Ensemble Methods

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professor : Christian Gagné

Week 11

UNIVERSITÉ LAVAL

11.1 Ensemble basics

Ensemble methods

- The *no free lunch* theorem
  - No learning algorithm is superior to others for every problem
- Statistical arguments for the use of ensembles
  - Average of a set of samples is more reliable than a single sample value
  - Eliminate variance by averaging on ensemble decisions
  - Removes noise from individual classifier decisions
- Several heads are better than one
  - Voting methods
  - Error-correcting output codes
  - Dynamic sampling of data or features
  - Mixture of experts

Condorcet's jury theorem

- What is the probability that a jury will get a majority decision that is correct?
  - Two possible decisions: correct decision or wrong decision
  - Each jury has a p probability of making a correct decision
  - When the probability $p > 1/2$, the probability of correct jury decision tends to 1 with a very large number of jury participants
  - Conversely, with a probability $p < 1/2$, the probability of a correct jury decision is reduced by increasing the size of the jury.
  - Assumes that the votes are independent and identically distributed (iid)
- Proposed by the Marquis de Condorcet in 1785
  - Mathematical justification of democracy, studied in political science

Ensemble creation approaches

- Different learning algorithms
  - Different assumptions about the data (bias and variance)
- Different hyperparameters
  - Number of hidden neurons/layers
  - Number of neighbours
  - Type of covariance matrix
- Different representations
  - Different measures/sensors
  - Different features (random forest, *random subspaces*)
- Different training datasets
  - Random sampling of data (*bagging*)
  - Sampling according to misclassified data (*boosting*)

Complexity, combination, formalization

- Complexity of basic classifiers
  - Basic classifiers don't have to be very precise individually
  - Simplicity is often better than performance
  - Diversity in classification, specialization in certain fields
  - If the classifier errors are independent and identically distributed (iid)

$$\lim_{L \to \infty} E_{ensemble} \to E_{Bayes}$$

- Approaches for combinations
  - Multiple expert combinations (parallel)
    - Votes, mixture of experts, *stacked generalization*
  - Multi-stage combinations (series)
    - Next stage classifiers called only when in doubt at previous stages (cascade classifiers)
- Formalization of ensemble methods

$$\bar{\mathrm{h}}(\mathbf{x}|\Phi) = \mathrm{f}(\mathrm{h}_1(\mathbf{x}), \mathrm{h}_2(\mathbf{x}), \ldots, \mathrm{h}_L(\mathbf{x})|\Phi)$$

11.2 Votes and Bayesian combination

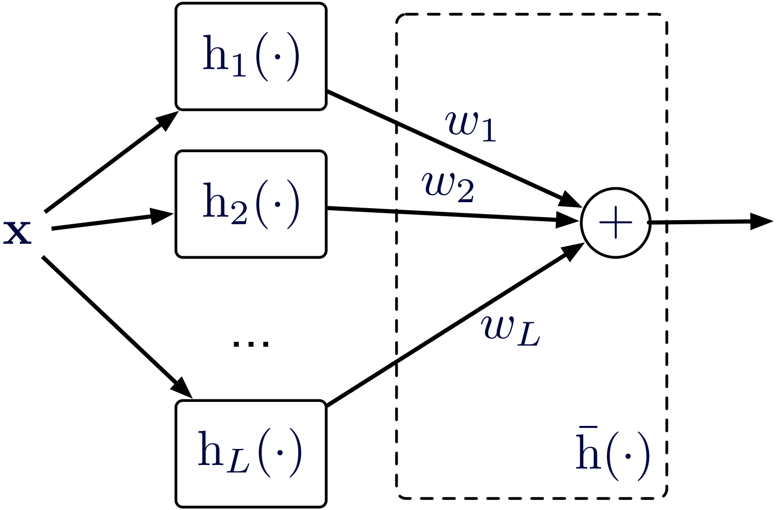
Votes

- Voting method
  - Assign to the most frequent class among the responses of the basic classifiers
- General formulation: weight each vote by a factor w_j

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} w_j \mathrm{h}_j(\mathbf{x}), \quad \text{where } w_j \geq 0, \forall j \text{ and } \sum_j w_j = 1$$

  - Linear model of parallel combination
  - In the case of simple voting, $w_j = 1/L$
  - Weights can represent the confidence in each classifier

