

Ensemble learning

Victor Kitov

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 Multiclass classification with binary classifiers
- 2 Accuracy improvement demos
- 3 Fixed integration schemes for classification
- 4 Stacking
- 5 Sampling ensemble methods

Multiclass classification with binary classifiers

- Solved problem: make C -class classification using many binary classifiers.
- Approaches:
 - one-versus-all
 - for each $c = 1, 2, \dots, C$ train binary classifier on all objects and output $\mathbb{I}[y_n = c]$,
 - assign class, getting the highest score in resulting C classifiers.
 - one-versus-one
 - for each $i, j \in [1, 2, \dots, C], i \neq j$ learn on objects with $y_n \in \{i, j\}$ with output y_n
 - assign class, getting the highest score in resulting $C(C - 1)/2$ classifiers.
 - error correcting codes

Error correcting codes

- Used in classification
- Each class ω_j is coded as a binary codeword W_j consisting of B bits:

$$\omega_j \rightarrow W_j$$

- Minimum sufficient amount of bits to code C classes is $\lceil \log_2 C \rceil$
- Given x , B binary classifiers predict each bit of the class codeword.
- Class is predicted as

$$\hat{c}(x) = \arg \min_c \sum_{b=1}^B |W_{cb} - \hat{p}_b(x)|$$

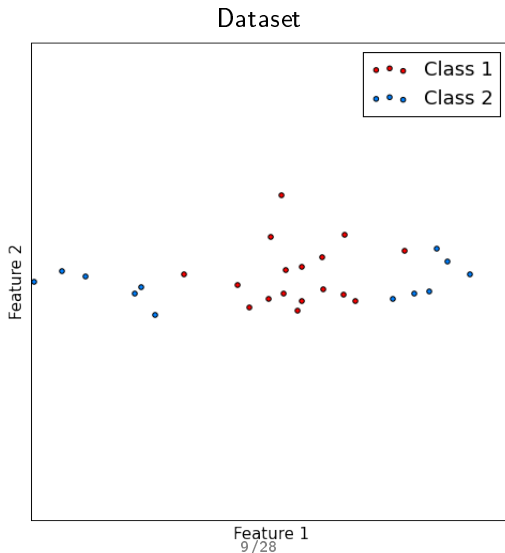
- where W_{cb} is the b -th bit of codeword, corresponding to class c .
- More bits are used to make classification more robust to errors of individual binary classifiers.
- Codewords are selected to have maximum mutual Hamming distance or randomly.

Table of Contents

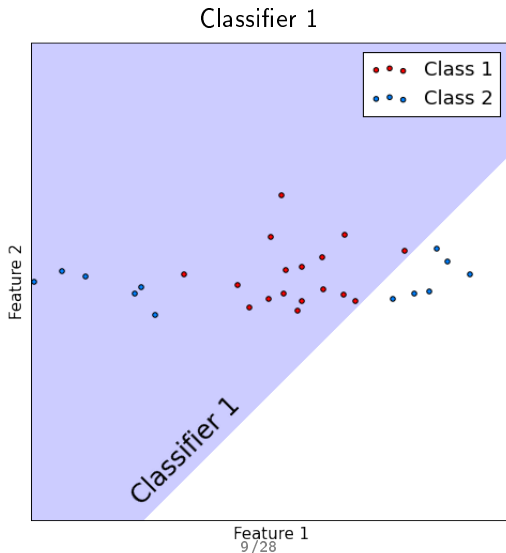
- 1 Multiclass classification with binary classifiers
- 2 Accuracy improvement demos**
 - Accuracy improvement for classification
 - Accuracy improvement for regression
- 3 Fixed integration schemes for classification
- 4 Stacking
- 5 Sampling ensemble methods

- 2 Accuracy improvement demos
 - Accuracy improvement for classification
 - Accuracy improvement for regression

