Ensemble Learning

Machine Learning, A.Y. 2022/23, Padova

Fabio Aiolli

December 7th, 2022

Ensemble methods

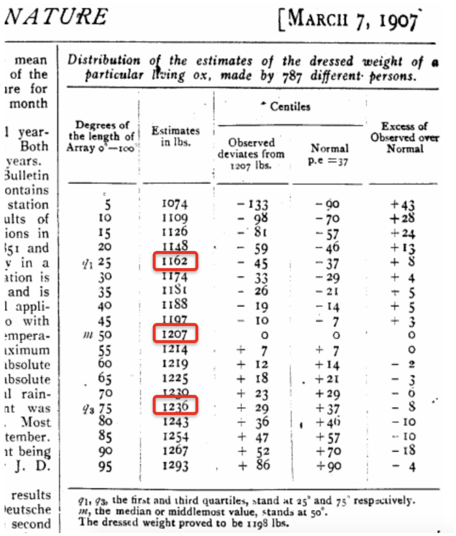
- **General IDEA**: get predictions from multiple models (ensemble) and aggregate the predictions;
- **Classification**: an ensemble of classifiers (base/weak learners) is a set of classifiers whose individual decisions are combined in some way to classify new examples;

Example of an ensemble method

Guess the weight of the cow

- Competition held in England in 1906;
- 787 participants;
- Correct answer: 1198 lb ($\approx$ 543 kg);
- Sir. Francis Galton recorded the results and published in **Nature**.

Guess the weight of the cow - Results



NATURE [MARCH 7, 1907

mean of the ıre for month

1 year- Both years. 3ulletin ontains station ults of ions in 351 and y in a ation is and is l appli- o with empera- ıximum ıbsolute ıbsolute ıl rain- nt was . Most tember. ıt being J. D.

results)eutsche second

Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles		Excess of Observed over Normal
		Observed deviates from 1207 lbs.	Normal p.e =37	
5	1074	−133	−90	+43
10	1109	− 98	−70	+28
15	1126	− 81	−57	+24
20	1148	− 59	−46	+13
q_1 25	1162	− 45	−37	+ 8
30	1174	− 33	−29	+ 4
35	1181	− 26	−21	+ 5
40	1188	− 19	−14	+ 5
45	1197	− 10	− 7	+ 3
m 50	1207	0	0	0
55	1214	+ 7	+ 7	0
60	1219	+ 12	+14	− 2
65	1225	+ 18	+21	− 3
70	1230	+ 23	+29	− 6
q_3 75	1236	+ 29	+37	− 8
80	1243	+ 36	+46	−10
85	1254	+ 47	+57	−10
90	1267	+ 52	+70	−18
95	1293	+ 86	+90	− 4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

- Percentiles:
  - 25th: 1162 lb;
  - 50th: 1207 lb;
  - 75th: 1236 lb.
- Mean: 1197 lb ⇒ correct answer: 1198 lb!!
- Ensemble method: average of predictors.

Justification

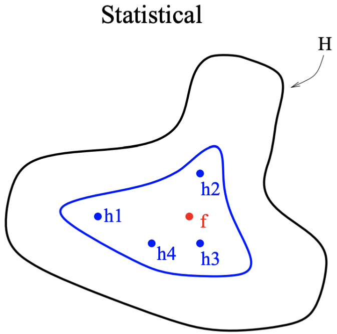
Question: *why (and when) a combination of classifiers is justified?*

We will try to answer this question in two ways:

- *Intuitively*: following the Dietterich's "3 reasons why":
  - Statistical;
  - Computational;
  - Representational.
- *Theoretically*: bias-variance trade-off.

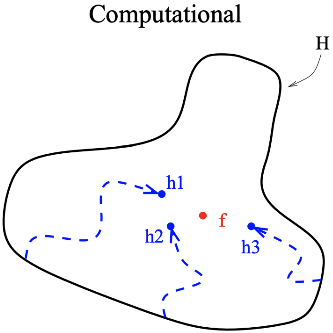
Statistical

- (Without sufficient data) many hypotheses can have the same level of accuracy on the training data;
- By “averaging” the votes of several ”good” classifiers the risk of choosing the wrong classifier is reduced.



