Ensemble methods

COMP5318 Machine Learning and Data Mining

semester 1, 2021, week 5b

Irena Koprinska

Reference: Tan ch.4.10, Witten ch.12, Géron ch.7

THE UNIVERSITY OF SYDNEY

Outline

- Motivation for creating ensembles
- Ensemble methods
  - Bagging
  - Boosting – AdaBoost and Gradient Boosting
  - Random Forest

What is an ensemble method?

- An ensemble combines the predictions of multiple classifiers
- The classifiers that are combined are called base classifiers, they are created using the training data
- There are various ways to make a prediction for new examples, e.g. by taking the majority vote of the base classifiers, the weighed vote, etc.
- Ensembles tends to work better than the base classifiers they combine

- Example of an ensemble:
  - 3 base classifiers are trained on the training data, e.g. k nearest neighbor, logistic regression and decision tree
  - To classify a new example, the individual predictions are combined by taking the majority vote

Motivation for ensembles

- When do ensemble methods work?
- Let's consider an example which illustrates how ensembles can improve the performance of a single classifier:
- An ensemble of 25 binary classifiers. Each base classifier has an error rate ε =0.35 on the test set (i.e. accuracy=0.65). To predict the class of a new example, the predictions of the base classifiers are combined by majority vote.

- Case 1: The base classifiers are identical, i.e. make the same mistakes. What will be the error rate of the ensemble on the test set?
  - 0.35

Motivation for ensembles (2)

- Case 2: The base classifiers are independent, i.e. their errors are not correlated. What will be the error rate of the ensemble on the test set?

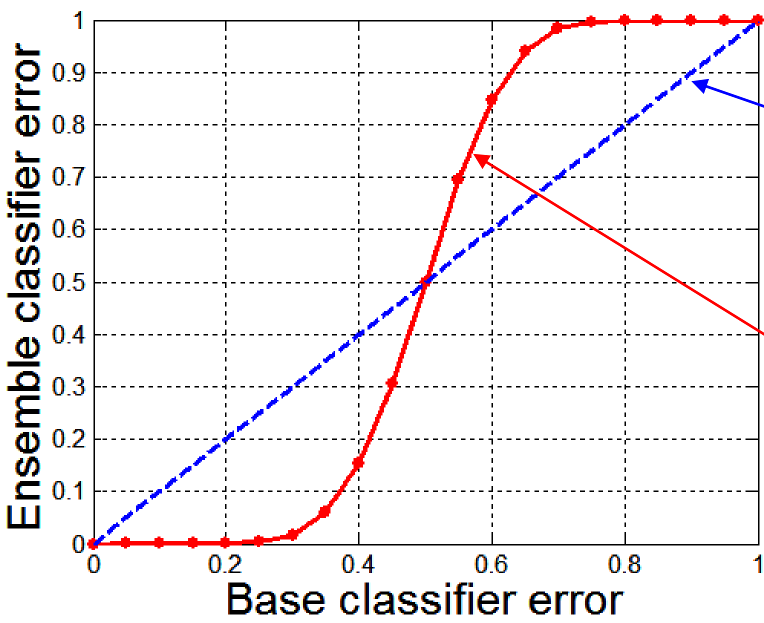
- When will a new example be misclassified? Only if more than half of the base classifiers predict incorrectly.
- It can be shown that the error rate of the ensemble will be:

$$e_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

- 0.06<0.35, i.e. the error rate of the ensemble is much lower than the error rate of the base classifiers

Error rate graph – ensemble vs base classifier

- For our example of 25 binary classifiers:

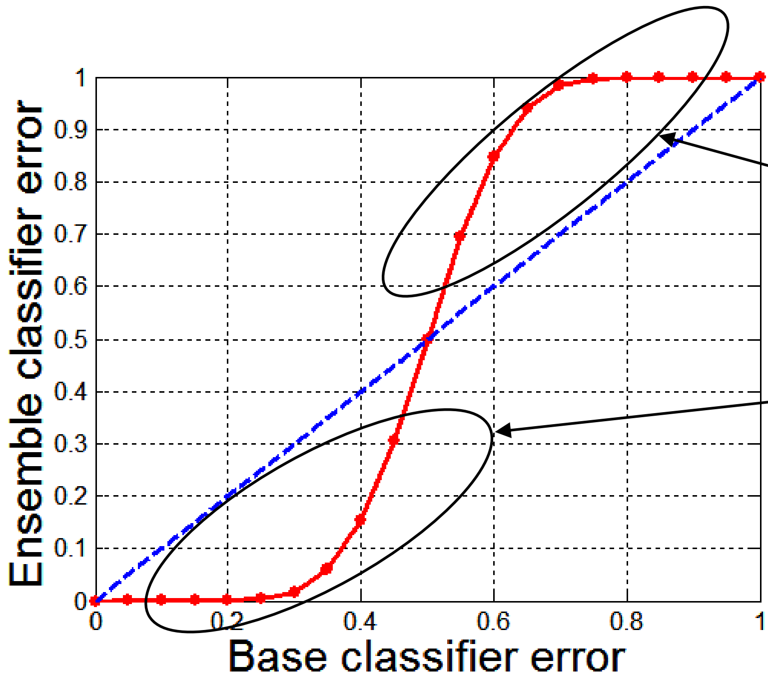


Case 1: base classifiers are identical; ensemble's error = base classifier's error

Case 2: base classifiers are independent; ensemble's error ≠ base classifier's error

Error rate graph – ensemble vs base classifier (2)

- When is the ensemble better and worse than the base classifier?



Ensemble is worse than a base classifier when $\varepsilon > 0.5$

Ensemble is better than a base classifier when $\varepsilon < 0.5$

- The ensemble is better if the error of the base classifier $\varepsilon < 0.5$, i.e. the predictions of the base classifier (which is binary) are better than random guess

When ensembles work well

- Conditions for an ensemble to perform better than a single classifier:
  - The base classifiers should be good enough, i.e. better than a random guessing (ε <0.5 for binary classifiers)
  - The base classifiers are independent of each other

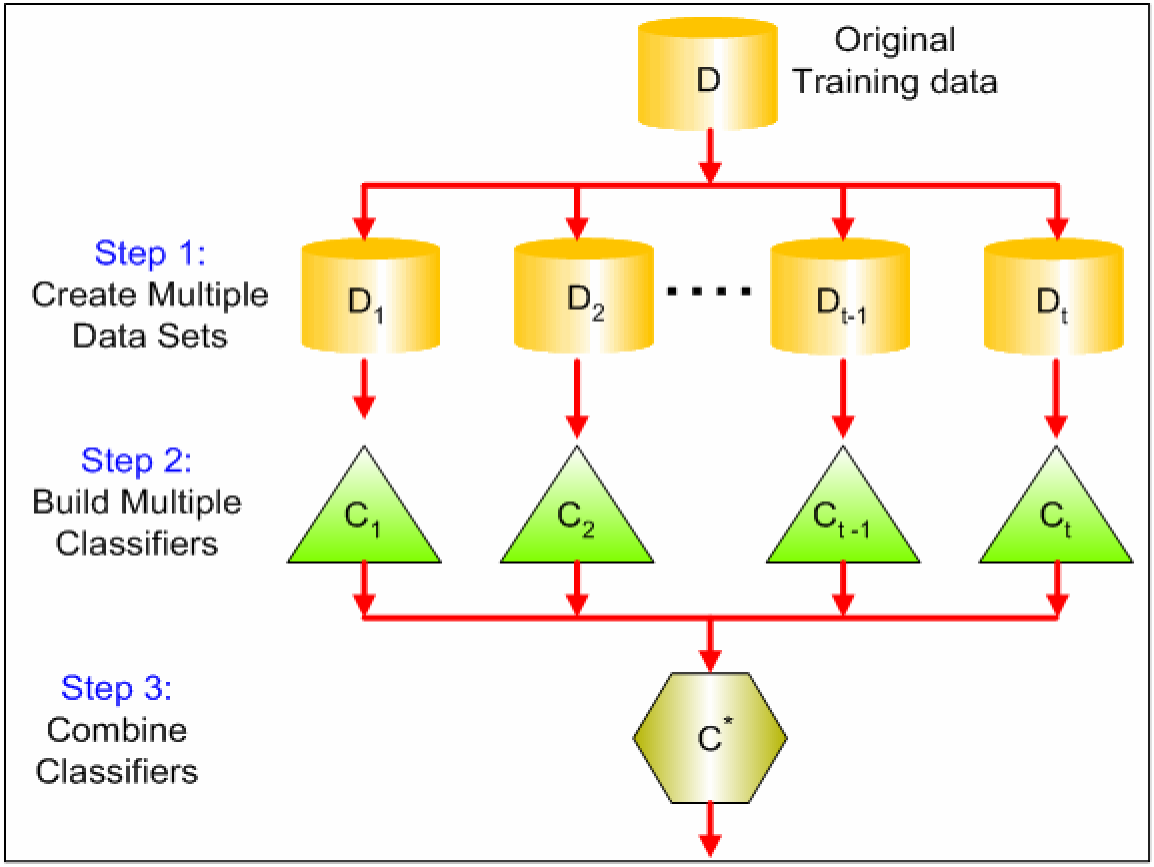
- Independence – in practice:
  - It is not possible to ensure total independence among the base classifiers
  - Good results have been achieved in ensemble methods when the base classifiers are slightly correlated

