Ensemble learning

Victor Kitov
v.v.kitov@yandex.ru

Yandex School of Data Analysis

Ensemble learning

Definition 1

Ensemble learning - using multiple machine learning methods for a given problem and integrating their output to obtain final result.

Synonyms: committee-based learning, multiple classifier systems.

Applications:

- supervised methods: regression, classification
- unsupervised methods: clustering

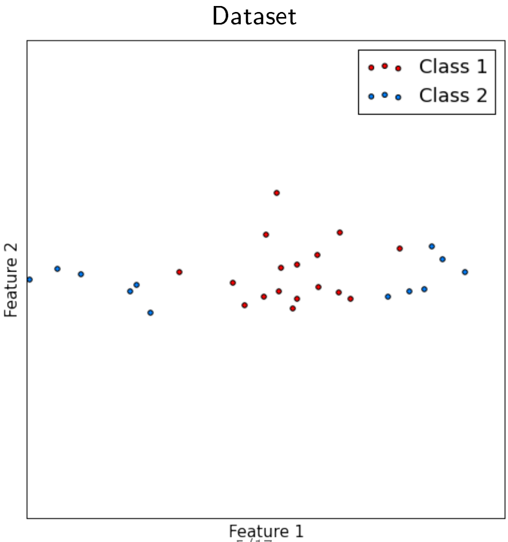
Ensembles use cases

- solving C class classification with many binary classifiers
- underfitting, high model bias
  - existing model hypothesis space is too narrow to explain the true one
  - different oversimplified models have bias in different directions, mutually compensating each other.
- overfitting, high model variance
  - avoid local optima of optimization methods
  - too small dataset to figure out concretely the exact model hypothesis
- when task itself promotes usage of ensembles with features of different nature
  - E.g. computer security:
    - multiple sources of diverse information (password, face detection, fingerprint)
    - different abstraction levels need to be united (current action, behavior pattern during day, week, month)

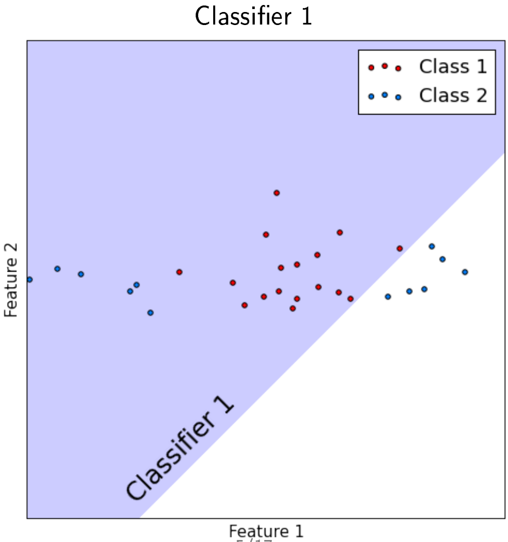
Table of Contents

1. Accuracy improvement demos
   - Accuracy improvement for classification
   - Accuracy improvement for regression

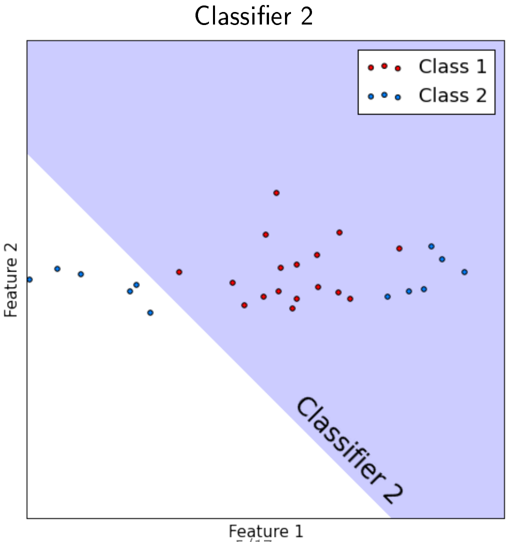
Classification: original model space too narrow



