is based on a weigthed vote of weak classifiers.

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Toy example

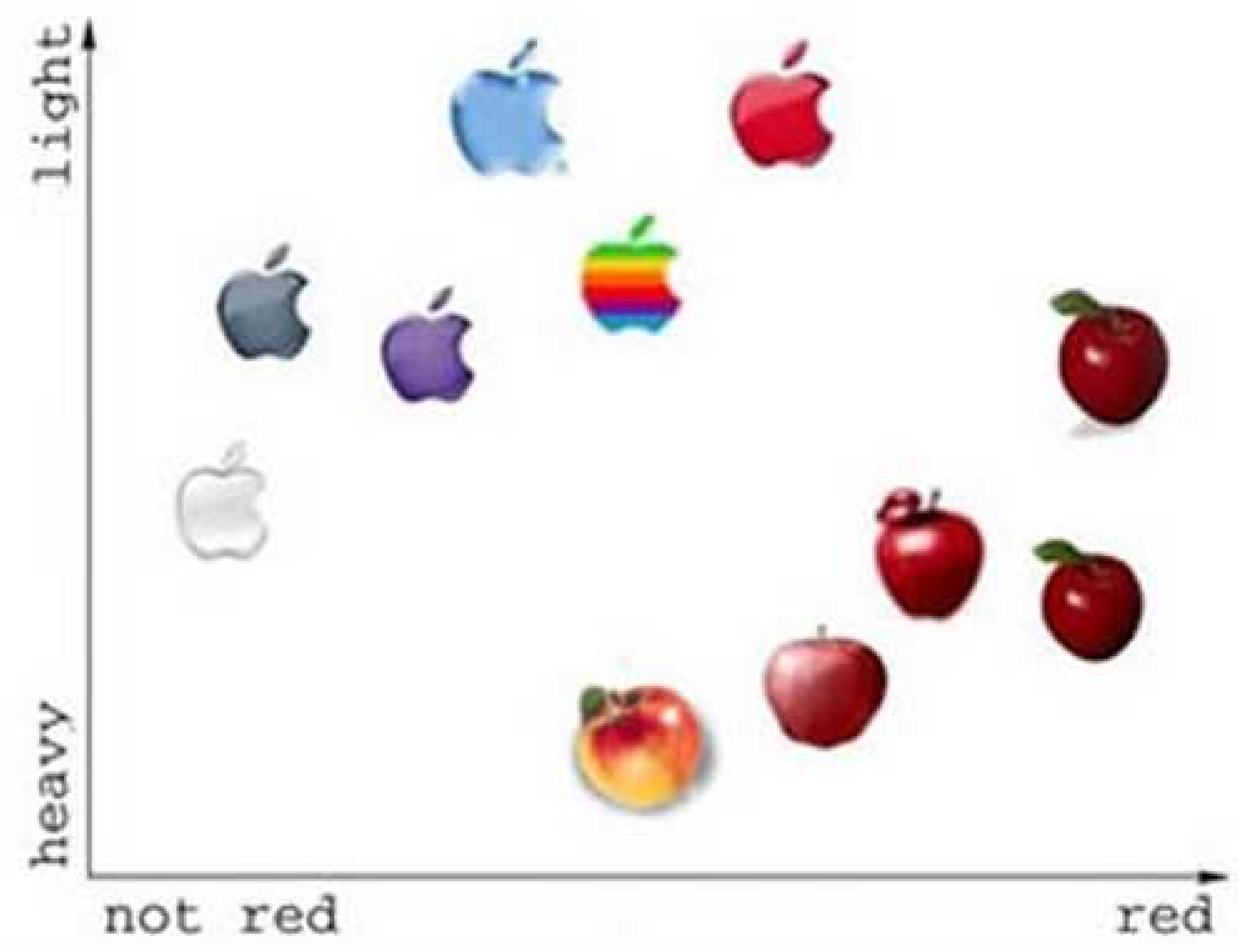


Natural

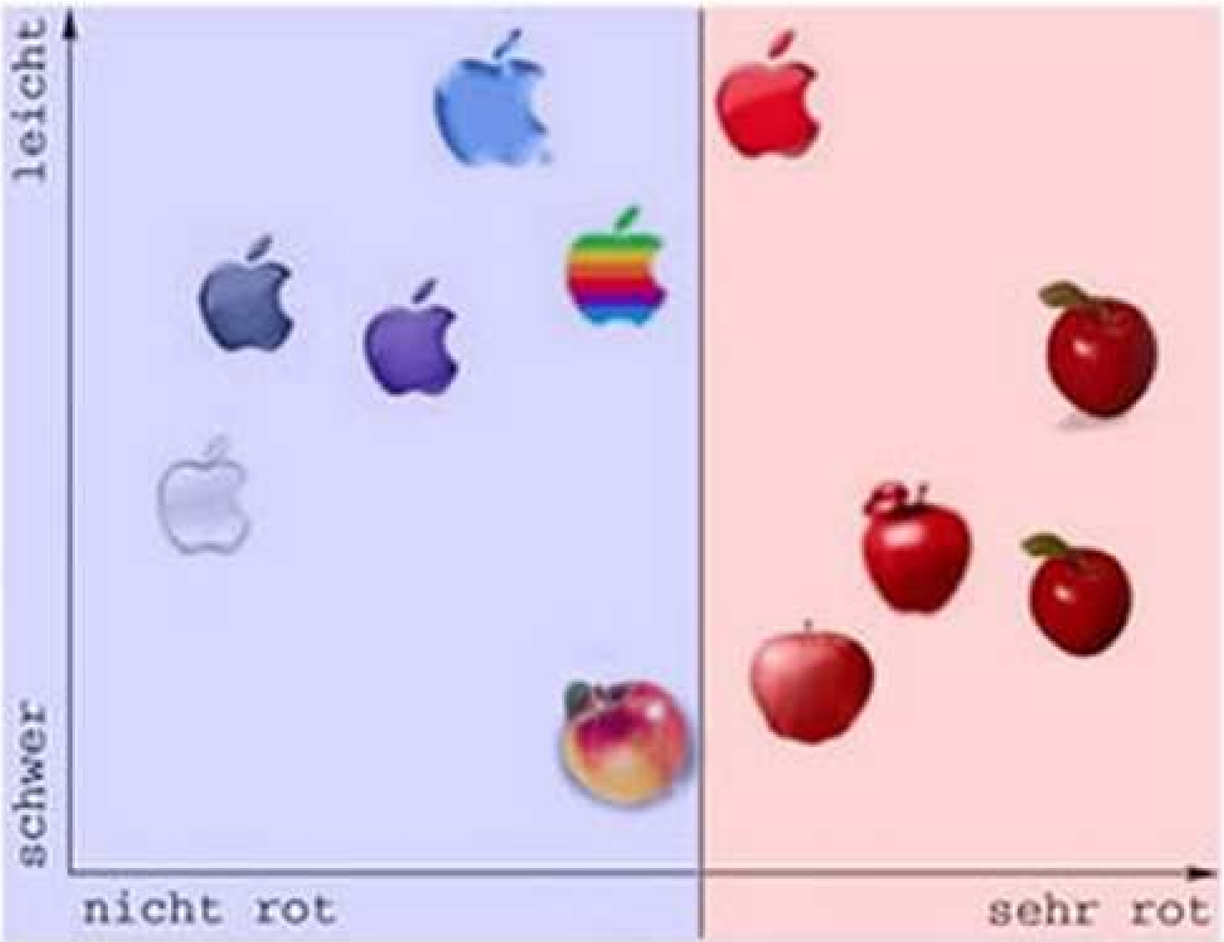
Not Natural

?