Methods for constructing ensembles

- Effective ensemble consists of base classifiers that are reasonably correct and also diverse (i.e. independent)
- Methods for creating ensembles focus on generating disagreement among the base classifiers by:
  - Manipulating the training data – creating multiple training sets by resampling the original data according to some sampling distribution and constructing a classifier for each training set (e.g. Bagging and Boosting)
  - Manipulating the attributes – using a subset of input features (e.g. Random Forest and Random Subspace)
  - Manipulating the class labels – will not be covered (e.g. error-correcting output coding)
  - Manipulating the learning algorithm – e.g. building a set of classifiers with different parameters

Manipulating training data - examples

- Creating multiple training sets from the original training set by sampling
- Examples: Bagging and Boosting



Bagging

Bagging

- Bagging is also called bootstrap aggregation
- A bootstrap sample - definition:
  - Given: a dataset D with n example (the original dataset)
  - Bootstrap sample D' from D: contains also n examples, randomly chosen from D with replacement (i.e. some examples from D will appear more than once in D', some will not appear at all)
- On average, 63% of the examples in D will also appear in D' as it can be shown that the probability to choose an example is $(1-1/n)^n$

Dataset with 10 examples:

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

- Create M bootstrap samples
- Use each sample to build a classifier
- To classify a new example: get the predictions of each classifier and combine them with a majority vote
  - i.e. the individual classifiers receive equal weights

Bagging - pseudocode

<u>Model generation</u>

Let *n* be the number of examples in the training data
For each of *M* iterations:
  Sample *n* examples with replacement from training data
  Apply the learning algorithm to the sample
  Store the resulting model

<u>Classification</u>

For each of the *M* models:
  Predict class of testing example using model
Return class that has been predicted most often (majority vote)

Bagging

- Typically performs significantly better than the single classifier and is never substantially worse
- Especially effective for unstable classifiers
- Unstable classifiers: small changes in the training set result in large changes in predictions, e.g. decision trees, neural networks are considered unstable classifiers

- Applying Bagging to regression tasks - the individual predictions are averaged

Boosting

Boosting – main idea

- The most widely used ensemble method
- Idea: Make the classifiers complement each other
- How: The next classifier should be created using examples that were difficult for the previous classifiers

- Many boosting algorithms have been proposed, the most popular are AdaBoost and Gradient Boosting