Votes



$\mathbf{x}$
$\mathrm{h}_1(\cdot)$
$\mathrm{h}_2(\cdot)$
...
$\mathrm{h}_L(\cdot)$
w_1
w_2
w_L
$+$
$\bar{\mathrm{h}}(\cdot)$

Combination of Bayesian models

- Bayesian combination model

$$P(C_i|\mathbf{x}) = \sum_{\forall \mathcal{M}_j} P(C_i|\mathbf{x},\mathcal{M}_j)P(\mathcal{M}_j)$$

  - $w_j = P(\mathcal{M}_j)$ and $\mathrm{h}_j(\mathbf{x}) = P(C_i|\mathbf{x},\mathcal{M}_j)$
  - Simple voting is the case of *a priori* equal probabilities, $P(\mathcal{M}_j) = 1/L$

Bias and variance

- Bias and variance in two-class classifier ensembles
  - h_j are iid, with expectation $\mathbb{E}[\mathrm{h}_j]$ and variance $\mathrm{Var}(\mathrm{h}_j)$

$$
\begin{aligned}
\mathbb{E}[\bar{\mathrm{h}}] &= \mathbb{E}\left[\sum_{j=1}^{L} \frac{1}{L}\mathrm{h}_j\right] = \frac{1}{L} L\, \mathbb{E}[\mathrm{h}_j] = \mathbb{E}[\mathrm{h}_j] \\
\mathrm{Var}(\bar{\mathrm{h}}) &= \mathrm{Var}\left(\sum_{j=1}^{L} \frac{1}{L}\mathrm{h}_j\right) = \frac{1}{L^2} L\, \mathrm{Var}(\mathrm{h}_j) = \frac{1}{L}\mathrm{Var}(\mathrm{h}_j)
\end{aligned}
$$

- Variance decreases as the number of independent voters L increases
  - With ensembles, we can therefore reduce the variance without affecting the bias.
  - Quadratic error is also reduced

Diversity and negative correlation

- Variance of ensembles, general case

$$\mathrm{Var}(\bar{\mathrm{h}}) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j \mathrm{h}_j\right) = \frac{1}{L^2}\left[\sum_j \mathrm{Var}\left(\mathrm{h}_j\right) + 2\sum_j \sum_{i>j} \mathrm{Cov}(\mathrm{h}_j,\mathrm{h}_i)\right]$$

  - Further variance reduction with negatively correlated voters
  - Quadratic error can be reduced, provided the negative correlation does not affect the bias of the set
- Diversity in the answers of the classifiers in the ensemble
  - Goal in the overall training: to obtain classifiers that do not make the same mistakes.
  - Borderline case without diversity: L copies of the same classifier

11.3 Decision matrices and error-correcting output codes

Decision matrix

- Multi-class classification with ensembles, with weighted vote

$$\bar{\mathrm{h}}_i(\mathbf{x}) = \sum_j w_{i,j} \mathrm{h}_{j,i}(\mathbf{x})$$

- Decision matrix $\mathbf{W}$: weight values $w_{i,j}$
- Decision matrix for a one-against-all classification (example with $L = K = 4$)

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

- Ambiguity when a basic classifier makes a bad decision
  - Two values $\bar{\mathrm{h}}_i(\mathbf{x}) = 0$
  - Too high similarity between codes (low Hamming distance)

Ensembles with redundancy

- Decision matrix for one-versus-one decisions (example with $K = 4$, $L = K(K-1)/2 = 6$)

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

  - Value of $w_{i,j} = 0$ means the decision is ignored
  - Error in a basic classifier does not necessarily imply ambiguity
  - Value L grows quadratically with K
- Generalization of the approach: error-correcting output codes
  - Use a decision matrix $\mathbf{W}$ of preset size L.
  - Hamming distance between lines is maximized

Error-correcting output codes

- Error-correcting Output Codes (ECOC)
  - With K classes, there are $2^{(K-1)} - 1$ problems with two different classes
  - Diversity of discriminants: different columns
  - Error correction: different components for a line
- Example of matrix with ECOC ($K = 4$ and $L = 9$)

$$\mathbf{W} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 \end{bmatrix}$$