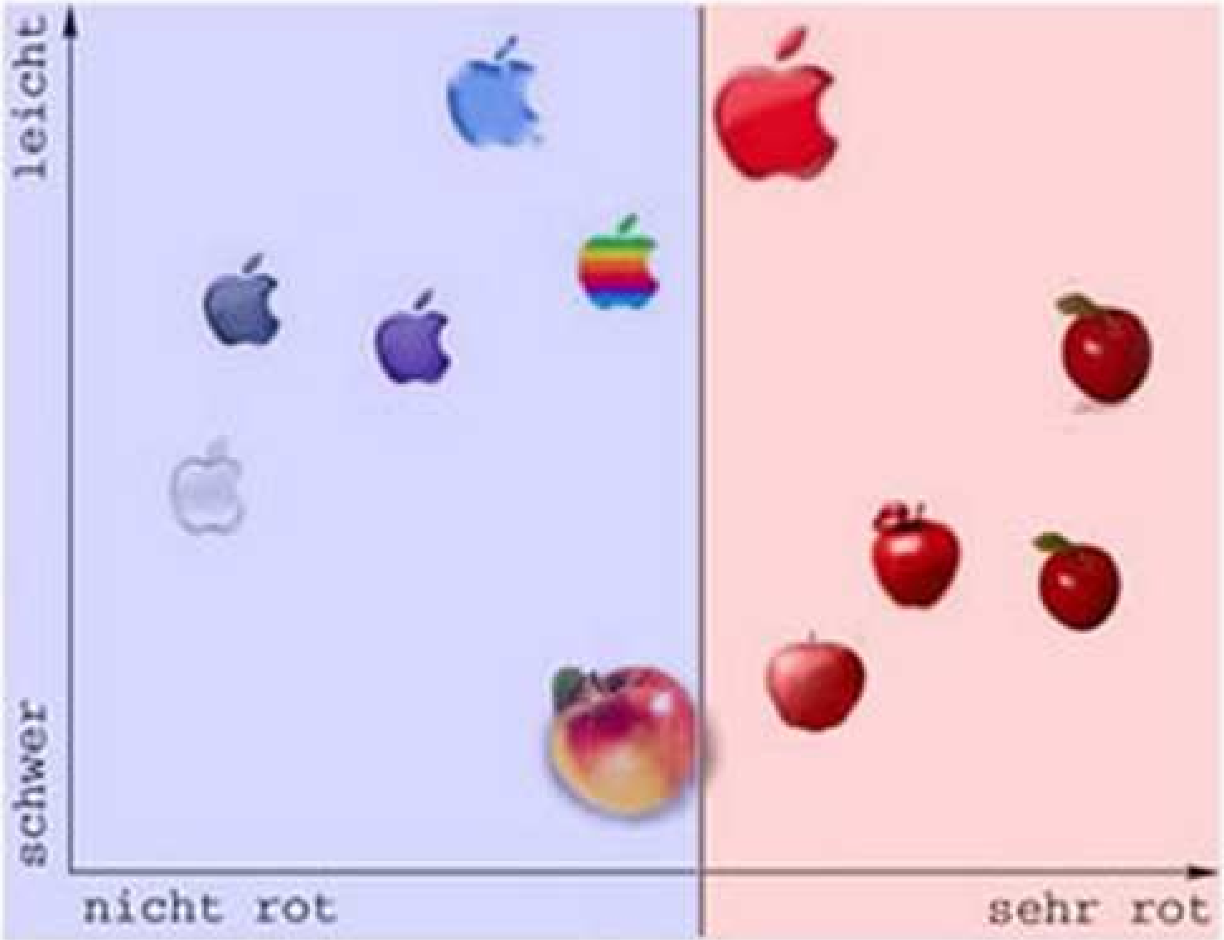
Toy example



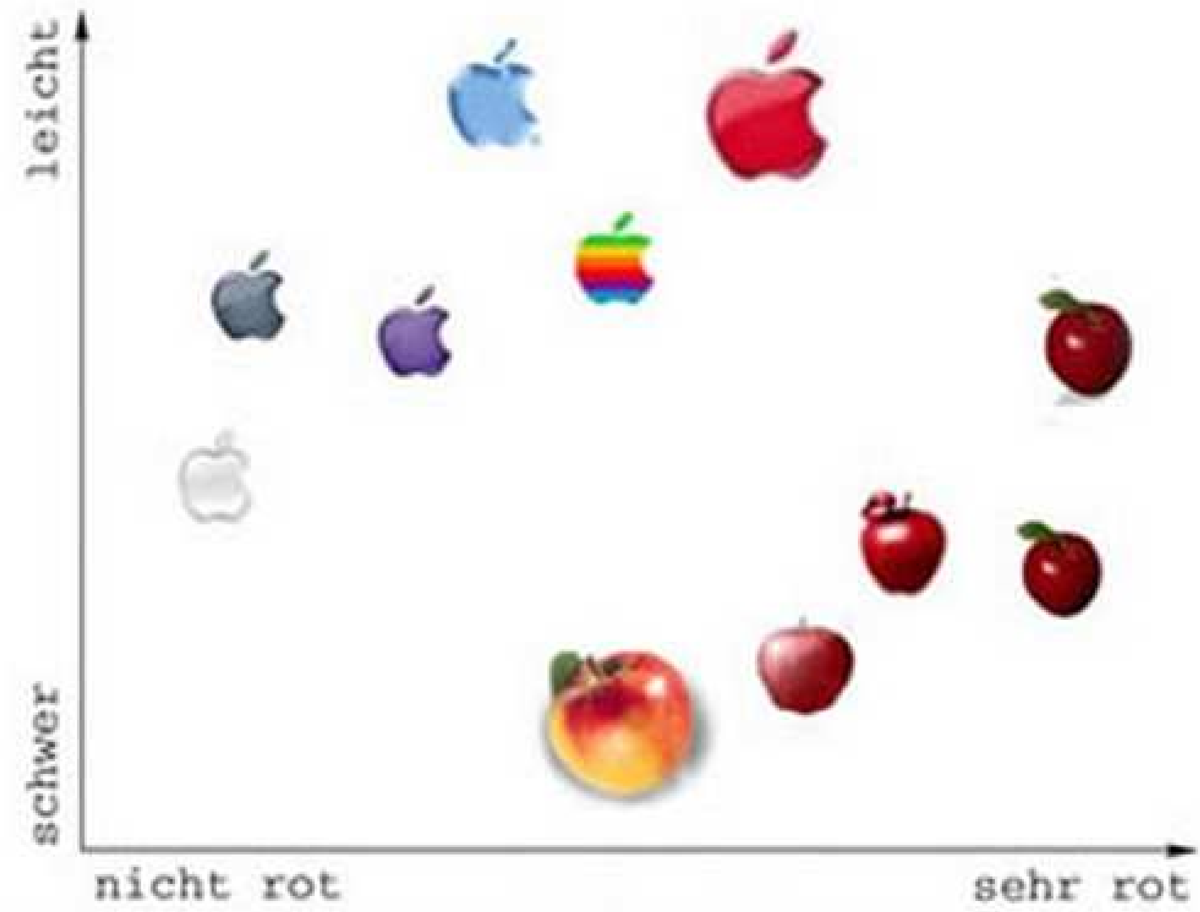
Toy example



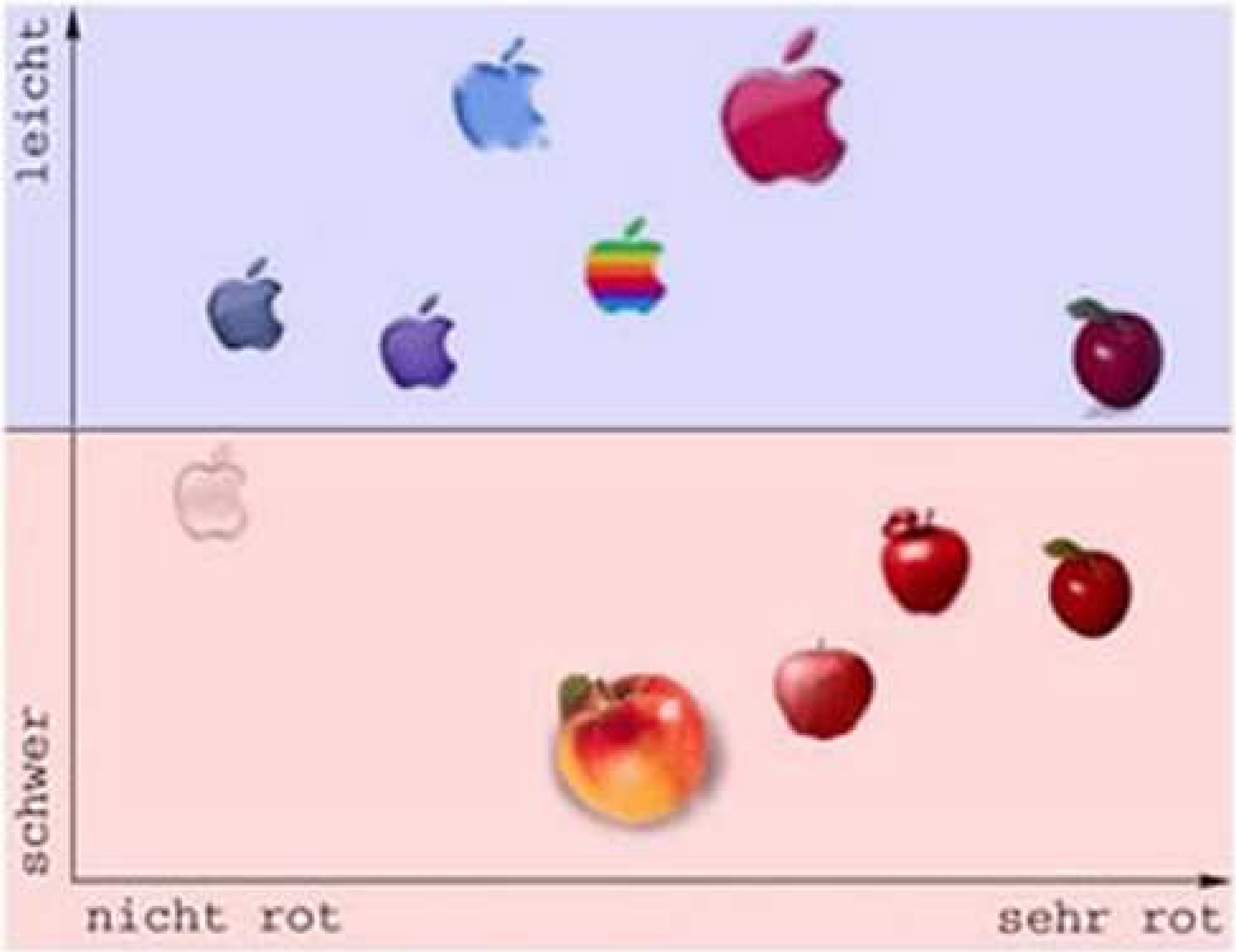
Toy example



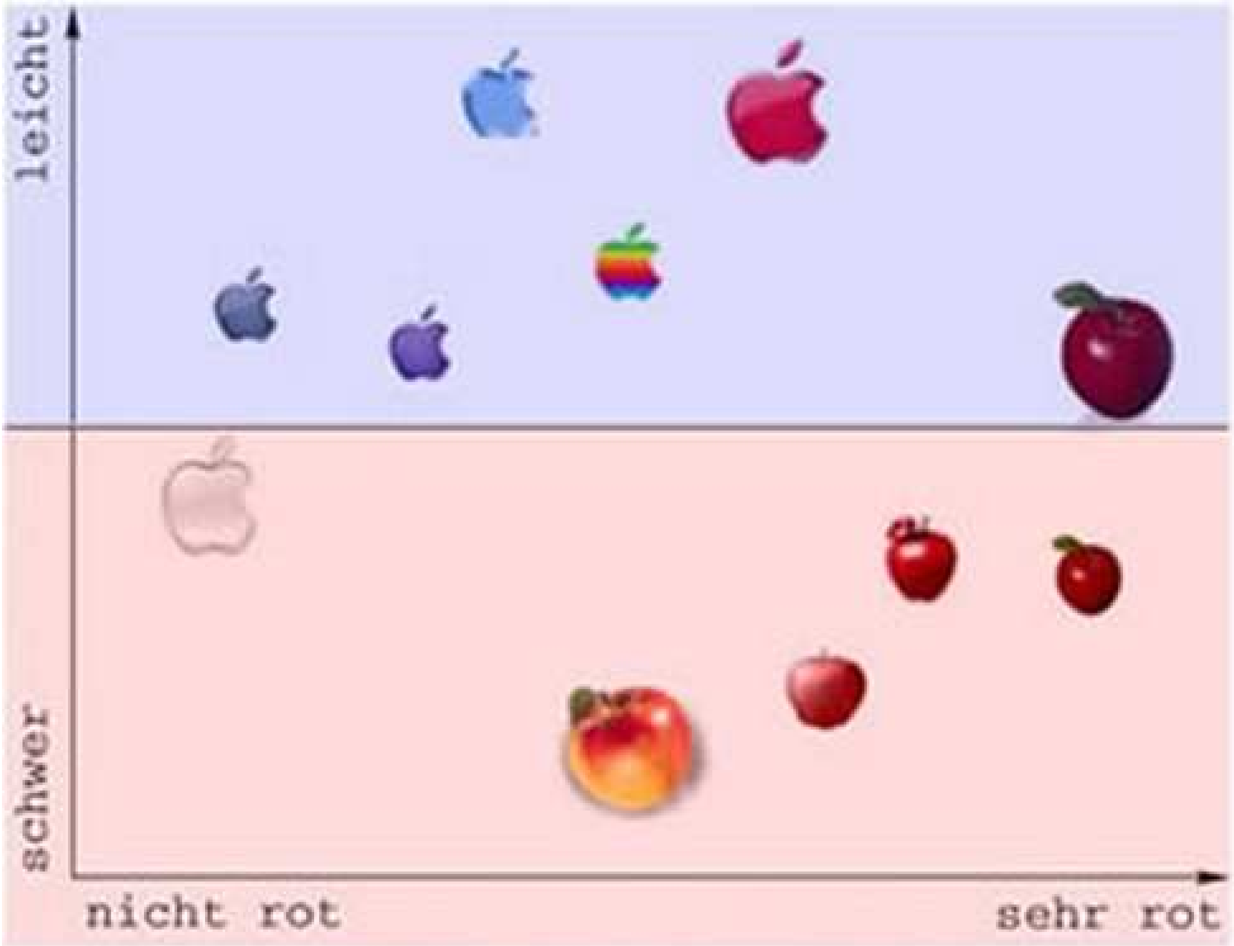
