SDU
SYDDANSK UNIVERSITET

Data Mining and Machine Learning

Part 8: Ensemble Learning

Melih Kandemir

University of Southern Denmark

Spring 2023

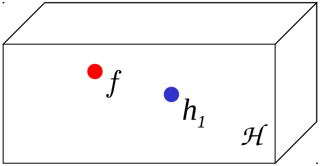
Classification as Approximation of the Target Function

Recall the classification task:

- For some domain $\mathcal{D}$ and a set of classes $C = \{c_1, \ldots, c_k\}$, $k \geq 2$, each $o \in \mathcal{D}$ belongs uniquely to some $c_i \in C$, i.e., there is a function $f : \mathcal{D} \to C$.
- Given a set of objects $O = \{o_1, o_2, \ldots, o_n\} \subseteq \mathcal{D}$ and a mapping $(O \to C) \subset f$ (examples):
  We want to also map any object $o_m \in \mathcal{D} \setminus O$ to C.
- A classifier is trained on some training set $TR \subseteq O$ to learn the mapping function (a model or hypothesis) $h : \mathcal{D} \to C$.
- Ideally we have $\forall o \in TR : h(o) = f(o)$ (if not for all, we should have this at least for most examples o).
- In general, the hypothesis h should be an approximation of f.

Hypothesis Space and Target Function

- The true function f (target function) is unknown.
- Based on the training data and its hypothesis space $\mathcal{H}$, a learning algorithm looks for the hypothesis $h_i \in \mathcal{H}$ that fits optimally to the training data.



- The true function f is not necessarily an element of the hypothesis space!

Approximation and Accuracy

- We apply h on elements $x \in \mathcal{D}$ to predict class $c_i = f(x)$.
- The accuracy of a classifier (hypothesis h) is the probability (statistically/empirically: frequency) of its predictions being correct:

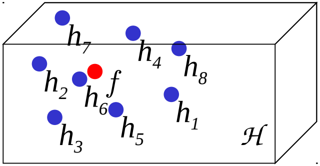
$$\mathrm{acc}(h) = \Pr(h(x) = f(x))$$

or

$$\mathrm{err}(h) = \Pr(h(x) \neq f(x)) = 1 - \mathrm{acc}(h)$$

Reduction of Error Probability by Averaging

- The basic idea of using ensembles (combinations of several classifiers) is the reduction of the probability of errors by asking a jury of experts instead of just one expert and by letting them vote to find a common prediction.
- Intuitively, we expect a better approximation of f by aggregating (e.g., averaging) over several approximations h_i.



- Cf. the considerations on Bayes optimal classification.

Simple Voting

- If we have two classes $C = \{-1, 1\}$, a simple voting procedure could be defined as:
  - Learn hypotheses $h_1, \ldots, h_k$ with associated weights $w_1, \ldots, w_k$.
  - ensemble classifier $\hat{h}$ is given by:

$$\hat{h}(x) = \begin{cases} \mathit{if}\ w_1 h_1(x) + \ldots + w_k h_k(x) \geq 0 & : x \mapsto 1 \\ \mathit{if}\ w_1 h_1(x) + \ldots + w_k h_k(x) < 0 & : x \mapsto -1 \end{cases}$$

- We can have $w_1 = \ldots = w_k = 1$ (i.e., unweighted voting).
- Weights can be based on (empirical) reliability of individual classifiers.
- We can have more complex voting procedures (and need to have, if we have more than two classes), resulting in many ensemble methods.

Error-Rate of Ensembles

$$\hat{h}(x) = \begin{cases} \text{if } w_1 h_1(x) + \ldots + w_k h_k(x) \geq 0 & : x \mapsto 1 \\ \text{if } w_1 h_1(x) + \ldots + w_k h_k(x) < 0 & : x \mapsto -1 \end{cases}$$

- The error rate of an ensemble depends on the error rate of the base classifiers (ensemble members) and on how many we combine.
- Assuming $\mathrm{err}(h_1) = \ldots = \mathrm{err}(h_k) = err$, the ensemble error follows a binomial distribution (the ensemble is wrong, if at least half of its members are wrong):

