

Ensemble methods: boosting

Pattern Recognition and Machine Learning - MuMeT 2017

Davide Abati

June 28th, 2017

University of Modena and Reggio Emilia

Boosting exploits the wisdom of the crowd:

- building many weak classifiers is easier than devising one complex prediction rule
- can I build a strong classifier out of a crowd of weak classifiers?

Boosting

Provides a general framework featuring:

- weak classifiers: slightly better than random chance
- cascade: each classifier is aware of previous ones errors
- the prediction rule is based on all classifiers' predictions

Boosting

Provides a general framework featuring:

- weak classifiers: slightly better than random chance
 - How?
- cascade: each classifier is aware of previous ones errors
 - How?
- the prediction rule is based on all classifiers' predictions
 - How?

Adaboost: setting

We are given

- a dataset D of examples $\{\vec{x}_i\}_{i=1}^N \{y_i\}_{i=1}^N$, where $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\} \quad \forall i = 1, \dots, N$
- a predefined number of weak learners K

Adaboost: training algorithm

- initialize a weight w_i for each training example
- repeat for $j = 1, \dots, K$:
 - from D , sample N examples with replacement and build a temporary dataset D_j .
Each example has a probability of being sampled equal to w_i
 - build a weak classifier M_j on D_j
 - compute the error rate ϵ_j of M_j , and the relative *efficiency* α_j
 - update weights for all samples in D_j
 - if correctly classified: decrease the weight
 - if incorrectly classified: increase the weight

Adaboost: a closer look (1)

To build a weak classifier for D_j , repeat until $\epsilon_j < 0.5$

- randomly choose a data dimension
- randomly choose a threshold
- set all points above the threshold (along the chosen axis) to 1 and all points below to -1 (or viceversa)
- compute ϵ_j over D_j

Adaboost: a closer look (2)

To compute the error and the efficiency of a weak classifier M_j

$$\epsilon_j = \sum_{M_j(x_i) \neq y_i} w_i$$

$$\alpha_j = \frac{1}{2} \ln \left(\frac{1 - \epsilon_j}{\epsilon_j} \right)$$

Adaboost: a closer look (3)

To update w_i :

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z^{(j)}} \begin{cases} \exp^{-\alpha_j} & \text{if } M_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } M_j(x_i) \neq y_i \end{cases}$$

$Z^{(j)}$ is a normalization factor that ensures $\sum_i w_i = 1$

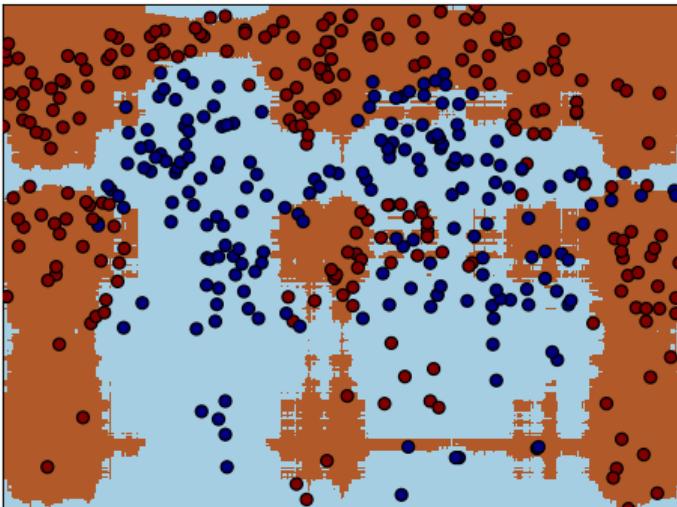
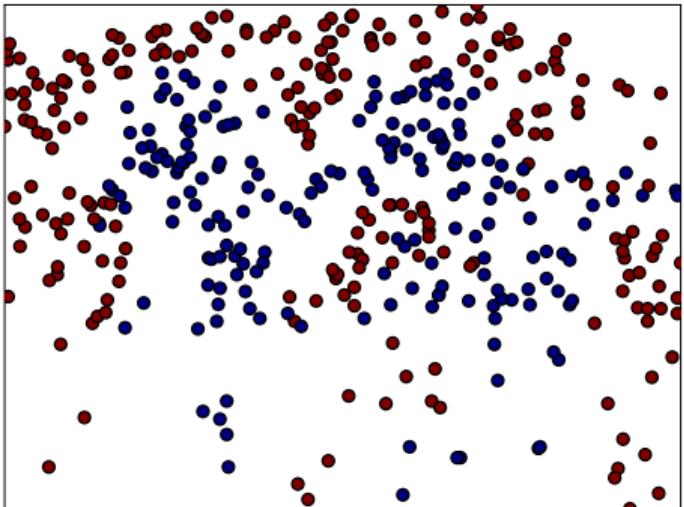
Adaboost: learning

Adaboost: decision rule

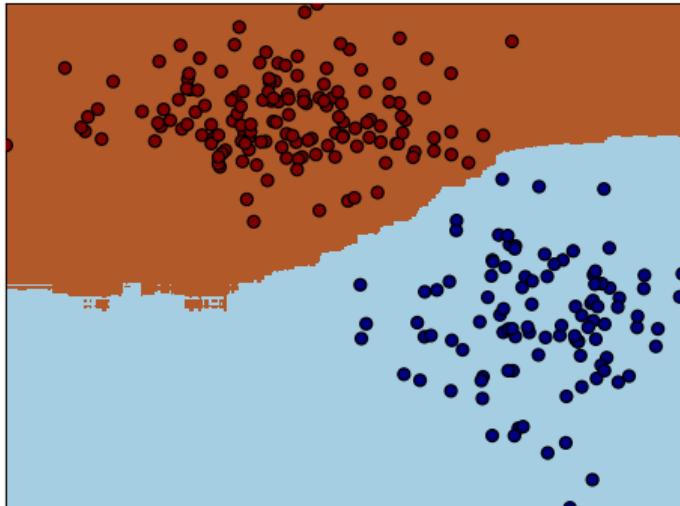
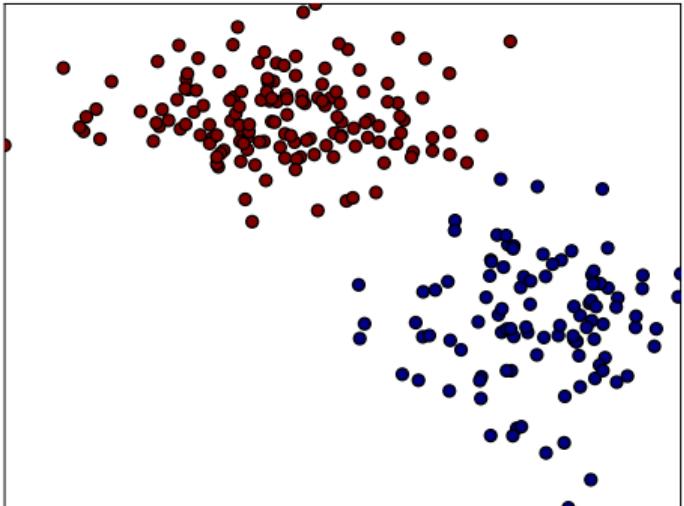
At test phase, you can classify an unknown vector \vec{u}

$$\mathcal{H}(\vec{u}) = \text{sign}\left(\sum_{j=1}^K \alpha_j M_j(\vec{u}) \right)$$

Adaboost: example (1)



Adaboost: example (2)



Adaboost: example (3)