Toy example



Toy example

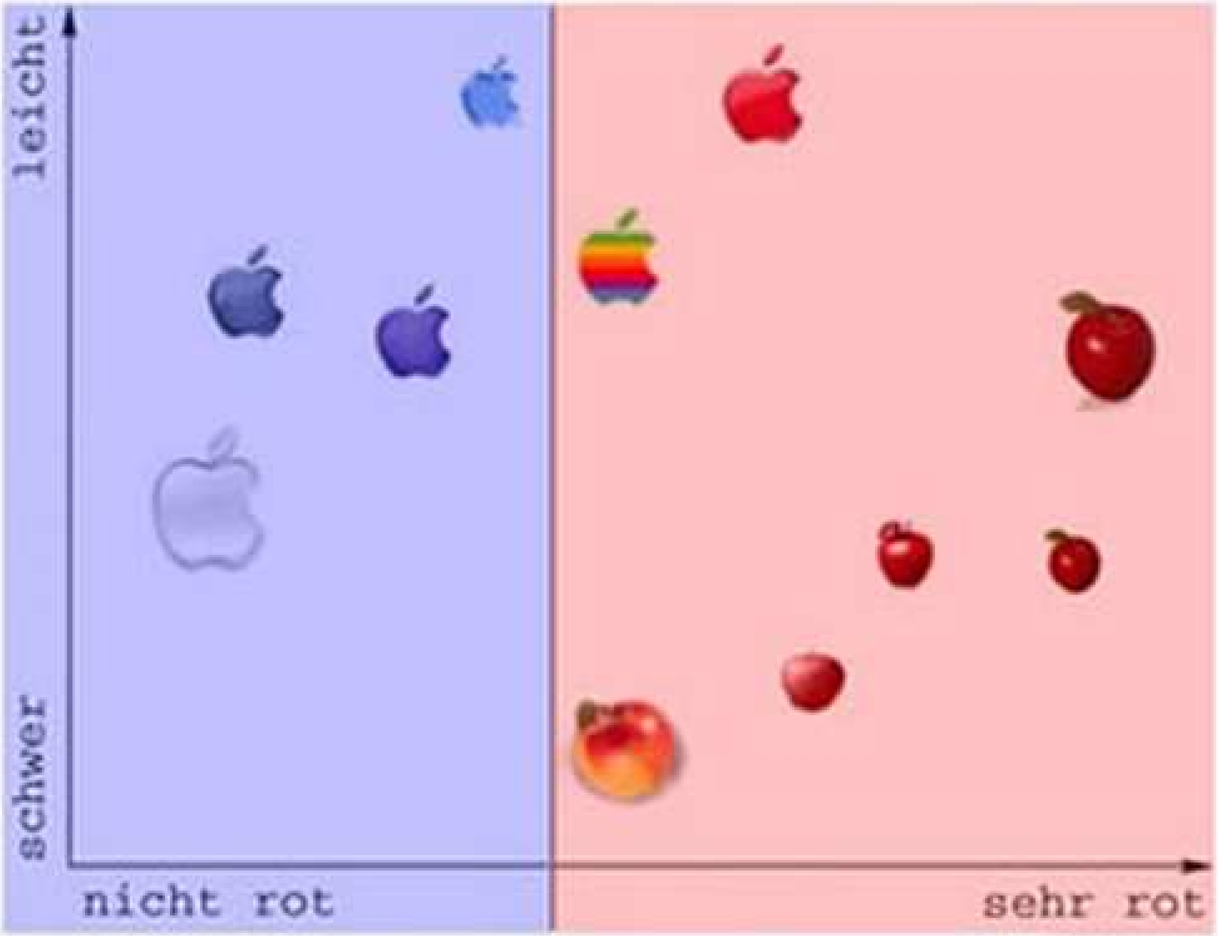


Toy example



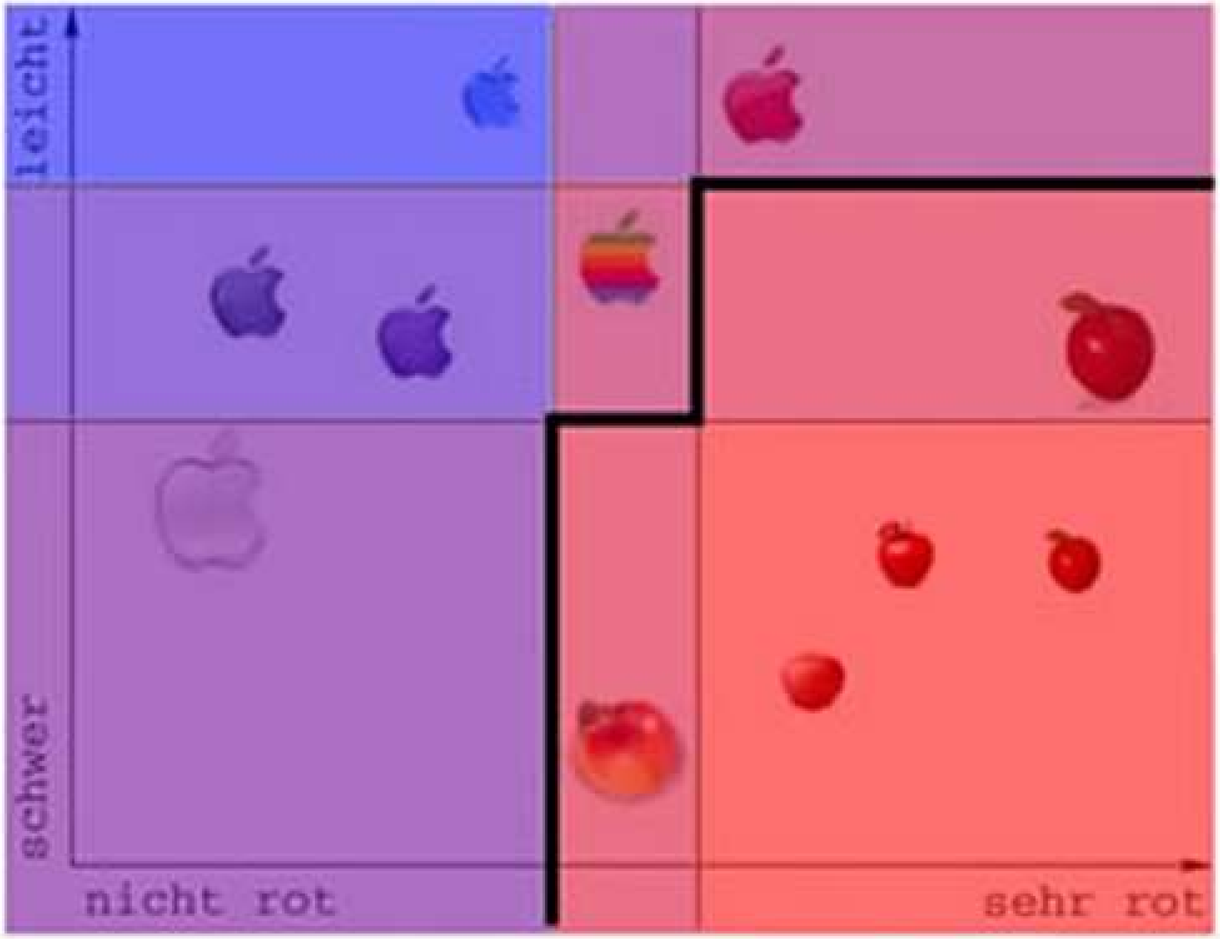
Toy example

Toy example

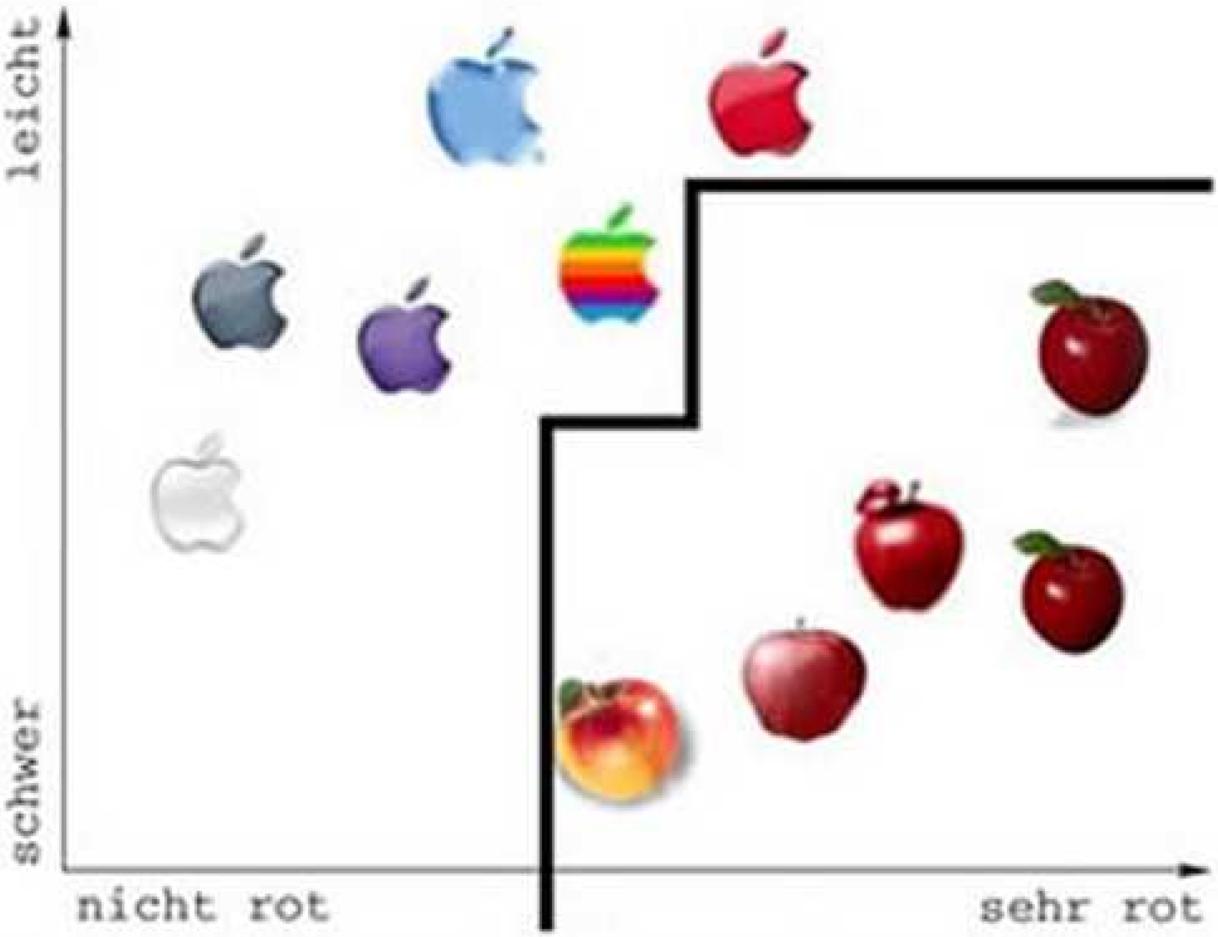


Toy example