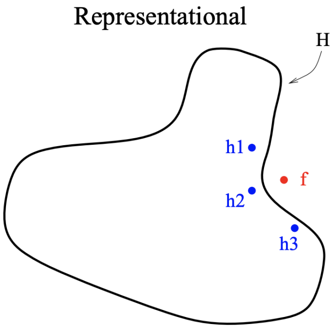
Computational

- Even in the presence of enough training examples learning algorithms may stuck in local optima;
- An ensemble constructed by running the local search from many different starting points may provide a better approximation to the true unknown function.



Representational

- In some applications of machine learning the true function cannot be represented by any of the hypotheses in H;
- By forming weighted sums of hypotheses drawn from H it may be possible to expand the space of representable functions.



Variance-bias decomposition

Given $y = f(x) + \epsilon$ and given a hypothesis g, the squared error can be decomposed as:

$$\mathbb{E}\left[(y - g(x))^2\right] = \textit{noise}^2 + \textit{bias}^2 + \textit{variance}$$

$$\begin{aligned} \text{Noise}^2 &: \mathbb{E}\left[(y - f(x))^2\right]\text{, that is irreducible;} \\ \text{Bias}^2 &: (\mathbb{E}[g(x)] - f(x))^2 \\ \text{Variance} &: \mathbb{E}\left[(g(x) - \mathbb{E}[g(x)])^2\right] \end{aligned}$$

Generally, averaging multiple hypotheses reduces variance, but can also reduce the bias.

Types of ensemble

Parallel : These methods take advantage of the **independence** between the base learners:

- Voting: Base learners make predictions then they pick the prediction which has the highest number of votes;
- Bagging: Multiple base learners are built from different samples of the training set to make predictions.

Sequential : These methods take advantage of the **dependence** between the base learners since the overall performance can be boosted in an incremental way. This approach is usually called boosting

Why parallel ensemble should work?

- Suppose that each base (binary) classifier h_i has an independent generalization error ϵ, i.e., $P(h_i(\mathbf{x}) \neq f(\mathbf{x})) = \epsilon$;
- Let us combine T of such classifier according to

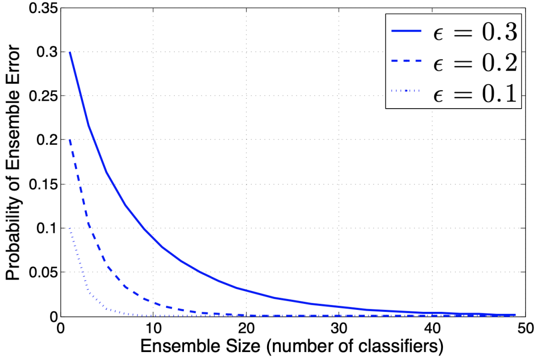
$$H(\mathbf{x}) = sign\left(\sum_{i=1}^{T} h_i(\mathbf{x})\right),$$

i.e., H makes an error when $> 50\%$ of its base classifiers make errors;
- Therefore, by **Hoeffding inequality** the generalization error of the ensemble is

$$P(H(\mathbf{x}) \neq f(\mathbf{x})) = \sum_{k=0}^{T/2} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \leq e^{-\frac{T}{2}(2\epsilon-1)^2},$$

clearly shows that the generalization error reduces exponentially to the ensemble size T.

Why this theory does not work in practice?

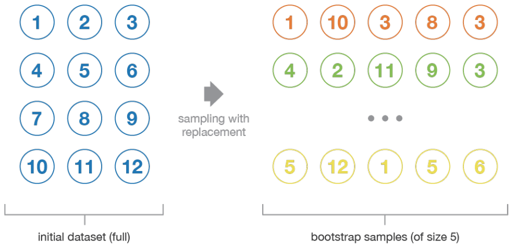


$\Rightarrow$ Errors of voters are **not independent**!!

Bootstrapping

How to achieve independence of models?

- **Bootstrapping**: sample with replacement M overlapping groups of instances of the same size.



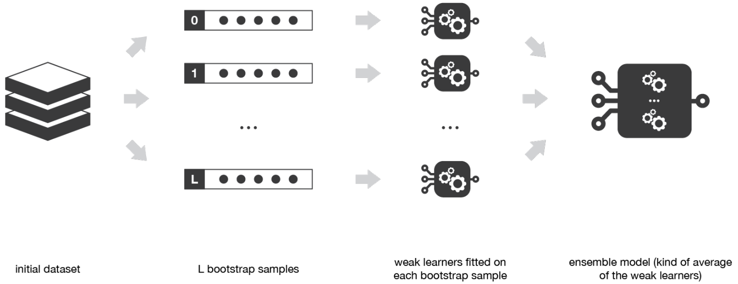
- **Feature randomization**: each model sees a random subset of features.