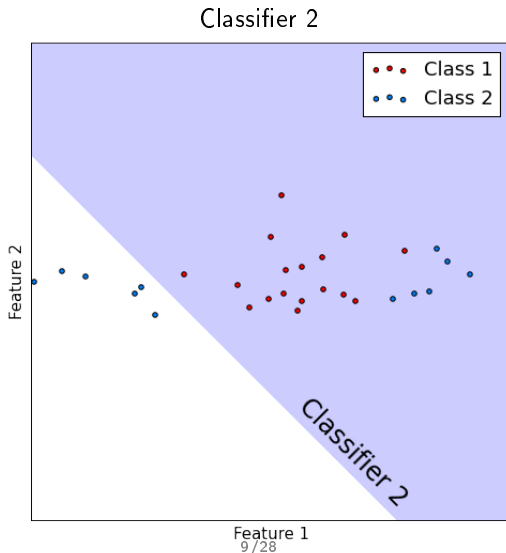
Classification: original model space too narrow



Classification: original model space too narrow

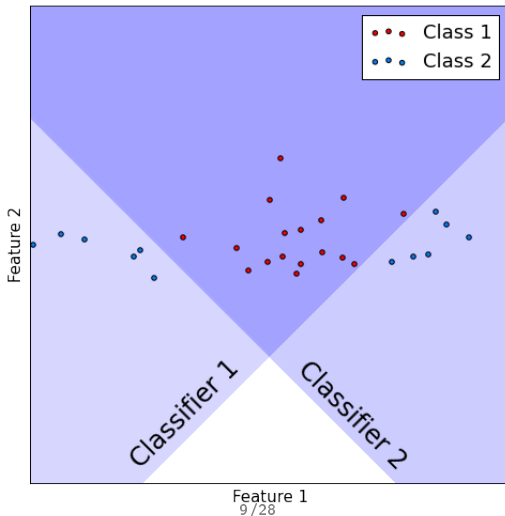


Classification: original model space too narrow



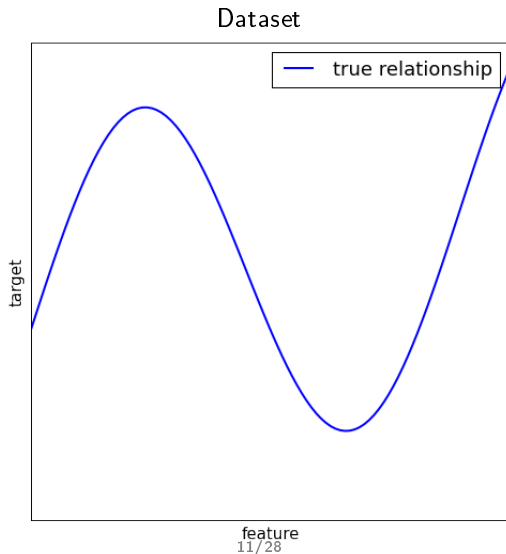
Classification: original model space too narrow

Classifier 1 and classifier 2 combined using AND rule

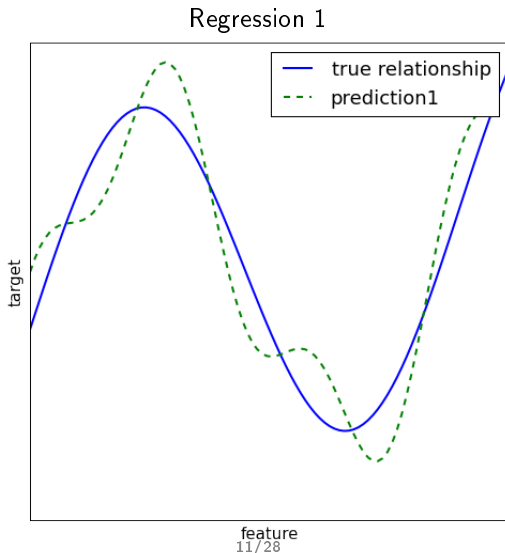


- 2 Accuracy improvement demos
 - Accuracy improvement for classification
 - Accuracy improvement for regression