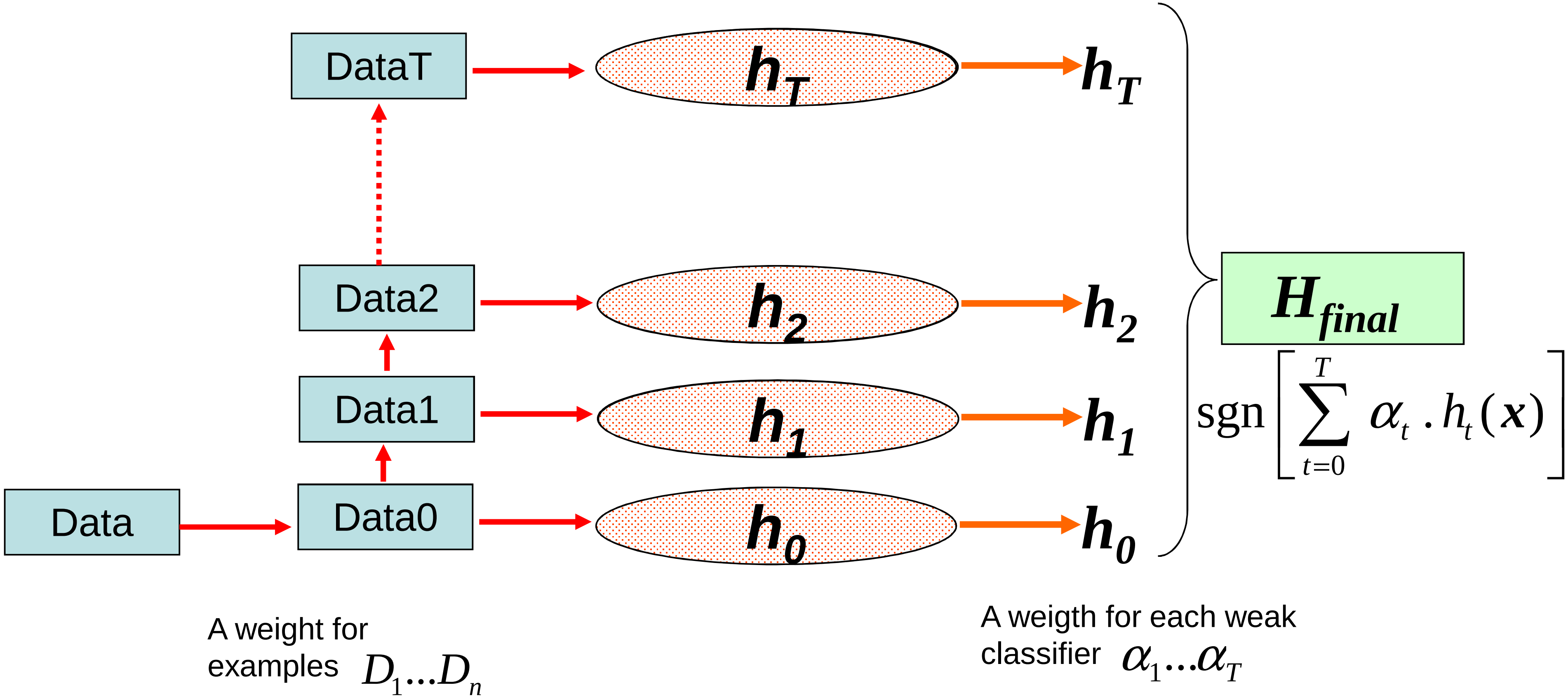
Toy example



Toy example



Principle



DataT
Data2
Data1
Data
Data0
h_T
h_2
h_1
h_0
h_T
h_2
h_1
h_0
H_{final}
$\mathrm{sgn}\left[\sum_{t=0}^{T} \alpha_t \, . \, h_t(\boldsymbol{x})\right]$
A weight for examples $D_1 \ldots D_n$
A weigth for each weak classifier $\alpha_1 \ldots \alpha_T$

Discrete Adaboost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialise $D_1(i) = \frac{1}{m}$.