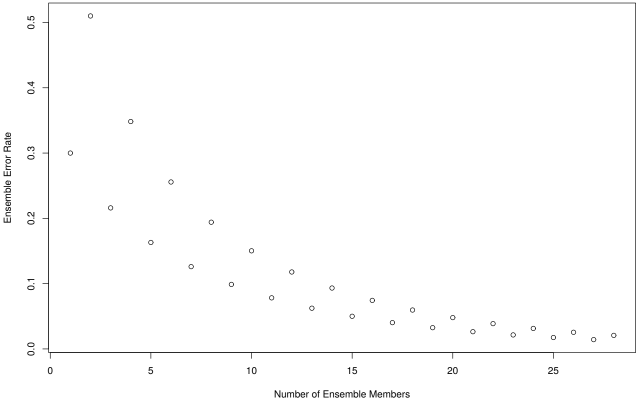
$$\overline{err}\,(k, err) = \sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} err^i (1 - err)^{k-i}$$

(relates to Condorcet's Jury theorem)

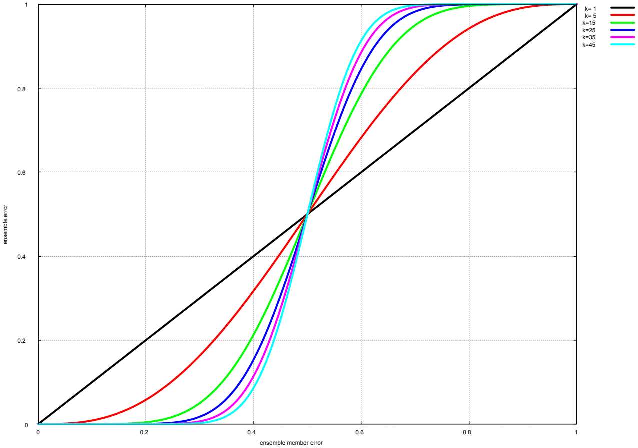
Error-Rate of Ensembles

Error rate of the ensemble, depending on the number of ensemble members (base classifiers), assuming $err = 0.3$:

$$\overline{err}\,(k, 0.3) = \sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} 0.3^i (1-0.3)^{k-i}$$



Error-Rate of Ensembles

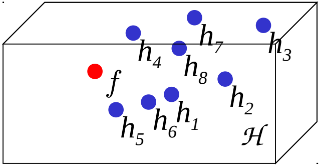


$$\overline{err}\,(k, err) = \sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} err^i (1 - err)^{k-i}$$

Independence of Errors

$$\overline{err}(k, err) = \sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} err^i (1 - err)^{k-i}$$

- Note that we require independence of errors for this formula.
- If the errors are not independent, we cannot expect much improvement from the average.



Conclusions

Note that:

We observed two necessary conditions for an improvement of the error rate by combining base classifiers into an ensemble:

1. *All base classifiers are accurate.*
2. *The individual base classifiers are diverse.*

- Accuracy is a mild condition — they need to be at least better than random.
- Diversity: there should be no (strong) correlation between the errors.
- Can we optimize both, diversity and accuracy?

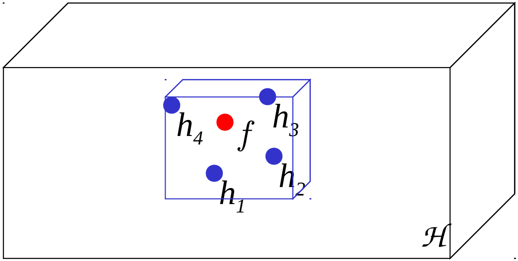
Reasons for Diversity

Without engineering diversity artificially, there are already inherent reasons for getting diverse classifiers on one and the same classification problem:

- statistical variance
- computational variance
- representation problem
- uncertain/noisy target function

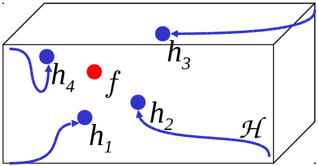
Statistical Variance

- The hypothesis space is too big to be explored based on the limited amount of training examples.
- Combination of several hypotheses reduces the risk of being very far off.



