Boosting

Nonprobabilistic Discriminative Classifiers

Contents

- Boosting
- AdaBoost: Training
- AdaBoost: Multi-class Case
- Probabilities
- Discussion

Boosting: Principle

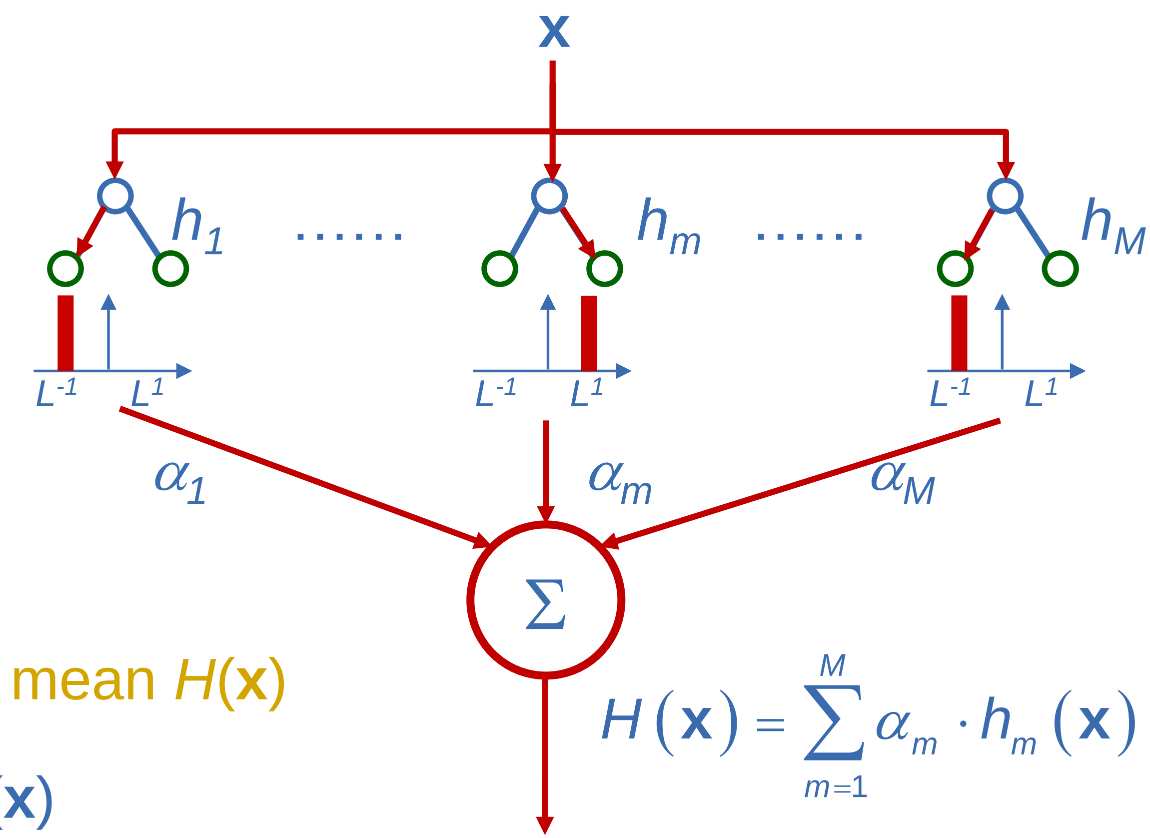
- **Boosting:** Combination of "weak classifiers" $f_b(x)$ to a strong combined classifier
- Prerequisite: The weak classifiers must provide results that are (at least slightly) better than chance (i.e. an error rate < 50% is sufficient in the two-class case)
- Consequently, very simple classifers can be combined
  → Speed!
- There are different variants of boosting
- Here: AdaBoost (Adaptive Boosting)

Weak Classifiers

- Two-class problem: class $C \in \{-1, +1\}$
- "Weak Classifier":
  - A function $h(\mathbf{x})$ that predicts the class C for the feature vector $\mathbf{x}$, i.e. $h(\mathbf{x}) = \pm 1$
  - Success rate must be > 50% ("better than chance")
  - The better the success rate, the faster boosting will be
  - However, with good classifiers, boosting does not make sense
- Examples for weak classifiers :
  - Decision Stumps and other types of trees

Combination of Weak Classifiers

- AdaBoost links M weak classifiers $h_m(\mathbf{x})$
- Each classifier returns a decision for the class C
- Each classifier has a weight α_m that is also determined in training
- Determine the weighted mean $H(\mathbf{x})$
- Class label C: sign of $H(\mathbf{x})$
- $H(\mathbf{x})$ is a strong classifier



$$H(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m \cdot h_m(\mathbf{x})$$

$$C = sign\{H(\mathbf{x})\}$$

Combination of Weak Classifiers

- Difference to bagging: Classifier $h_m(\mathbf{x})$ depends on the classifiers $h_1(\mathbf{x})$, ..., $h_{m-1}(\mathbf{x})$ learned previously(since classifier is adding to the AdaBoost one after one)
- This is achived by considering a weight w_n^m for each training sample $\mathbf{x}_n$ when learning $h_m(\mathbf{x})$
  - w_n^m large: $\mathbf{x}_n$ was classified incorrectly by $h_{m-1}(\mathbf{x})$
  - w_n^m small: $\mathbf{x}_n$ was classified correctly by $h_{m-1}(\mathbf{x})$
- Thus, AdaBoost focuses on "difficult" (incorrectly classified) training samples: if a sample is mis-classified by several weak classifiers in a row, its weight will become larger and larger
- The training error can be made arbitrarily small by increasing M (adding weak classifiers)