For $t = 1, \ldots, T$:

- Find the classifier $h_t : X \to \{-1, +1\}$ that minimizes the error with respect to the distribution D_t:

  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$, where $\epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- Prerequisite: $\varepsilon_t < 0.5$, otherwise stop.
- Choose $\alpha_t \in \mathbf{R}$, typically $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$ where ε_t is the weighted error rate of classifier h_t.
- Update:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalisation factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

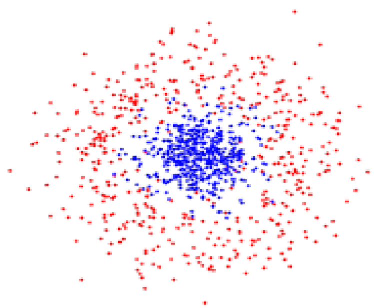
Find the Weak Classifier

Loop step: Call *WeakLearn*, providing it with the distribution D_t;
get back weak classifier $h_t : \mathcal{X} \to \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)
- *WeakLearn* examples:
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ *infinite*
  - Selecting the best one from given *finite* set $\mathcal{H}$

Demonstration example

Training set



Weak classifier = perceptron

$\bullet \sim N(0,1)$ $\quad$ $\bullet \sim \frac{1}{2\pi} e^{-1/2(r-4)^2}$

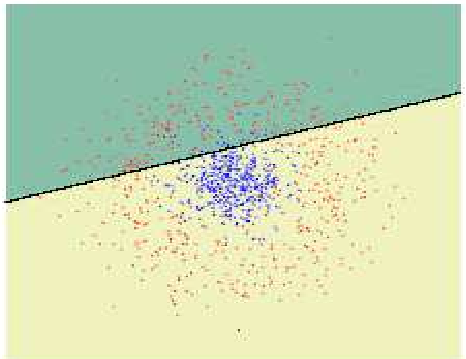
Find the Weak Classifier

Loop step: Call *WeakLearn*, providing it with the distribution D_t;
get back weak classifier $h_t : \mathcal{X} \rightarrow \{-1, 1\}$ from $\mathcal{H} = \{h(x)\}$

- Select a weak classifier with the smallest weighted error
  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- Prerequisite: $\epsilon_t < 1/2$ (otherwise stop)
- *WeakLearn* examples:
  - Decision tree builder, perceptron learning rule – $\mathcal{H}$ *infinite*
  - Selecting the best one from given *finite* set $\mathcal{H}$

Demonstration example

Training set



Weak classifier = perceptron

• $\sim N(0,1)$ • $\sim \frac{1}{2\pi} e^{-1/2(r-4)^2}$

Reweighting

Effect on the training set

Reweighting formula:

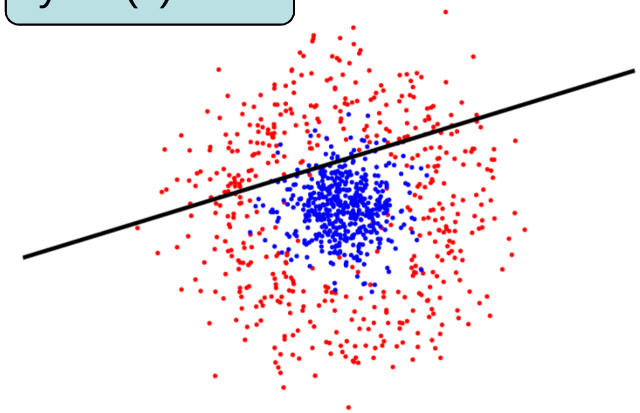
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod^{t} Z}$$

y * h(x) = 1

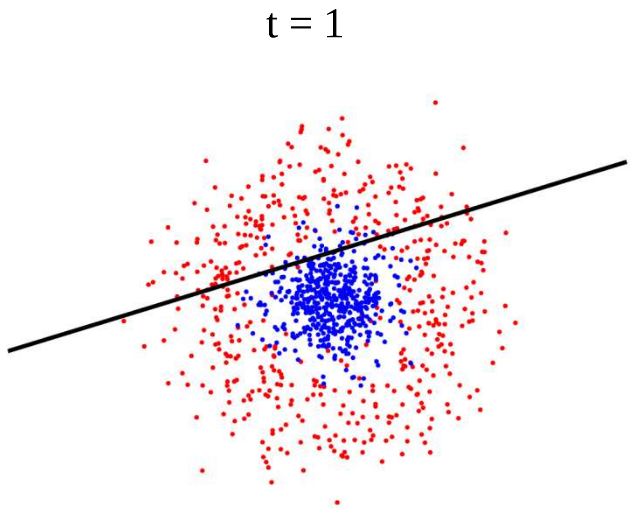
$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

y * h(x) = -1

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples



Algorithm recapitulation



Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

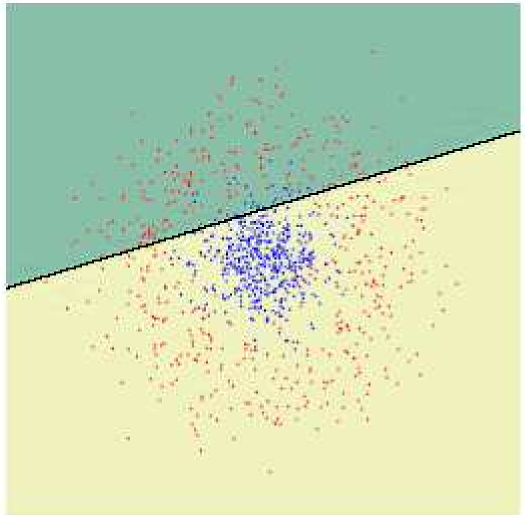
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapitulation

$t = 1$

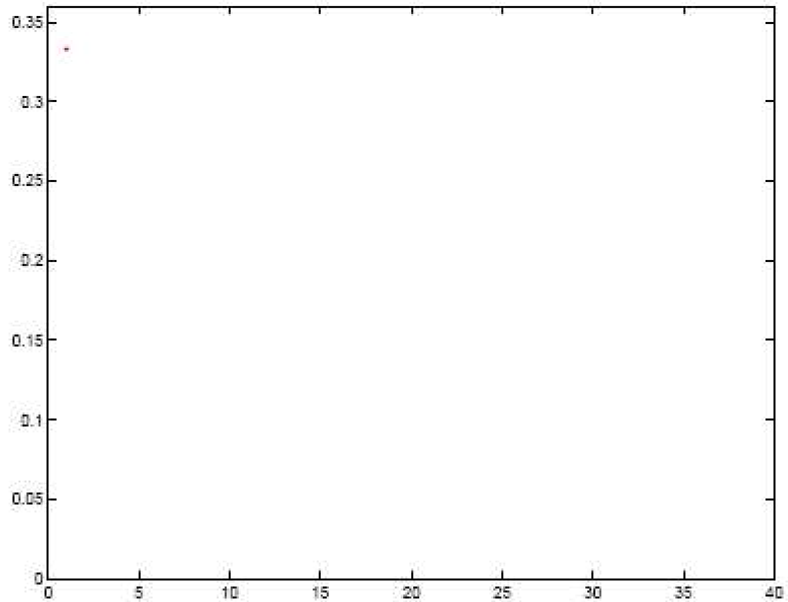


Initialization...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

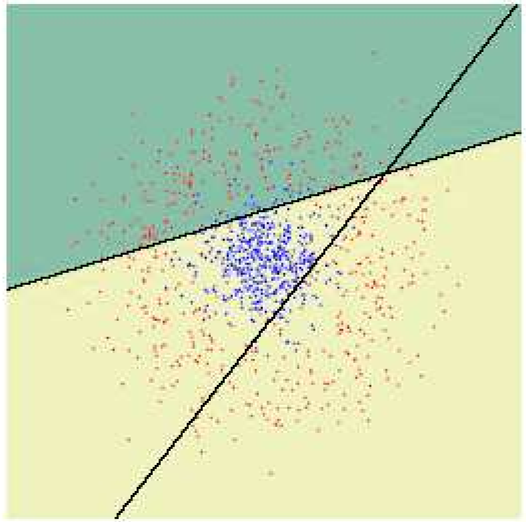


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapitulation

$t = 2$



Initialization...