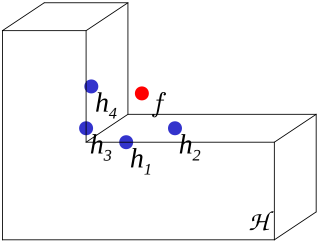
Computational Variance

- Some learning algorithms cannot guarantee to find the best hypothesis, as that would be computationally infeasible.
- Instead, they use learning heuristics such that the search could get stuck in local optima.
- Combination of several hypotheses reduces the risk to stick to the wrong (local) optimum.



Representation Problem

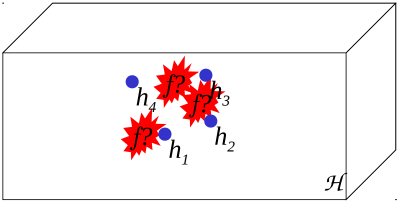
- The hypothesis space does not contain any close approximation to the true target function f.
- The combination of several hypotheses can effectively enlarge the hypothesis space.



SDU SYDDANSK UNIVERSITET

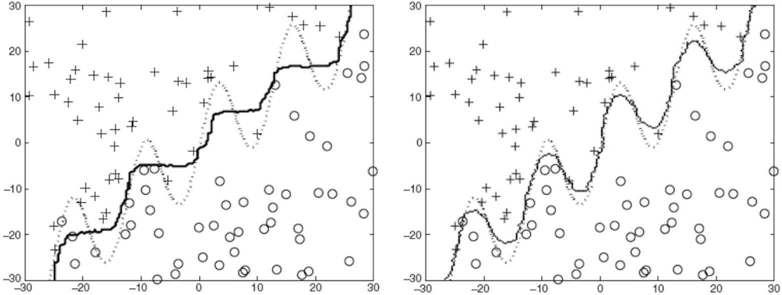
Uncertain Target Function

- The training examples do not allow to draw unambiguous conclusions on the target function.
  - Noisy training data: there could be contradictory examples.
  - Some class labels might be non-deterministic.
- Combination of several hypotheses reduces the risk, to approximate the wrong target.



Different Approximation Error of Different Classifiers

True decision boundary (dotted) and average (over 100 classifiers trained on 100 variants of the data set) decision boundary (solid) of decision trees (left) and k-nearest neighbor classifiers (right).



based on a figure by Tan et al.

Bias and Variance of SVM

Varying the regularization parameter C for a quadratic kernel (trade-off between slack variables and margin):

- small C emphasizes margin (stronger bias)
- large C de-emphasizes margin (weaker bias)

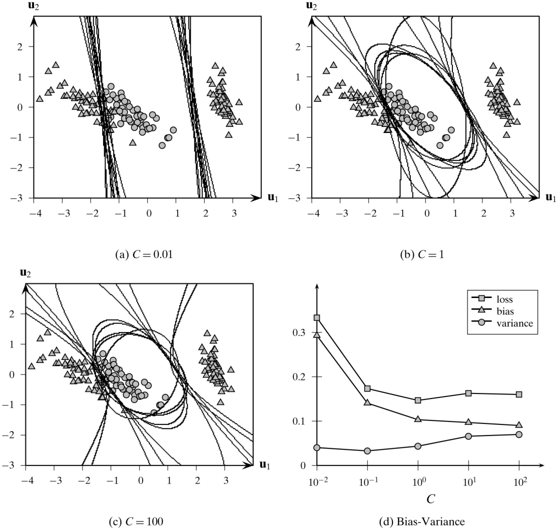
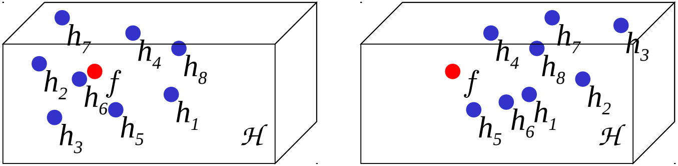


Figure by Zakia et al.

Ensembles

- Any individual classifier would have either a strong bias or a large variance on a non-trivial learning task.
- The combination of classifiers can reduce both, bias and variance:
  - We can combine classifiers with a weak bias, thus a large variance.
  - Averaging reduces the overall variance.