AdaBoost

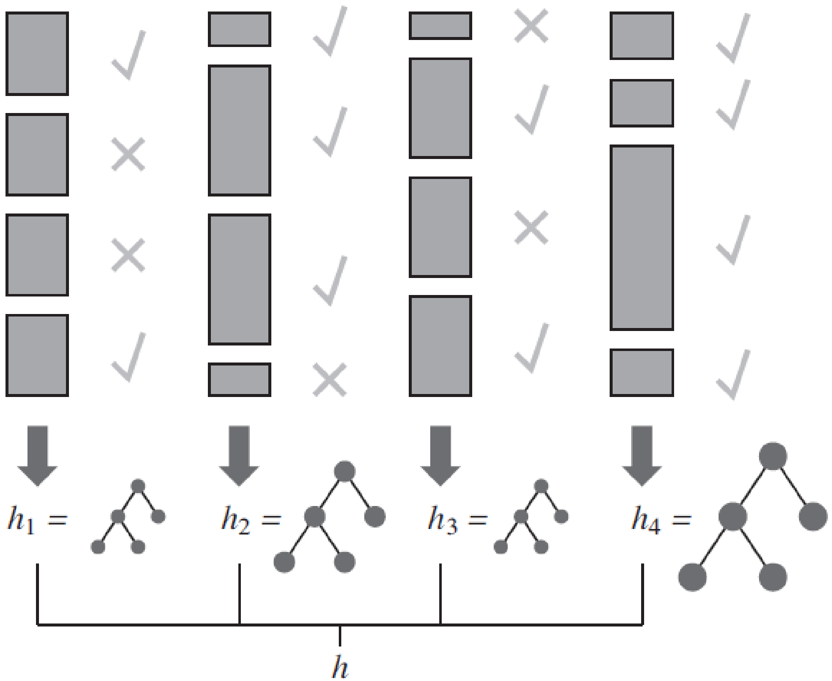
- Uses a *weighed training set*
- Each training example has an associated weight ($\geq$0)
- The higher the weight, the more difficult the example was to classify by the previous classifiers
- Examples with higher weight will have a higher chance to be selected in the training set for the next classifier

- AdaBoost was proposed by Freud and Shapire in 1996

Ada Boost – more details

- Initially all training examples have equal weights, e.g. 1/*m*, *m* is the number of training examples
- From this set, the first classifier (hypothesis) h1 is generated
- It classifies some of the training examples correctly and some incorrectly
- We would like the next classifier to learn to classify the misclassified examples, so we increase the weights of the misclassified examples and decrease the weights of the correctly classified examples
- There is a mechanism for selecting examples for the training set of the next classifier; the ones with higher weight are more likely to be selected
- From this new weighed set, we generate classifier h2
- Continue until M classifiers are generated
- Final ensemble: combine the M classifiers using a weighted vote based on how well the classifier performed on the training set

Diagram



Combining decision trees

- 1 rectangle = 1 training example
- height of the rectangle corresponds to the weight of the example
- √ and X - how the example was classified by the current classifier (correctly or incorrectly)
- size of the tree corresponds to the weight of its prediction in the ensemble

AdaBoost algorithm

D – given training set, m – number of training examples, M – number of models

<u>Model generation</u>

Assign equal weight p_i to each training example *i*, e.g. 1/m

p_i determines the probability of *i* to be selected in the training set for the next model

For k=1 to *M* iterations: //building M models

Create data subset D_k from D with *m* ex. by sampling with replacement using p

Apply learning algorithm to D_k and store resulting model

Compute error e_i of model on each training example *i*:

e_i = 0 if correctly classified, e_i =1 if incorrectly classified

Calculate the weighed error of the model $e = sum(p_i * e_i)$ over all *m* examples

Update the weights:

If example classified correctly by model, multiply its weight by e / (1 - e)

Normalize weights (probabilities) of all examples so that they sum to 1

k=k+1

This results in increasing the weights of the misclassified examples and decreasing the weights of the correctly classified

AdaBoost algorithm (2)