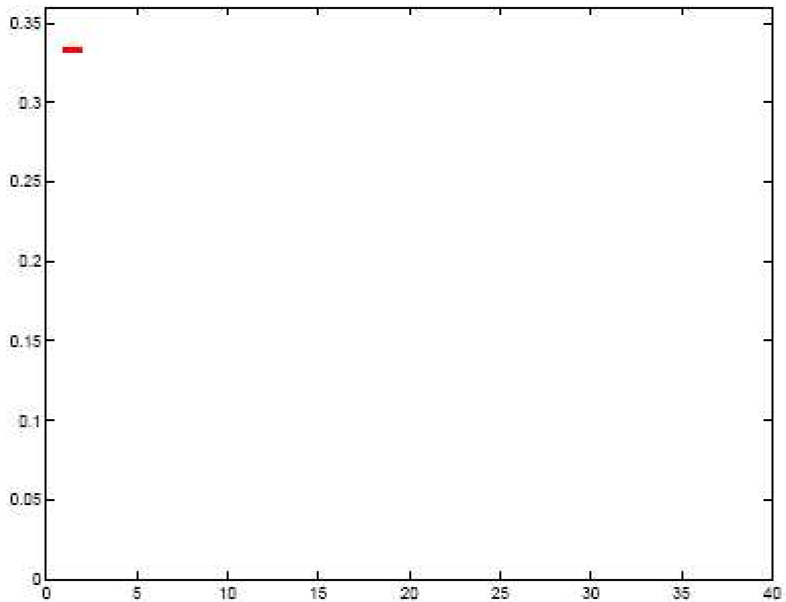
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

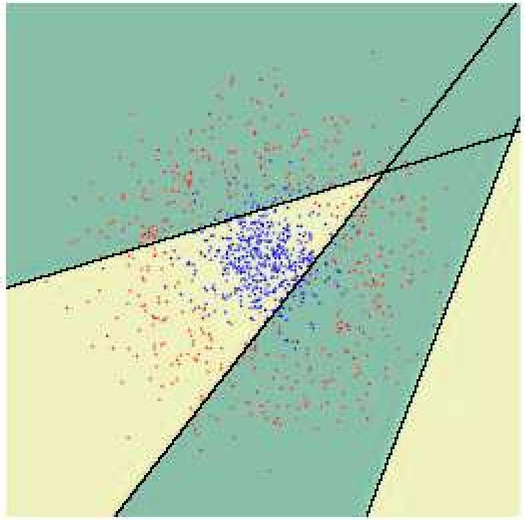
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$



Algorithm recapitulation $_{t\,=\,3}$

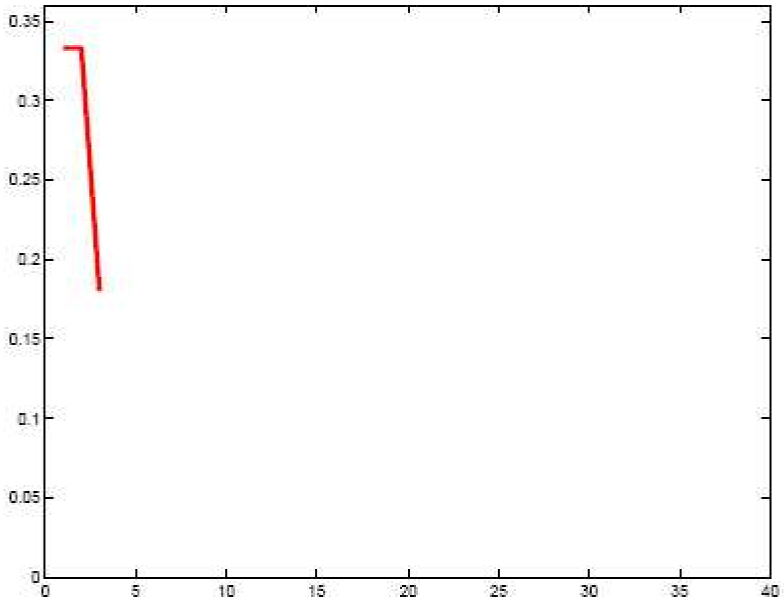


Initialization...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

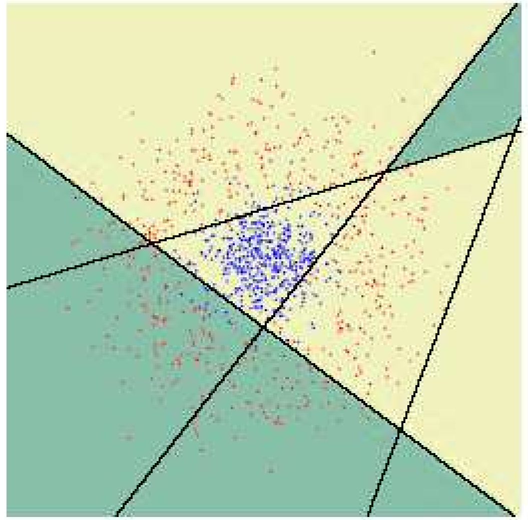


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapitulation

$t = 4$



Initialization...

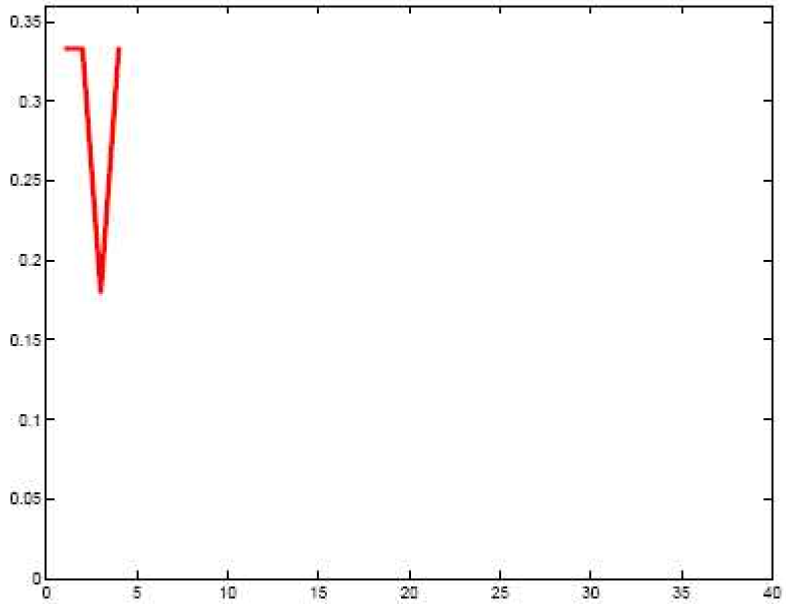
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$



Algorithm recapitulation

Initialization...
For $t = 1, ..., T$:

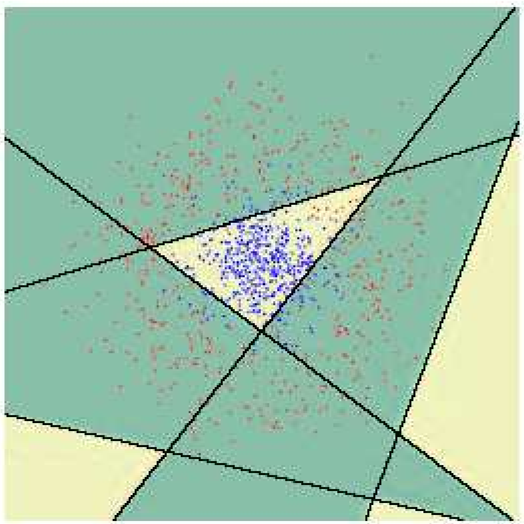
- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

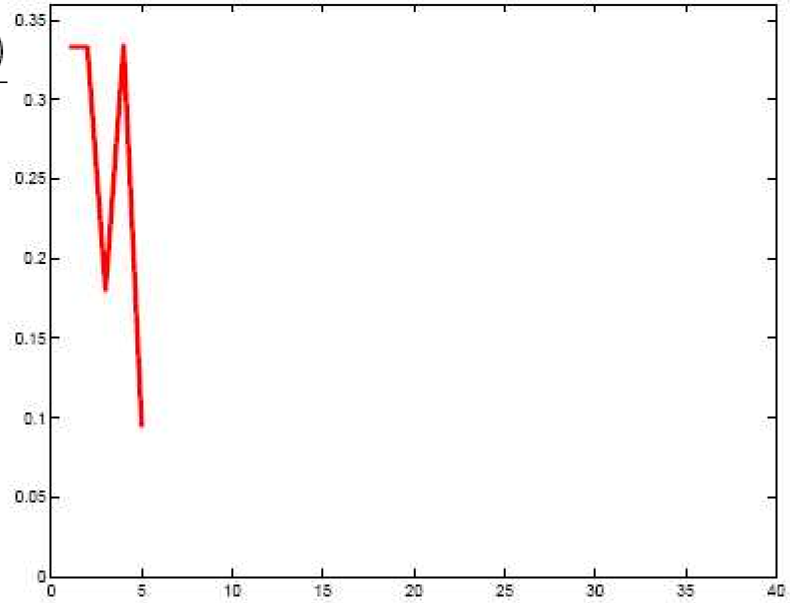
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

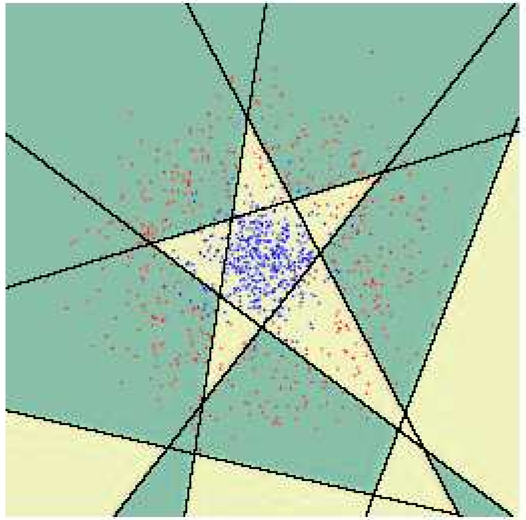
$t = 5$





Algorithm recapitulation

$t = 7$

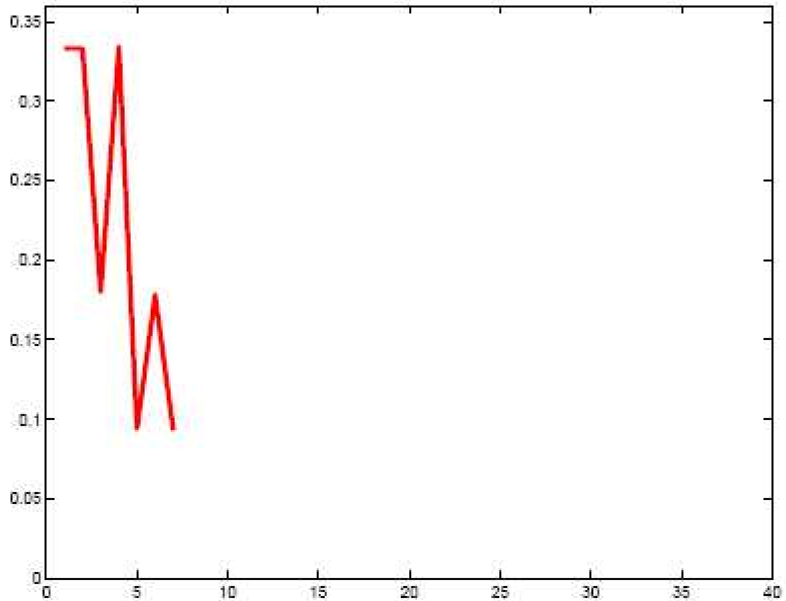


Initialization...

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

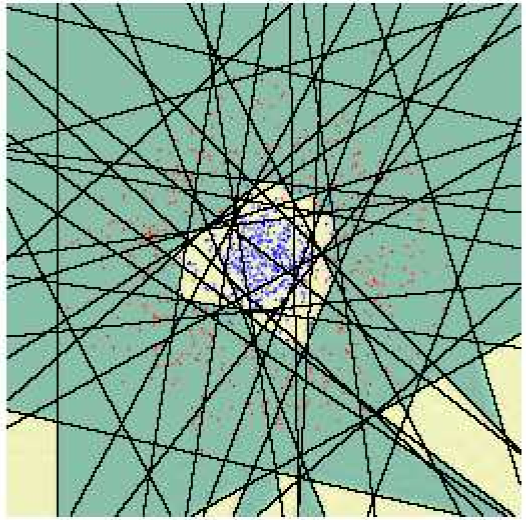


Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm recapitulation

$t = 40$



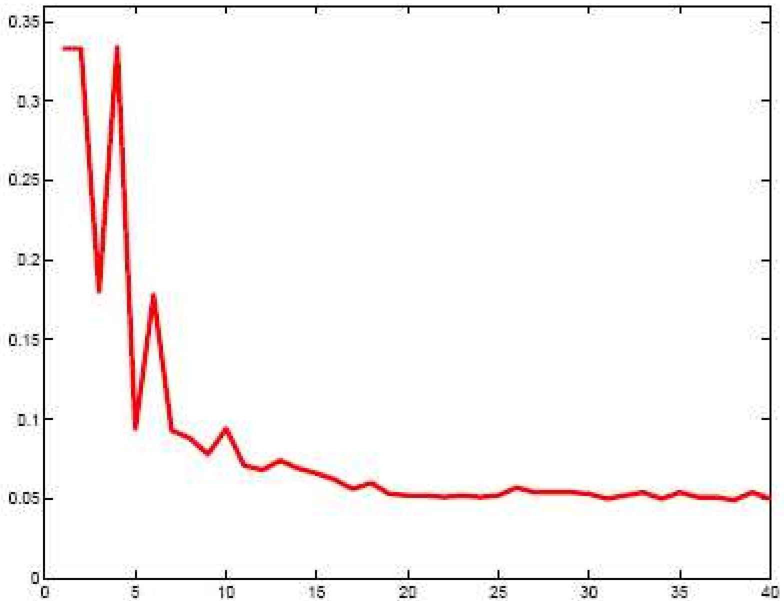
Initialization...
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$



Advantages and Drawbacks

Advantages

- Simple to use and implement
- Good performance in generalization

Drawbacks

- Not optimal solution
- Sensitive to noise
- High computing time

Package R

- Ada: ada()
- Adabag: adaboost.M1()
- Bst: bst()
- Mboost: glmboost()
- wSVM: wsvm()