  - We can combine classifiers with strong bias (and thus typically small variance), but choose them in a way to diversify the biases.
  - Averaging reduces the overall bias.

Possibilities to Achieve Diverse Classifiers

- vary the training set
  - bagging
  - boosting
- manipulate data descriptors
  - use different subspaces/projections
  - use different representations
- manipulate class labels
  - different mappings from polytomous to dichotomous problems

meta methods

- manipulate learning algorithm
  - use elements of randomness
  - use different start configurations for local optimizers

specialized methods

Varying the Training Set

- based on the instability of learning algorithms:
  - An algorithm is the more stable, the less classifiers (hypotheses) differ that have been learned on varied training data for the same classification problem.
  - Instability is based on the variance of the learned decision boundary: high variance makes the learner susceptible to overfitting.
  - For an unstable learning algorithm, small changes of the training set can induce large changes in the model.
- To build ensembles based on varying the training set, instable learners are beneficial, e.g.:
  - decision trees
  - neuronal nets
  - rule learners (not covered in the lecture)

Bagging

- Bagging is an acronym for **B**ootstrap **Agg**regat**ing.**
- Idea: get diverse training sets by repeated bootstrapping.

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Bagging trains a classifier on each bootstrap sample and aggregates the models.
- With unstable learners, sufficiently diverse hypotheses will be learned.
- New data objects are classified by voting over all learned hypotheses.

Boosting

- While the bootstrap is sampled uniformly, boosting assigns a weight to each data object.
- The weights are adjusted (increased) for difficult objects (where previous hypotheses made errors).
- The weights change the probability of drawing the object in the next round of sampling.
- As a result, difficult objects will show up more frequently in the next round and thus get implicitely a higher weight for training the classifier.

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Varying the Data Descriptors

DM566

Melih Kandemir

Motivation for Ensemble Learning

Diversity

Ensemble Methods

Varying the Training Set

Varying Data Descriptors

Manipulating Class Labels

Manipulating the Learning Algorithm

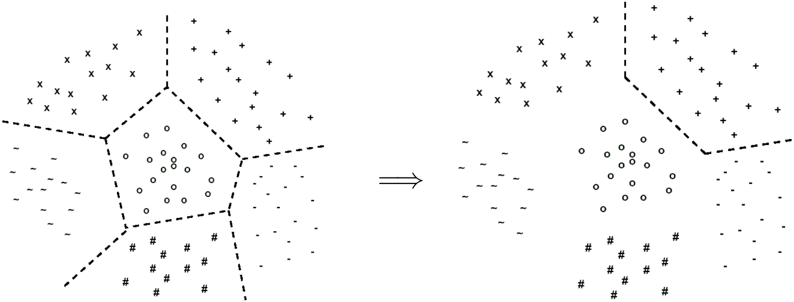
- “feature bagging”:
  - sample attributes
  - learn in the subspace
  - repeat several times and combine the models/predictions
- instead of sampling individual attributes, use different feature combinations/projections (e.g., random projections)
- use different feature spaces

Manipulating Class Labels

- Some classification algorithms can only solve dichotomous classification problems (e.g., SVMs).
- Complex problems with more than 2 classes can be tackled by learning several classifiers on subproblems, reduced to two classes.
- Most prominent methods:
  - one-versus-rest
  - all pairs
  - Error correcting output codes (ECOC)

One-Versus-Rest

- a.k.a. one-versus-all, one-versus-others, one-per-class
- For n classes, we train n classifiers, each separating one of the classes, in turn, from all the others.



Figures by Fue et al.