Regression: high variance

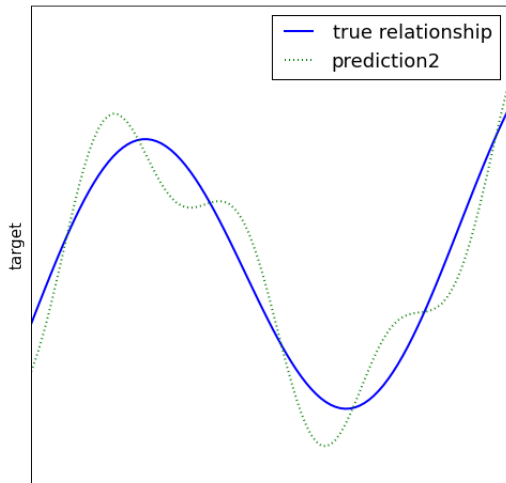


Regression: high variance



Regression: high variance

Regression 2



Regression: high variance

Regression 1 and regression 2 combined using averaging

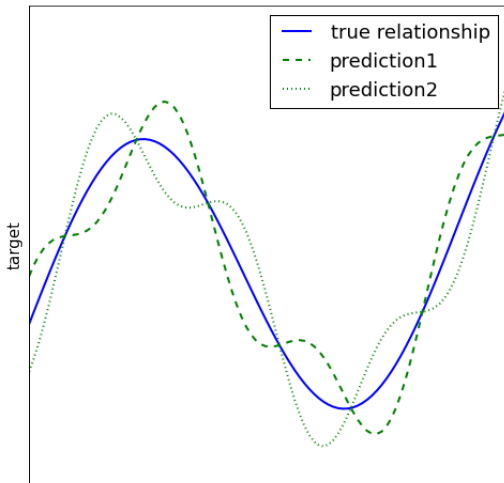


Table of Contents

- 1 Multiclass classification with binary classifiers
- 2 Accuracy improvement demos
- 3 Fixed integration schemes for classification**
- 4 Stacking
- 5 Sampling ensemble methods

Fixed combiner at class level

Output of base learner k

Exact class: ω_1 or ω_2 .

Combiner predicts ω_1 if:

- all classifiers predict ω_1 (AND rule)
- at least one classifier predicts ω_1 (OR rule)
- at least k classifiers predict ω_1 (k-out-of-N)
- majority of classifiers predict ω_1 (majority vote)

Each classifier may be assigned a weight, based on its performance:

- weighted majority vote
- weighted k-out-of-N (based on score sum)

Fixed combiner - ranking level

Output of base learner k

Ranking of classes:

$$\omega_{k_1} \succeq \omega_{k_2} \succeq \dots \succeq \omega_{k_C}$$

Ranking is equivalent to scoring of each class (with incomparable scoring between classifiers).

Definition 2

Let $B_k(i)$ be the count of classes scored below ω_i by classifier k . **Borda count** $B(i)$ of class ω_i is the total number of classes scored below ω_i by all classifiers:

$$B(i) = \sum_{k=1}^K B_k(i)$$

Combiner predicts ω_i where $i = \arg \max_i B(i)$

Fixed combiner at class probability level

Output of base learner k

Vectors of class probabilities:

$$[p^k(\omega_1), p^k(\omega_2), \dots, p^k(\omega_C)]$$

Combiner predicts ω_i if $i = \arg \max_i F(p^1(\omega_i), p^2(\omega_i), \dots, p^K(\omega_i))$

- F = mean or median.