  - Minimum difference (Hamming distance) of $d = 5$ between each pair of lines
    - Therefore tolerates up to $\left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{5-1}{2} \right\rfloor = 2$ basic classifier errors
  - Choice of the class according to $\bar{\mathrm{h}}_i(\mathbf{x})$ maximum
  - Value $\bar{\mathrm{h}}_i(\mathbf{x})$ normalized in $[0, 1]$ can be interpreted as a probability
  - Choice of values $\mathbf{W}$ partly arbitrary, some dichotomies may be more difficult than others

11.4 Bagging, random subspaces and random forests

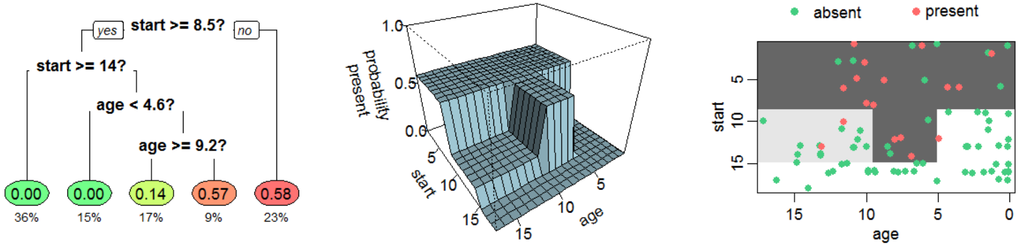
Bagging and random subspaces

- *Bagging*: ensemble of classifiers trained on slightly different datasets
  - Each basic classifier trained on dataset $\mathcal{X}_j$
    - $\mathcal{X}_j$: draw **with replacement** of N data in $\mathcal{X}$
    - Replacement: several copies of some data, absence of some others
  - Ideally, basic classifiers should be unstable
    - Unstable training algorithm: for slightly different datasets, gives classifiers with different behaviors
    - Stable: k-nearest neighbours, parametric classification
    - Unstable: multilayer perceptron, Hart condensation
    - In general, unstable algorithms have a large variance
- *Random subspaces*
  - Generate each basic classifier by random sampling of a subset of features

Decision trees

- Decision trees
  - Hierarchical (recursive) separation of the input space
  - Each node of the tree is a test on a value with discrete outcomes
  - Performs a finer and finer division of the input space
- Decision tree properties
  - Top-down construction of trees according to performance criteria (ex. entropy)
  - Pruning reduces over-specialization
  - Useful to extract interpretable decision rules

Decision trees



By Stephen Milborrow, CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Cart_tree_kyphosis.png`.

Random forest

- Problem with decision trees for classification
  - Classifiers with low bias and high variance
  - Which implies high risk of overfitting (even if pruning is used)
- Solution: make an ensemble of trees
  - Averaging keeps bias low while reducing overall variance
  - But the ensemble must include a good diversity of trees
- Generate "randomized" trees with bagging and random subspaces
  - To learn each node, use different subsets of data and variables
- Ensemble of random trees corresponds to a random forest
  - Averaging tree decisions
  - Variance on decisions is a good indicator of overall confidence

11.5 Boosting

Boosting

- *Bagging*: requires unstable algorithms
  - Passively generated diversity
- *Boosting*: actively generate new classifiers from data that is difficult for existing classifiers to handle
  1. Randomly divide the dataset into three subsets ($\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$)
  2. Train Classifier h_1 on $\mathcal{X}_1$
  3. Evaluate data $\mathcal{X}_2$ with h_1, use misclassified data and an equal number of well-classified data to form $\mathcal{X}_2'$
  4. Train classifier h_2 on $\mathcal{X}_2'$
  5. Evaluate data $\mathcal{X}_3$ with h_1 and h_2, use data where h_1 and h_2 disagree to form $\mathcal{X}_3'$
  6. Train classifier h_3 on $\mathcal{X}_3'$
- Evaluate data classification: test data with h_1 and h_2, if they disagree, use decision of h_3
- Improves performance, but requires very large datasets