<u>Classification</u>

For each of the *M* models:
  Predict class of testing example using model
Combine predictions using weighed vote, where the weight of each model depends on its accuracy on training set used to build the model

AdaBoost theorem

- If the base learners are weak learners, then AdaBoost will return an ensemble that classifies the training data perfectly for a large enough number of iterations (learners combined) M
- Weak learner is a classifier whose classification performance is slightly better than random guessing (i.e. 50% for binary classification)
- Thus, AdaBoost boosts a set of weak learners into a strong learner
- => Boosting allows building a powerful combined classifier from very simple ones, e.g. 1-level decision trees

- More theoretical results – Boosting works well:
  - If base classifiers are not too complex
  - Their errors do not become too large too quickly as more iterations are done
  - The meaning of "too" is defined by Shapire et al. (1997)

Gradient Boosting

- Introduced by Leo Breiman in 1997:

  Arcing the Edge, https://statistics.berkeley.edu/tech-reports/486

- Further developed by Jerome Friedman in 1999:

  Greedy function approximation: A gradient boosting machine.

- https://projecteuclid.org/euclid.aos/1013203451
- Uses decision trees as base learners, typically shallow (weak) trees
- As AdaBoost, it works by sequentially adding base learners to the ensemble, each one focusing on the examples that were difficult to classify by the previous base learner
- However, while AdaBoost updates the weights of the examples at each iteration, Gradient Boosting adds a new model that minimizes the error of the previous model