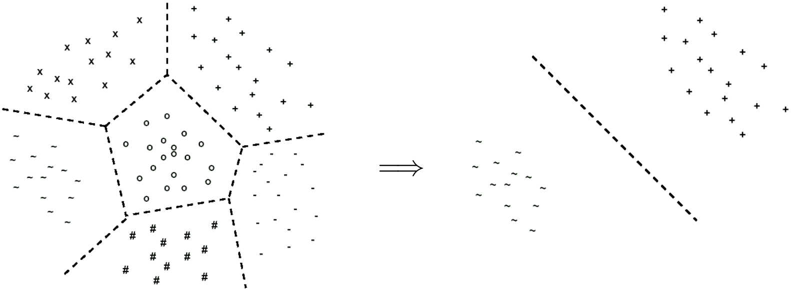
All Pairs

DM566

Melih Kandemir

Motivation for Ensemble Learning
Diversity
Ensemble Methods
Varying the Training Set
Varying Data Descriptors
Manipulating Class Labels
Manipulating the Learning Algorithm

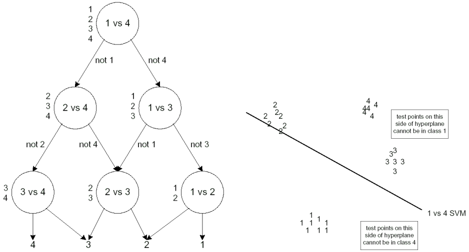
- a.k.a. all-versus-all, one-versus-one, round robin, pairwise
- For each pair of n classes, we train a classifier for separating just these classes from each other.



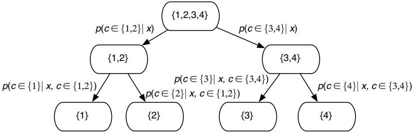
Figures by Fue et al.

Combination of Multiple Two-Class Classifiers

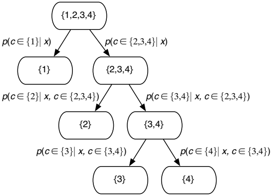
- collect all votes
- directed acyclic graph: sequential votes, follow only up on the winners



Combination of Multiple Two-Class Classifiers



- nested dichotomies



- incorporate domain knowledge (hierarchies of classes): hierarchically nested dichotomies

Error Correcting Output Codes (ECOC)

ECOC defines both: getting diverse classifiers, and combining their votes.

Diversity

- Set C of classes is randomly split k times into two subsets A and B.
- Examples $\in A$ get assigned the new label -1, the other classes ($\in B$) get label 1.
- Train k classifiers on the resulting k two-class problems.

Combination

- If in one of the problems a classifier votes for A, all classes $\in C$ that belong to A in this iteration get a vote.
- The class $\in C$ receiving most votes is the decision of the ensemble.

ECOC — Example

- Let $C = \{c_1, c_2, c_3, c_4\}$, choose $k = 7$ (i.e., we have a 7-bit encoding):

class	code						
c_1	1	1	1	1	1	1	1
c_2	0	0	0	0	1	1	1
c_3	0	0	1	1	0	0	1
c_4	0	1	0	1	0	1	0

- For each bit of the code, we train a classifier.
- We get seven decisions, e.g., $(0, 1, 1, 1, 1, 1, 1)$ — what is the ensemble's decision?

ECOC — Motivation

- The name "error correcting" relates to the idea that we introduce a redundancy for the decisions.
- The codes can be chosen randomly.
- For a good diversity, the codes should separate well:

  row separation: each pair of codes should have a large Hamming-distance.

  column separation: the k binary classifiers should be rather uncorrelated.
- What is the Hamming distance of the vote $(0, 1, 1, 1, 1, 1, 1)$ in our example?

Use vs. Design of Random Elements

Motivation for Ensemble Learning
Diversity
Ensemble Methods
Varying the Training Set
Varying Data Descriptors
Manipulating Class Labels
Manipulating the Learning Algorithm

- Some learning algorithms include random elements.
- An example are random starting points for local optimizers (e.g., the weights used in a neural net).
- Other learning algorithms can be changed to incorporate random elements.
- The greedy optimization used in decision trees is an obvious choice, leading to random forests.

Random Forests

"Random Forests" is a speaking name for an ensemble consisting of trees — decision trees that involve some random component:

- train each decision tree on an independent randomly selected subset of the features (Forest-RI: Random Input);
- generate at each node a set of random linear combinations of a subset of the features, select the best of them for the split (Forest-RC: Random Combination);
- select at each node randomly one of the n best splits;

  or

- some combination of the three approaches.