AdaBoost

- AdaBoost (*adaptive boosting*): reuse the same dataset for basic classifiers
  - Unlike classic *boosting*, does not require very large datasets
  - Can generate an arbitrarily high number of classifiers
- AdaBoost.M1: the probability of sampling a data changes according to the errors of the basic classifiers
  - Initially, $p_1^t = 1/N$, $t = 1, \dots, N$
  - Sample dataset $\mathcal{X}_j$ from $\mathcal{X}$ according to probabilities p_j^t
  - Train classifier h_j with $\mathcal{X}_j$
  - If error rate of h_j is higher than $\epsilon_j > 0.5$, interrupt the algorithm, $\epsilon_j = \sum_t p_j^t \ell_{0-1}(r^t, \mathrm{h}_j(\mathbf{x}^t))$
  - Calculate the probabilities p_{j+1}^t according to the classification of $\mathcal{X}$ with h_j
  - Repeat to generate the L basic classifiers

Weak learner

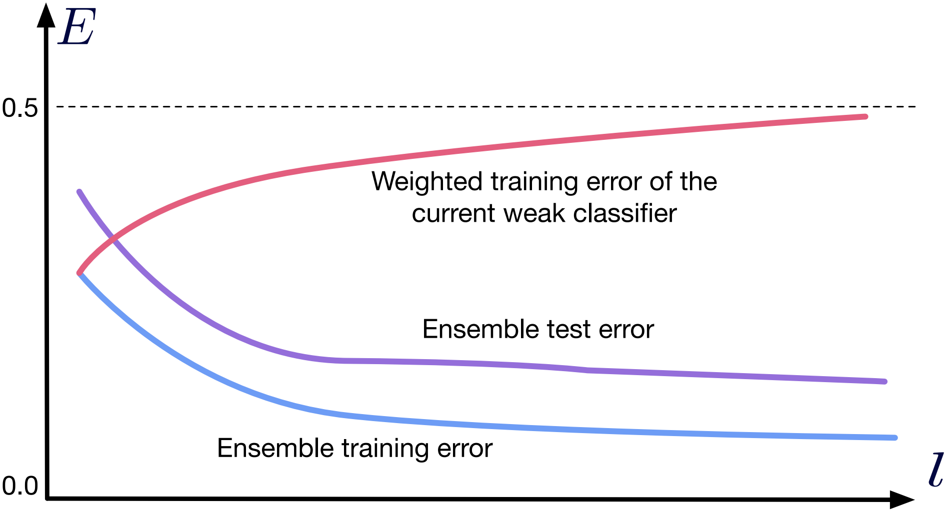
- *Boosting* and AdaBoost do not require very precise classifiers
  - *Weak learner*: algorithm with an error probability of less than 1/2 in two classes (better than random classification) and relatively unstable (sustained variations in classification)
  - Using *weak learners* allows a good diversity for the classification
- Decision stumps: *weak learner* commonly used with AdaBoost
  - Decisions based on a threshold applied to a single dimension

$$\mathrm{h}(\mathbf{x}|\theta,\upsilon,\gamma) = \mathrm{sgn}(\theta(x_\gamma - \upsilon)), \quad \theta \in \{-1,1\},\, \gamma \in \{1,\ldots,D\},\, \upsilon \in \mathbb{R}$$

  - Deterministic training of decision stumps

$$\begin{aligned}
\tilde{x}_j^k &= x_j^t \mid \tilde{x}_j^1 \leq \tilde{x}_j^2 \leq \cdots \leq \tilde{x}_j^{k-1} \leq x_j^t \leq \tilde{x}_k^{k+1} \leq \cdots \leq \tilde{x}_j^N \\
v_j^k &= 0.5(\tilde{x}_j^k + \tilde{x}_j^{k+1}),\ k = 1,\ldots,N-1 \\
\mathcal{A}_j &= \left\{(s_j, v_j^k, j) \mid \forall s_j \in \{-1,1\},\ \forall k \in \{1,\ldots,N-1\}\right\} \\
\mathcal{A} &= \mathcal{A}_1 + \mathcal{A}_2 + \cdots + \mathcal{A}_D \\
(\theta,\upsilon,\gamma) &= \operatorname*{argmin}_{(s_j,u_j^k,j)\in\mathcal{A}} E(\mathrm{h}(\cdot|s_j,u_j^k,j)|\mathcal{X})
\end{aligned}$$