Bagging

Bagging

The Bootstrap AGGregating approach:

- Create k bootstrap samples;
- Train a distinct classifier on each sample;
- Classify new instance by majority vote / average.

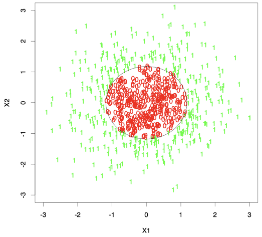


0
1
...
L
...
...
initial dataset
L bootstrap samples
weak learners fitted on each bootstrap sample
ensemble model (kind of average of the weak learners)

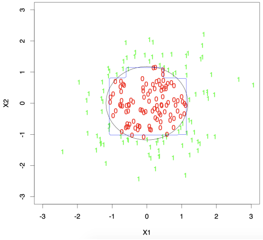
Bagging reduces variance

- Ideally, bagging eliminates variance altogether while keeping the bias almost unchanged; Averaging reduces variance: let $Z_1, \dots, Z_N$ be i.i.d random variables, then $Var(\frac{1}{N}\sum_i Z_i) = \frac{1}{N}Var(Z_i)$
- In practice, **weak learners are not independent** hence bagging tends to reduce variance and increase bias.

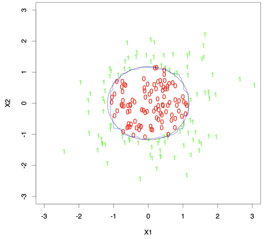
Bagging decision boundary



(a) Data



(b) Single DT



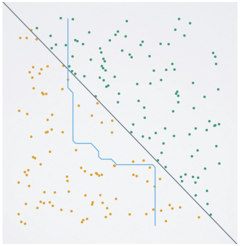
(c) Ensemble of DTs

When bagging fails?

- Bagging is bad if models are very similar (not independent enough);
- This happens if the learning algorithm is **stable**, i.e., models do not usually change much after changing a few instances;

Bagging is strongly affected by the quality of individual models,

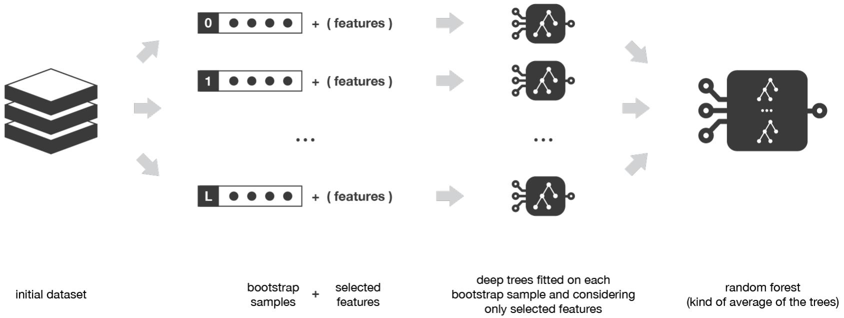
- e.g., Bagged decision stumps (trees with 1 decision node) are bad



Random Forests

Random forest algorithm

- Use k bootstrap replicates to train k different decision trees (DTs);
- At each node, pick a subset of features at random;
- Aggregate the predictions of each tree to make classification decision.

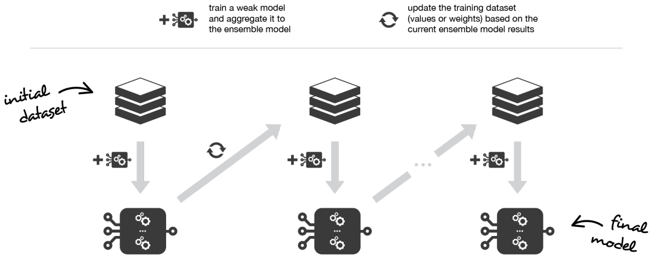


0
1
L
+ (features)
+ (features)
+ (features)
...
...
initial dataset
bootstrap samples
+
selected features
deep trees fitted on each bootstrap sample and considering only selected features
random forest (kind of average of the trees)