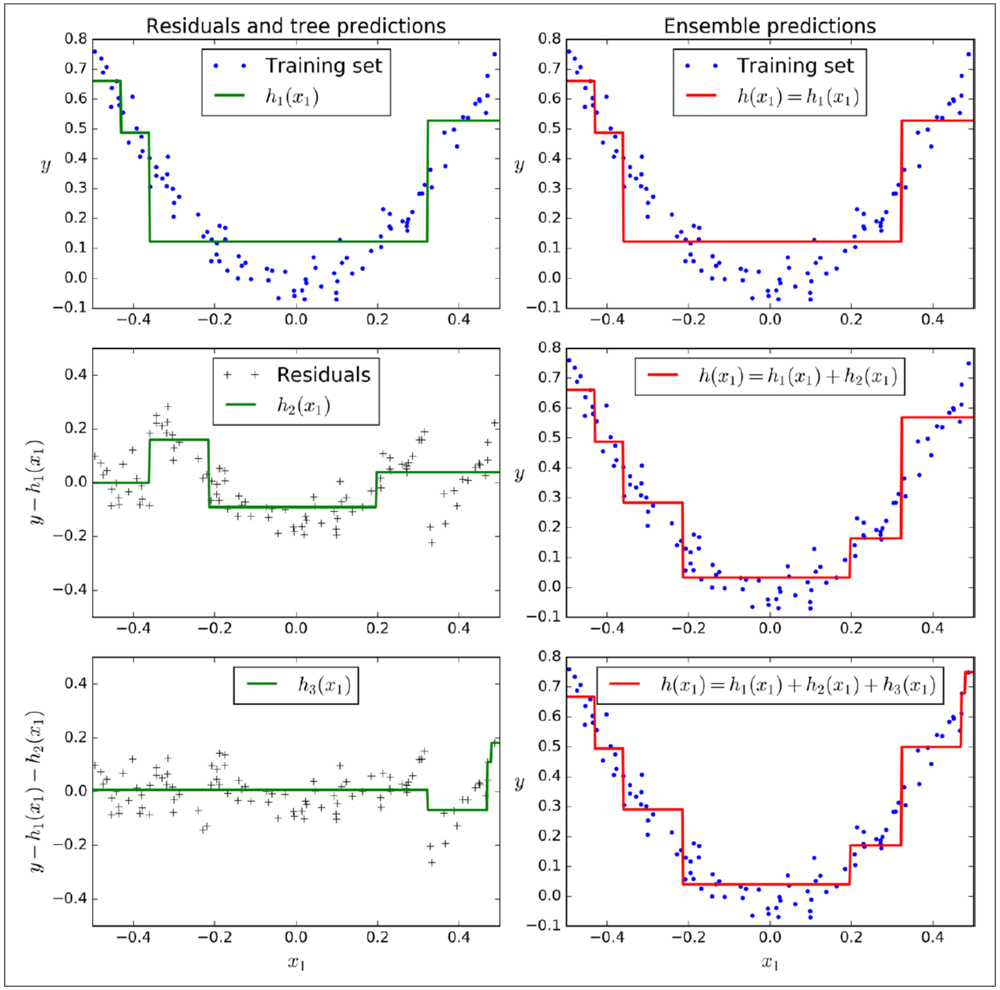
Gradient Boosting - example

- Example for regression tasks, see Géron textbook p.203
- Create model 1: DT1 fit on training data (X,y), store model
- To create model 2:
  - Evaluate DT1 on training data, calculate error:

    y2 = y (actual value) - predicted value by DT1
  - Create model 2: DT2 fit on (X,y2), store model
- To create model 3:
  - Evaluate DT2 on training data, calculate error:

    y3 = y2 - predicted value by DT2
  - Create model 3: DT3 fit on (X,y3), store model
- Now we have 3 decision trees. To make a prediction for a new example: sum the predictions of DT1, DT2 and DT3

Gradient Boosting - example



The ensemble performance improves as more trees are added to it

Bagging and Boosting - comparison

- Similarities
  - Use voting (for classification) and averaging (for prediction) to combine the outputs of the individual learners
  - Combine classifiers of the same type, typically trees – e.g. decision stumps or decision trees
- Differences
  - Creating base classifiers:
    - Bagging – separately
    - Boosting – iteratively – the new ones are encouraged to become experts for the misclassified examples by the previous base learners (complementary expertise)
  - Combination method
    - Bagging – equal weighs to all base learners
    - Boosting – different weights - based on performance on training data

Random Forest

Creating ensembles by manipulating the attributes

- Each base classifier uses only a subset of the features
- E.g. training data with K features, create an ensemble of M classifiers each using a smaller number of features L (L<K)

  1) Create feature subsets by random selection from the original feature set => creating multiple versions of the training data, each containing only the selected features

- 2) Build a classifier for each version of the training data
- 4) Combine predictions with majority vote
- Example: Random Forest
  - Combines decision trees
  - Uses 1) bagging + 2) subset of features (during decision tree building, when selecting the most important attribute)

Random Forest algorithm

n - number of training examples, *m* – number of all features, *k* – number of features to be used by each ensemble member (k<m), *M* – number of ensemble members

Model generation:

For each of *M* iteration

1. Bagging – generate a bootstrap sample

   Sample *n* instances with replacement from training data

2. Random feature selection for selecting the best attribute

   Grow decision tree without pruning. At each step select the best feature to split on by considering only *k* randomly selected features and calculating information gain

Classification:

Apply the new example to each of the *t* decision trees starting from the root. Assign it to the class corresponding to the leaf. Combine the decisions of the individual trees by majority voting.

Comments on Random Forests

- Performance depends on
  - Accuracy of the individual trees (strength of the trees)
  - Correlation between the trees
- Ideally: accurate individual trees but less correlated
- Bagging and random feature selection are used to generate diversity and reduce the correlation between the trees
- As the number of features k increases, both the strength and correlation increase
- Random Forest typically outperforms a single decision tree
- Robust to overfitting
- Fast as only a subset of the features are considered

Ensembles - summary

- Ensembles combine the predictions of several classifiers
- They work when the individual classifiers are accurate and diverse
- Diversity is generated by manipulating the
  - training data (Bagging, Boosting)
  - attributes (Random Forest = bagging + random selection of attributes)
  - learning algorithm
- Some ensembles combine classifiers of the same type, some not
- Most ensembles use a majority vote to make predictions on new data, others used a weighted vote
- Ensembles have shown excellent performance – often the winning solution in ML competitions is an ensemble method