Errors with AdaBoost



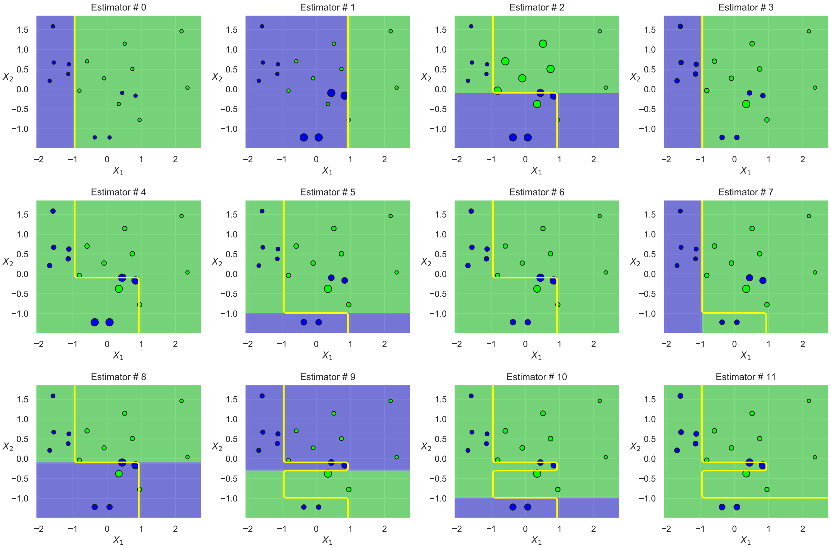
AdaBoost algorithm

1. Initialize the probabilities of each data, $p_1^t = 1/N$, $t = 1, \ldots, N$
2. For each basic classifier $j = 1, \ldots, L$:
   2.1 Sample dataset $\mathcal{X}_j$ from $\mathcal{X}$ according to probabilities p_j^t
   2.2 Train classifier h_j with dataset $\mathcal{X}_j$
   2.3 Calculate the error of the classifier, $\epsilon_j = \sum_t p_j^t \ell_{0-1}(r^t, \mathrm{h}_j(\mathbf{x}^t))$
   2.4 If error $\epsilon_j > 0.5$, then $L = j - 1$ and stop the algorithm
   2.5 Calculate $\beta_j = \frac{\epsilon_j}{1-\epsilon_j}$
   2.6 Calculate the new probabilities p_{j+1}^t

$$p_{j+1}^t = \frac{q_j^t}{\sum_s q_j^s}, \quad q_j^t = \begin{cases} \beta_j p_j^t & \text{if } \mathrm{h}_j(\mathbf{x}^t) = r^t \\ p_j^t & \text{otherwise} \end{cases}, \quad t = 1, \ldots, N$$

Evaluating the classification of a data: $\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) \mathrm{h}_j(\mathbf{x})$

Example with AdaBoost



Maximizing margins with AdaBoost

- AdaBoost maximizes margins for the classification
  - Learning with higher probabilities for difficult-to-classify data
    - Difficult data: data in the margin
  - $\bar{\mathrm{h}}_i$ is the result of a weighted vote

$$\bar{\mathrm{h}}_i = \frac{\text{votes for class } i - \text{votes against class } i}{\text{total number of votes}}$$

  - With many classifiers, $\bar{\mathrm{h}}_i(\mathbf{x}) \rightarrow 1$ if $\mathbf{x} \in C_i$ and $\bar{\mathrm{h}}_i(\mathbf{x}) \rightarrow -1$ otherwise
  - Wide margins $\Rightarrow$ better generalization
- Many variants of *boosting*
  - LPBoost: learning $\alpha_j = \log \frac{1}{\beta_j}$ by linear programming
    - At each generation of basic classifier, relearns the α_j of all current classifiers
    - Many parallels to be made with SVMs