Boosting

Boosting

- Use the training set to train a simple (weak) predictor.
- Re-weight the training examples, putting more weight on examples that were poorly classified in the previous predictor;
- Repeat n times;
- Combine the simple hypotheses into a single accurate predictor.



train a weak model and aggregate it to the ensemble model
update the training dataset (values or weights) based on the current ensemble model results
initial dataset
...
final model

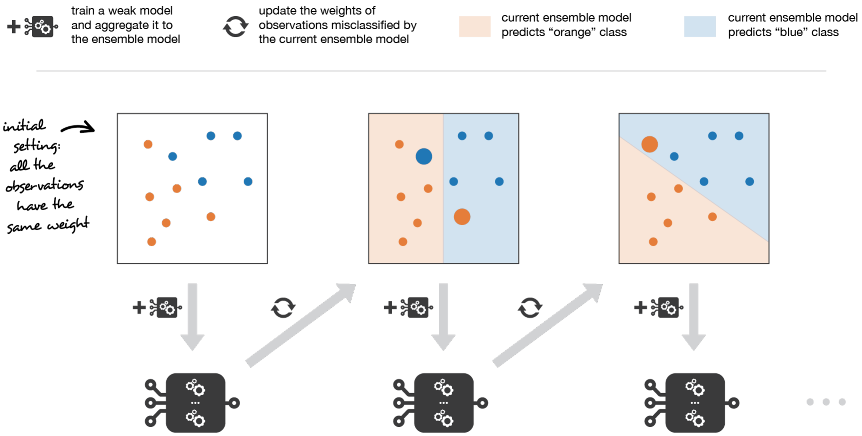
Boosting overview

- Boosting assumes weak learners slightly better than random guessing $\textit{accuracy}{=}0.5 + \epsilon$;
- It reduces bias by making each classifier focus on previous mistakes;
- AdaBoost (binary classification) is the most representative model;
- How can we take a “weak” classifier slightly better then chance and “boost” it to get low training error?
  - $\Rightarrow$ Sequential training with examples re-weighting.
- Hypothesis: $H(\mathbf{x}) = \textit{sign}(\sum_t \alpha_t h_t(\mathbf{x}))$. Note that we sum predictions (after sign) so, for example, sum of linear classifiers isn't a linear classifier!

Adaboost (Adaptive Boosting)

- **IDEA**: At each iteration t the training sample is reweighted (D_t), giving larger weights to points that were classified wrongly and train a new weak classifier;
- Weak learners need to maximize **weighted accuracy** (i.e., minimize weighted error $\epsilon_t = P_{i\sim D_t}[h_t(\mathbf{x}_i) \neq y_i] = \sum_i D_t(i)[h_t(\mathbf{x}_i) \neq y_i]$);
- The weight of each classifier is computed accordingly to its weighted error
$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t};$$
- Instance weights D_t are updated using an exponential rule $\Rightarrow$ harder examples weigh exponentially more than "easy" ones.
- Loss function: $E = \sum_{i=1}^{n} e^{-y_i H(\mathbf{x}_i)}$ where $H(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x})$.

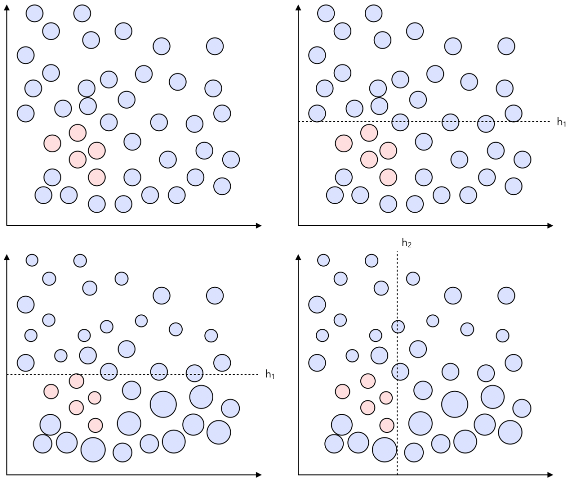
Adaboost (Adaptive Boosting)



train a weak model and aggregate it to the ensemble model
update the weights of observations misclassified by the current ensemble model
current ensemble model predicts “orange” class
current ensemble model predicts “blue” class

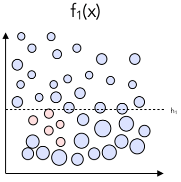
initial setting: all the observations have the same weight