AdaBoost: Training

- Given:
  - N feature vectors $\mathbf{x}_n$ with known class labels C_n
  - Number M of the classifiers to be learned
  - Type of classifiers to be learned
- Wanted:
  - M weak classifiers $h_m(\mathbf{x})$
  - Weights α_m for the combination of these classifiers

AdaBoost: Training

1) Initialise the weights: $w_n^1 = 1 / N$
   Iteration 1: “normal“ learning of $h_1(\mathbf{x})$ (identical weights)

2) For $m = 2, \dots M$:
   Starting from iteration 2 the training samples have different weights and therefore different influence on the result

   a) Train the classifier $h_m(\mathbf{x})$ by minimizing the weighted training error J_m:

$$J_m = \sum_{n=1}^{N} w_n^m \cdot \delta\left(h_m\left(\mathbf{x}_n\right) \neq C_n\right) \quad \text{with} \quad \delta(a) = \begin{cases} 1 & \dots \quad a = true \\ 0 & \dots \quad a = false \end{cases}$$

   b) Calculate $\varepsilon_m = \dfrac{J_m}{\sum_{n=1}^{N} w_n^m}$ and weight of h_m: $\alpha_m = \ln\left(\dfrac{1-\varepsilon_m}{\varepsilon_m}\right)$

AdaBoost: Training

(to be continued): The smaller ε_m, the bigger the weight of α_m of $h_m(\mathbf{x})$
→ Classifiers with small training errors have a larger influence on the result

c) New weights: $w_n^{m+1} = w_n^m \cdot \exp\left\{\alpha_m \cdot \delta\left(h_m\left(\mathbf{x}_n\right) \neq C_n\right)\right\}$
Each weak classifier is trained so that it is more likely to correctly classify a training sample having a high weight than a sample having a low weight

d) Normalise the weights w_n^{m+1} such that their sum is 1
The quantity ε_m is the normalized weighted training error J_m of the classifier $h_m(\mathbf{x})$, $\varepsilon_m \in [0, 1]$

AdaBoost: Training

- Update of the weights: $w_n^{m+1} = w_n^m \cdot \exp\{\alpha_m \cdot \delta(h_m(\mathbf{x}_n) \neq C_n)\}$
- Results of the classifier $h_m(\mathbf{x})$ for the training sample $\mathbf{x}_n$:
  - correct → Weight of the sample is not changed: $w_n^{m+1} = w_n^m$
  - not correct → The weight of the sample increases: $w_n^{m+1} = w_n^m \cdot e^{\alpha_m}$
- The training procedure can be derived mathematically by the sequential minimization of the exponential error function (cf. [Bishop, 2006]):

$$E = \sum_{n=1}^{N} \exp\left\{\frac{1}{2} \cdot C_n \cdot f_m(\mathbf{x}_n)\right\} = \sum_{n=1}^{N} \exp\left\{\frac{1}{2} \cdot C_n \cdot \sum_{k=1}^{m} \alpha_l \cdot h_l(\mathbf{x}_n)\right\}$$

AdaBoost: Multi-Class Case

- AdaBoost can be directly applied to the multi-class case
  - Definition of the weak classifiers $h_m(\mathbf{x})$ must be modified so that their output directly becomes the class label C
    → **AdaBoost.M1**
- Problem:
  - The individual classifiers must still be better than 50% (otherwise α_m becomes negative)
  - This is actually much better rather than "slightly better than chance“ ($1 / N_c$ with N_c classes)
- Possible solution 1: Split problem into several two-class problems
  - One against all or one against one (see SVM)

AdaBoost: Multi-Class Case

- Solution 2: SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function)
  - Similiar to AdaBoost.M1
  - Difference: Consider the number of classes N_c for computing the weights in iteration *m:*

$$\alpha_m = \ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right) + \ln[N_c - 1]$$

  - In case $\varepsilon_m < 0.5$, this modification ensures that $\alpha_m > 0$
  - For $N_c = 2$ is this identical to AdaBoost

Probabilities

- AdaBoost does not deliver probabilities

- For the two-class case one can show that

$$C = sign\{H(\mathbf{x})\} = sign\left(\sum_{i=1}^{M} \alpha_m \cdot h_m(\mathbf{x})\right)$$

leads to the following interpretation of $H(\mathbf{x})$:

$$H(\mathbf{x}) = \frac{1}{2} \cdot \ln\left\{\frac{P(C=1|\mathbf{x})}{P(C=-1|\mathbf{x})}\right\}$$

- Therefore, using the logistic Sigmoid function σ

$$P(C=1|\mathbf{x}) = \frac{1}{1+e^{-2 \cdot H(\mathbf{x})}} = \sigma(2 \cdot H(\mathbf{x}))$$

AdaBoost: Discussion

- AdaBoost can improve the quality of the results significantly above those of the weak classifiers
- AdaBoost is regarded as a very good and fast classifier that is easy to apply
- AdaBoost actually describes a family of classifiers (“meta classifier“)
- There are theoretical connections, e.g., to SVM
- AdaBoost is a bit slower than random forests
- AdaBoost can have problems with noisy training data
- Derivation of probabilities is not as clear as with RF