11.6 Other combination models

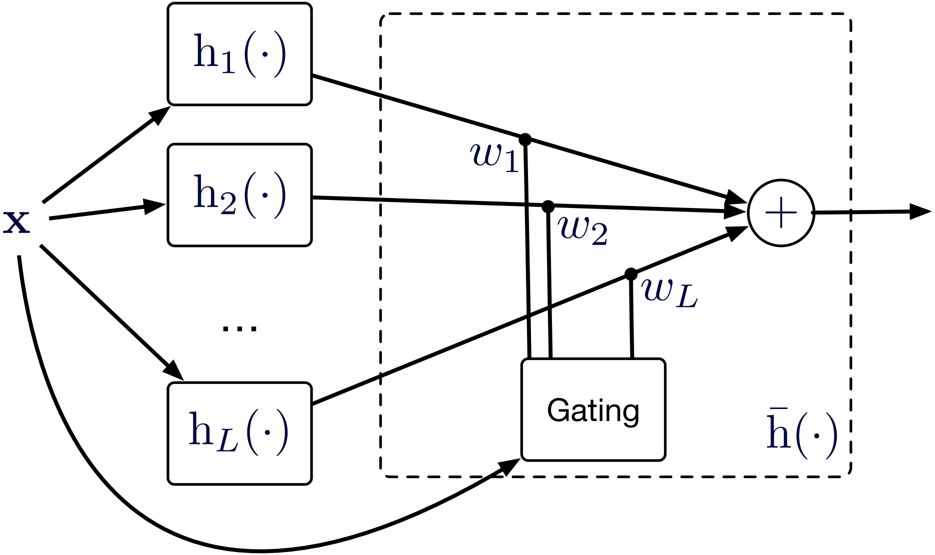
Mixture of experts

- Mixture of experts
  - Experts-classifiers specialized on certain aspects of the problem
  - Work in parallel, with routing function weighting decisions according to expertise
  - Similar to weighted voting, but with non-constant weighting

$$\bar{\mathrm{h}}(\mathbf{x}) = \sum_{j=1}^{L} w_j(\mathbf{x}) \mathrm{h}_j(\mathbf{x})$$

- Specialization in different regions of reduced space correlation
- Thus generates biased but negatively correlated experts
  - Implies an overall reduction of the variance, and thus of the error
- Routing function can be non-linear (e.g. multilayer perceptron)
  - May reduce bias, at the risk of increasing variance (overfitting)

Mixtures of experts



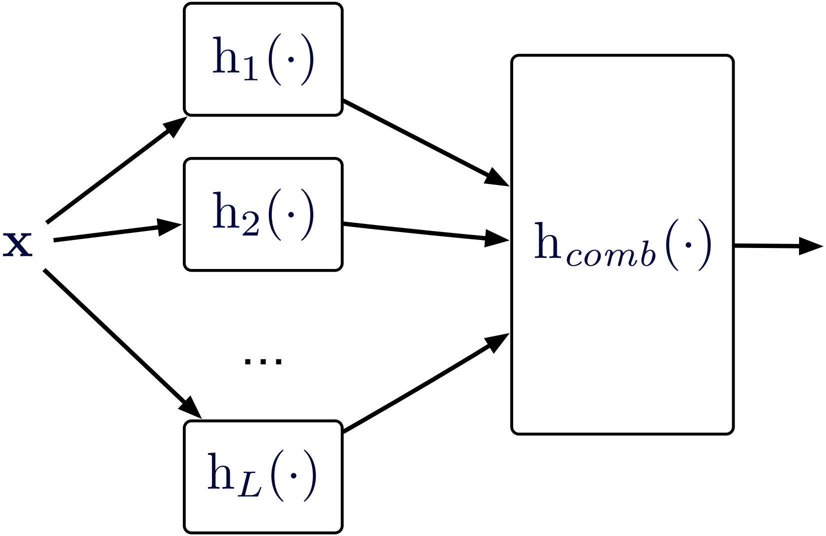
Stacked generalization

- *Stacked generalization*: two-stage system
    - First stage: basic classifiers working in parallel
    - Second stage: combination system associating the output of the basic classifiers with the desired label

$$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_{comb}(\mathrm{h}_1(\mathbf{x}),\mathrm{h}_2(\mathbf{x}),\ldots,\mathrm{h}_L(\mathbf{x}))$$

- Combination system: standard classifier
    - Learn how basic classifiers make mistakes
    - Training of the combination system must be done on data not seen by the basic classifiers
    - Allows to estimate and correct the biases of the basic classifiers