Table of Contents

- 1 Multiclass classification with binary classifiers
- 2 Accuracy improvement demos
- 3 Fixed integration schemes for classification
- 4 Stacking**
- 5 Sampling ensemble methods

Weighted averaging

Consider regression with K predictor models $f_k(x)$, $k = 1, 2, \dots, K$. (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

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

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

Linear stacking

- Let training set $\{(x_i, y_i), i = 1, 2, \dots, N\}$ 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

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

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

General stacking

Definition

Generalized stacking is prediction

$$f(x) = A_{\theta}(f_1(x), f_2(x), \dots, f_K(x)),$$

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

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

- 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

- 1 Multiclass classification with binary classifiers
- 2 Accuracy improvement demos
- 3 Fixed integration schemes for classification
- 4 Stacking
- 5 Sampling ensemble methods**
 - Bias-variance decomposition
 - Bagging and random forest

- 5 Sampling ensemble methods
 - Bias-variance decomposition
 - Bagging and random forest

Bias-variance decomposition

Theorem 3 (Bias-variance decomposition)

Unknown relationship $y = f(x) + \varepsilon$ is reconstructed using a set of points (x_n, y_n) , $n = 1, 2, \dots, N$ as $\hat{f}(x)$. Noise ε is independent of any x , $\mathbb{E}\varepsilon = 0$ and $\text{Var}[\varepsilon] = \sigma^2$. Then

$$\mathbb{E}_{X,Y}[\hat{f}(x) - f(x)]^2 = [\mathbb{E}_{X,Y}\hat{f}(x) - f(x)]^2 + \mathbb{E}_{X,Y}[\hat{f}(x) - \mathbb{E}\hat{f}(x)]^2$$

$$\mathbb{E}_{X,Y,\varepsilon}[\hat{f} - y]^2 = [\mathbb{E}_{X,Y}\hat{f}(x) - f(x)]^2 + \mathbb{E}_{X,Y}[\hat{f}(x) - \mathbb{E}\hat{f}(x)]^2 + \sigma^2$$

Comments:

- Intuition: $MSE = \text{bias}^2 + \text{variance} + \text{irreducible error}$
- Dartz visualization of models with high bias/variance

Proof of bias-variance decomposition (3)

For simplicity of notation define $f := f(x)$, $\hat{f} := \hat{f}(x)$, $\mathbb{E}\{\cdot\} := \mathbb{E}_{X,Y,\varepsilon}\{\cdot\}$

$$\begin{aligned}\mathbb{E}(\hat{f} - f)^2 &= \mathbb{E}(\hat{f} - \mathbb{E}\hat{f} + \mathbb{E}\hat{f} - f)^2 = \mathbb{E}(\hat{f} - \mathbb{E}\hat{f})^2 + (\mathbb{E}\hat{f} - f)^2 \\ &\quad + 2\mathbb{E}[(\hat{f} - \mathbb{E}\hat{f})(\mathbb{E}\hat{f} - f)] \\ &= \mathbb{E}(\hat{f} - \mathbb{E}\hat{f})^2 + (\mathbb{E}\hat{f} - f)^2\end{aligned}$$

We used that $(\mathbb{E}\hat{f} - f)$ is a constant number and hence

$$\mathbb{E}[(\hat{f} - \mathbb{E}\hat{f})(\mathbb{E}\hat{f} - f)] = (\mathbb{E}\hat{f} - f)\mathbb{E}(\hat{f} - \mathbb{E}\hat{f}) = 0.$$

$$\begin{aligned}\mathbb{E}(\hat{f} - y)^2 &= \mathbb{E}(\hat{f} - f - \varepsilon)^2 = \mathbb{E}(\hat{f} - f)^2 + \mathbb{E}\varepsilon^2 - 2\mathbb{E}[(\hat{f} - f)\varepsilon] \\ &= \mathbb{E}(\hat{f} - \mathbb{E}\hat{f})^2 + (\mathbb{E}\hat{f} - f)^2 + \sigma^2\end{aligned}$$

We used that $\mathbb{E}[(\hat{f} - f)\varepsilon] = \mathbb{E}[(\hat{f} - f)]\mathbb{E}\varepsilon = 0$ since ε is independent of X, Y .

- 5 Sampling ensemble methods
 - Bias-variance decomposition
 - Bagging and random forest

Bagging & random subspaces

- Bagging
 - random selection of samples (with replacement)¹²
 - 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 $N \rightarrow \infty$?*

Random forests

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

for $b = 1, 2, \dots, B$:

- 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 , $i = 1, 2, \dots, N$ 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

³why 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!