Classification: original model space too narrow

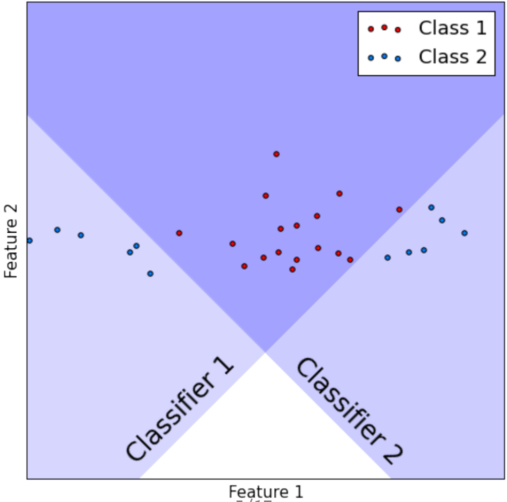


Classification: original model space too narrow



Classification: original model space too narrow

Classifier 1 and classifier 2 combined using AND rule

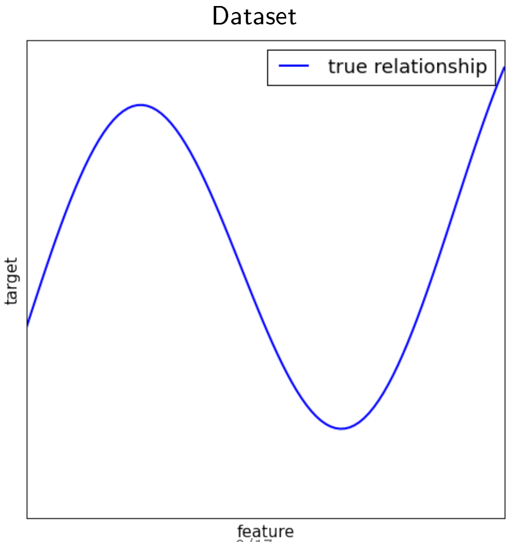


Motivation for ensembles

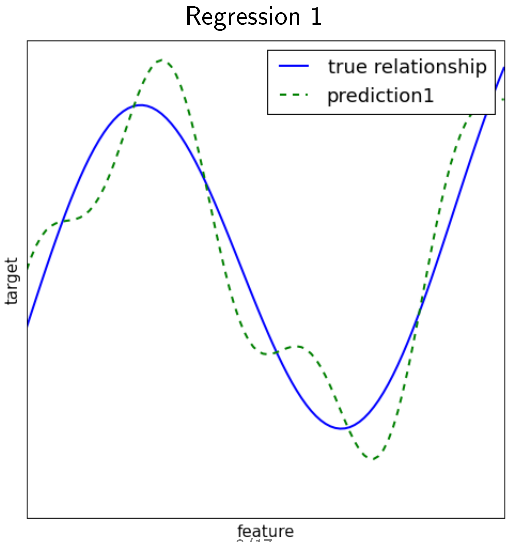
- Consider M classifiers $f_1(x), ...f_M(x)$, performing binary classification.
- Let probability of mistake be constant $p \in (0, \frac{1}{2})$: $p(f_m(x) = y) = p \, \forall m$
- Suppose all models make mistakes or correct guesses independently of each other.
- Let $F(x)$ be majority voting combiner.
- Then $p(F(x) \neq y) \to 0$ as $m \to \infty$