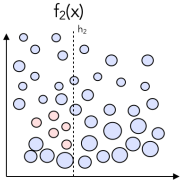
Adaboost w/ decision stumps - Training



Adaboost w/ decision stumps - Prediction

$$H(x) = \text{sign}\left(\alpha_1 \; f_1(x) \; + \; \alpha_2 \; f_2(x) \; + \; \ldots\right)$$





Adaboost loss function

$$E = \sum_{i=1}^{n} e^{-y_i H(\mathbf{x}_i)}$$

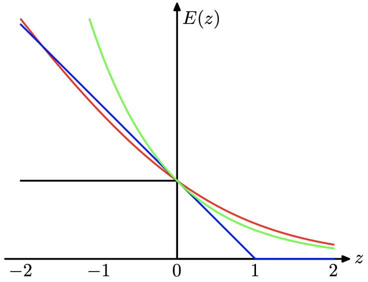


Figure: Plot of the exponential (green) and rescaled cross-entropy (red) error functions along with the hinge error (blue) used in support vector machines, and the misclassification error (black)

When/why does boosting works?

- Bagging may fail if the considered weak learners are mistaken in the same region
  - $\Rightarrow$ boosting solves the problem by concentrating the efforts on those regions!
- Weak learners have high bias. By combining them, we get more expressive classifiers. Hence, boosting is a bias-reduction technique;
- By focusing the effort on hard examples, boosting is very **sensitive to noise** (e.g., outliers) in the data.

Does Adaboost overfit?

Many iterations of Adaboost generate more and more complex hypothesis: *is boosting going to overfit?*

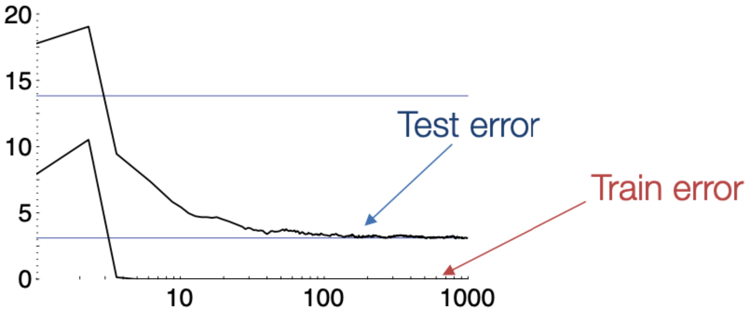


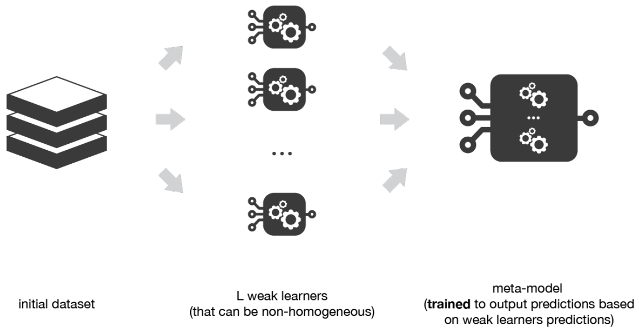
Figure: Typical run of Adaboost: Test error continues to drop even after training error reaches 0. **Conjecture: boosting does not overfit!**

Stacking

Both bagging and boosting assume we have a single *base learning* algorithm. What if we want to combine an arbitrary set of classifiers??

Stacking

Technique for combining an arbitrary set of learning models using a meta-model.


initial dataset
L weak learners
(that can be non-homogeneous)
meta-model
(**trained** to output predictions based
on weak learners predictions)

What is a meta-model?

- Any supervised model can be used as a meta-model!!
- Common choices:
  - Averaging (regression);
  - Majority vote (classification);
  - Linear regression (regression);
  - Logistic regression (classification).

When does stacking works?

- Stacking works best when the base model have **complementary strengths and weaknesses**, i.e., different inductive biases;
- Stacking performs very well in practice.

Recap

Topics:

- Bagging;
  - Random Forest.
- Boosting;
  - Adaboost.
- Stacking.

Try this at home (`sklearn.ensemble`):

- `BaggingClassifier`: for implementing bagging;
- `BoostingClassifier`: for trying Adaboost (e.g., check whether it does overfit or not);
- `StackingClassifier`: for implementing stacking.