Stacked generalization

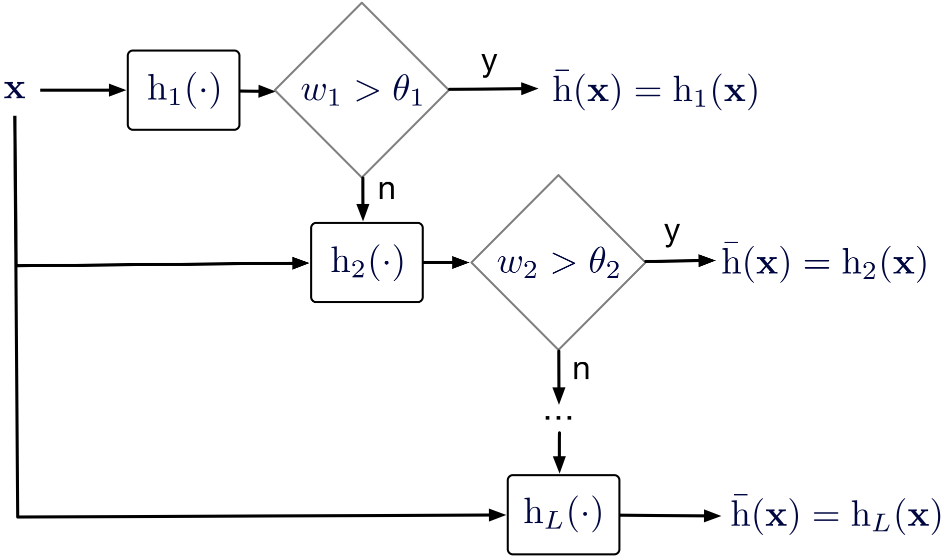


Cascading classifiers

- Cascading classifiers: sequence of basic classifiers
  - Moving from one stage to another if the classifier k has a low confidence in its classification, $w_j(\mathbf{x}) < \theta_j$

$$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_j(\mathbf{x}) \quad \text{if } w_j(\mathbf{x}) \geq \theta_j \text{ and } w_k(\mathbf{x}) < \theta_k,\ \forall k < j$$

  - Confidence $w_j(\mathbf{x})$ can correspond to the *a posteriori* probability $P(C_i|\mathbf{x})$ of the classifier
  - Threshold on confidence θ_j should be high (high rejection rate) for first stages
- Training of the cascade
  - Classifier h_1 trained with $\mathcal{X}_1 = \mathcal{X}$
  - Dataset $\mathcal{X}_{j+1}$ is formed from the rejects of $\mathcal{X}_j$ with classifier h_j
  - Classifier h_{j+1} trained with dataset $\mathcal{X}_{j+1}$
- Basic classifiers of increasing complexity
  - Simple (inexpensive) classifiers handle most cases
  - Complex (expensive) classifiers on the top stages handle difficult cases


$\mathbf{x}$
$\mathrm{h}_1(\cdot)$
$w_1 > \theta_1$
y
$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_1(\mathbf{x})$
n
$\mathrm{h}_2(\cdot)$
$w_2 > \theta_2$
y
$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_2(\mathbf{x})$
n
...
$\mathrm{h}_L(\cdot)$
$\bar{\mathrm{h}}(\mathbf{x}) = \mathrm{h}_L(\mathbf{x})$

Overproduction and selection

- $\bar{\mathrm{h}}(\mathbf{x}|\Phi) = f(\mathrm{h}_1(\mathbf{x}), \mathrm{h}_2(\mathbf{x}), \ldots, \mathrm{h}_L(\mathbf{x})|\Phi)$: meta-classifier
  - Each classifier $\mathrm{h}_i(\mathbf{x})$ can be seen as a feature (or a basic function) of the meta-classifier
- Overproduction and selection
  - Generate a wide variety of candidate classifiers
    - E.g. *random subspaces* method
  - Select a subset of these classifiers to form the final ensemble
- Possible selection by feature selection methods
  - Sequential forward selection
  - Sequential backward selection
  - Multiobjective evolutionary algorithms

11.7 Ensembles in scikit-learn

Scikit-learn

- `ensemble.BaggingClassifier`: several variants of *Bagging* classifiers, including *random subspaces*
- `ensemble.RandomForestClassifier`: random forest for classification
- `ensemble.AdaBoostClassifier`: AdaBoost.SAMME variants of the AdaBoost algorithm
- `ensemble.VotingClassifier`: vote of classifiers, including majority vote and probability weighted summation
- `multiclass.OutputCodeClassifier`: combination of classifiers with a decision code, which can be an error-correcting output codes