1. Accuracy improvement demos
   - Accuracy improvement for classification
   - Accuracy improvement for regression

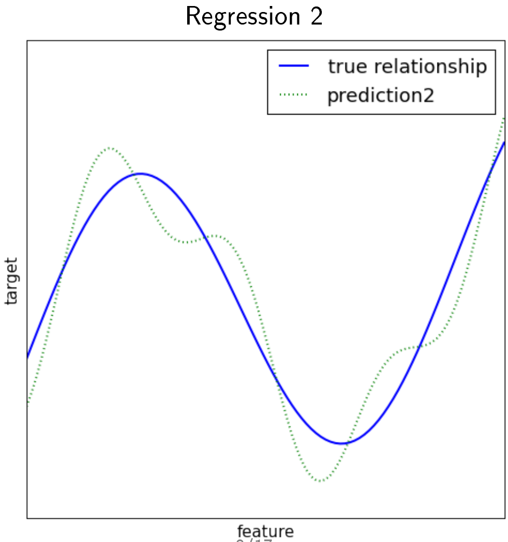
Regression: high variance



Regression: high variance



Regression: high variance



Regression: high variance

Regression 1 and regression 2 combined using averaging

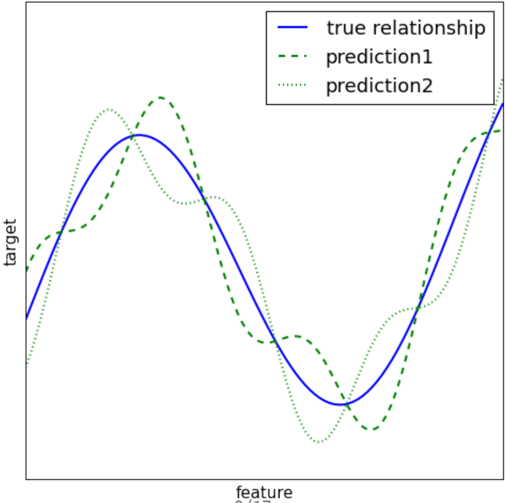


Table of Contents

Weighted averaging

Consider regression with K predictor models $f_k(x)$, $k = 1, 2, ...K$. (Alternatively we may consider K discriminant functions in classification)

Weighted averaging combiner

$$f(x) = \sum_{k=1}^{K} w_k f_k(x)$$

Naive fitting

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{N} \mathcal{L}(y_i, \sum_{k=1}^{K} w_k f_k(x_i))$$

will overfit. The mostly overfitted method will get the most weight.

Linear stacking

- Let training set $\{(x_i, y_i), i = 1, 2, ...N\}$ be split into M folds.
- Denote $fold(i)$ to be the fold, containing observation i
- Denote $f_k^{-fold(i)}$ be predictor k trained on all folds, except $fold(i)$.

Definition

Linear stacking is weighted averaging combiner, where weights are found using

$$\widehat{w} = \arg\min_w \sum_{i=1}^{N} \mathcal{L}(y_i, \sum_{k=1}^{K} w_k f_k^{-fold(i)}(x_i))$$

- For decreased overfitting we may add constraints$\{w_k \geq 0\}_{k=1}^K$ or regularizer $\sum_{k=1}^K \left(w_k - \frac{1}{K}\right)^2$.

General stacking

Definition

Generalized stacking is prediction

$$f(x) = A_\theta \left(f_1(x), f_2(x), \dots f_K(x) \right),$$

where A is some general form predictor and θ is a vector of parameters, estimated by

$$\widehat{\theta} = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}\left(y_i,\ A_\theta \left(f_1^{-fold(i)}(x), f_2^{-fold(i)}(x), \dots f_K^{-fold(i)}(x) \right) \right)$$

- Stacking is the most general approach
- It is a winning strategy in most ML competitions.
- $f_i(x)$ may be:
  - class number (coded using one-hot encoding).
  - vector of class probabilities
  - any initial or generated feature

Table of Contents

Bagging& random subspaces

- **Bagging**:
    - random selection of samples (with replacement)[1][2]
    - efficient for methods with high variance w.r.t. X, Y.
- **Random subspace method**:
    - random selection of features (without replacement)
- We can apply both methods jointly
- Also we may sample different

[1] *what is the probability that observation will not belong to bootstrap sample?*
[2] *what is the limit of this probability with $N \to \infty$?*

Random forests

Input: training dataset $TDS = \{(x_i, y_i), 1 = 1, 2, ...N\}$; the number of trees B and the size of feature subsets m.
for $b = 1, 2, ...B$:

1. generate random training dataset TDS^b of size N by sampling (x_i, y_i) pairs from TDS with replacement.
2. build a tree using TDS^b training dataset with feature selection for each node from random subset of features of size m (generated **individually for each node**).

Output: B trees. Classification is done using majority vote and regression using averaging of B outputs.

Comments

- Random forests use random selection on both samples and features
- Step 1) is optional.
- Left out samples may be used for evaluation of model performance.
  - *Out-of-bag* prediction: assign output to x_i, $i = 1, 2, ...N$ using majority vote (classification) or averaging (regression) among trees with $b \in \{b : (x_i, y_i) \notin T^b\}$
  - *Out-of-bag* quality - lower bound for true model quality.[3]
- Less interpretable than individual trees
- +: Parallel implementation
- -: different trees are not targeted to correct mistakes of each other

[3] why *lower* bound?

Comments

- Extra-Random trees-random sampling of (feature,value) pairs
  - more bias and less variance for each tree
  - faster training of each tree
- RandomForest and ExtraRandomTrees do not overfit with increasing B
- Each tree should have high depth
  - otherwise averaging over oversimplified trees will also give oversimplified model!