Ensemble Learning

CMT307 Session 7

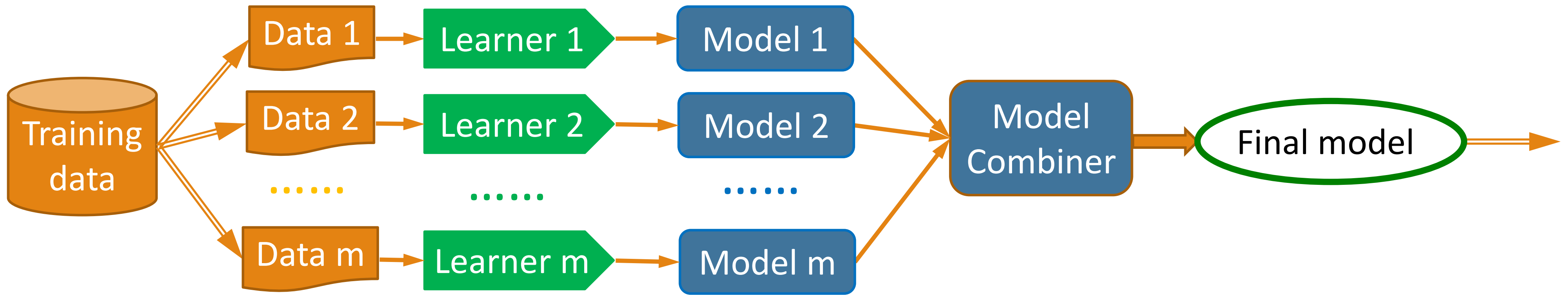
Yuhua Li

Ensemble Learning

- A model is specified by its parameters, a set of different parameters gives a different model.
- So far – learning methods that learn a single model, chosen from a model space that is used to make predictions.
- Ensemble learning →
  - A group of models/predictors is called an *ensemble*;
  - select a collection (ensemble) of models and combine their predictions.
- Example 1 - generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.
- Key motivation: reduce the error rate. Hope is that it will become much more unlikely that the ensemble of models will misclassify an example.

Learning Ensembles

- Learn multiple alternative models for a given problem using different training data or different learning algorithms.



- Combine decisions of multiple learned models, e.g. using weighted voting.

Value of Ensembles

- "No Free Lunch" Theorem
  - No single algorithm wins all the time!

- When combing multiple independent and diverse decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.

- Examples: Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who wins the general election: citizens vote.

Example: Weather Forecast

Reality	⛈	☀	☀	⛈	⛈	☀	☀
1	⛈	⛈ ✗	☀	☀ ✗	⛈	☀	⛈ ✗
2	☀ ✗	☀	☀	☀ ✗	⛈	☀	⛈ ✗
3	⛈	☀	⛈ ✗	⛈	☀ ✗	⛈ ✗	☀
4	⛈	☀	⛈ ✗	⛈	☀ ✗	☀	☀
5	⛈	⛈ ✗	☀	⛈	⛈	⛈ ✗	☀
Combine	⛈	☀	☀	⛈	⛈	☀	☀

Intuitions

- Majority vote

  Suppose we have 5 completely independent classifiers…

  - If accuracy is 70% for each

$$\frac{5!}{5!\,(5-5)!}\,0.7^5\,(1-0.7)^{5-5}+\frac{5!}{4!\,(5-4)!}\,0.7^4\,(1-0.7)^{5-4}+\frac{5!}{3!\,(5-3)!}\,0.7^3\,(1-0.7)^{5-3}$$

$$=0.7^5+5*0.7^4*0.3+10*0.7^3*0.3^2$$

83.7% majority vote accuracy

  - 101 such classifiers
    - **99.9% majority vote accuracy**

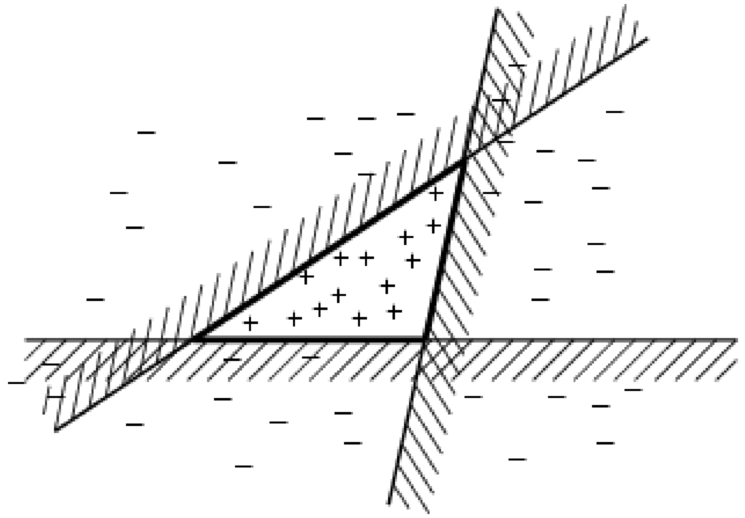
- This is only true if all classifiers are perfectly independent, making uncorrelated errors, which is clearly not the case since they are trained on the same data or the same type of learners, etc.

- **Note: Binomial Distribution:** The probability of observing *x* heads in a sample of *n* independent coin tosses, where in each toss the probability of heads is *p*, is

$$P(X=x|p,n)=\frac{n!}{x!\,(n-x)!}\,p^x\,(1-p)^{n-x}$$

Ensemble Learning

- Another way of thinking about ensemble learning:
- → way of enlarging the hypothesis space, i.e., the ensemble itself is a hypothesis and the new hypothesis space is the set of all possible ensembles constructible form hypotheses of the original space.



Increasing power of ensemble learning:

Three linear threshold hypothesis
(positive examples on the non-shaded side);
Ensemble classifies as positive any example classified positively be all three. The resulting triangular region hypothesis is not expressible in the original hypothesis space.

Different Learners

- Different learning algorithms
- Algorithms with different choice for parameters
- Data set with different features
- Data set with different subsets

Ensemble methods work best when the predictors are as independent from one another as possible:,

diversifying predictors as much as possible.

This increases the chance that they will make very different types of errors, improving the ensemble's accuracy.

Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m

- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data

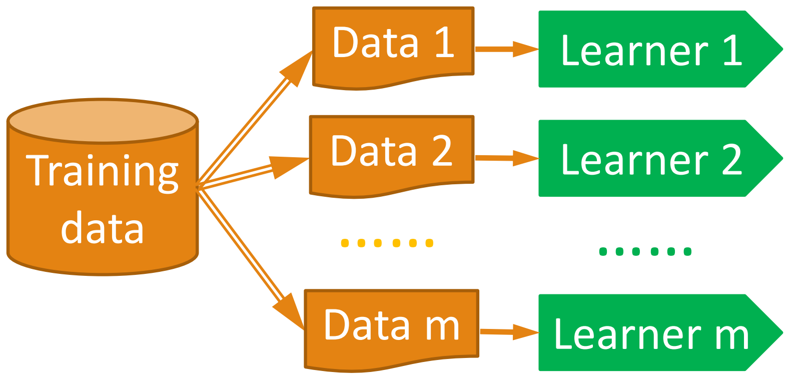
- In some ML software such as WEKA, these are called ***meta-learners***, they take a learning algorithm as an argument (***base learner***) and create a new learning algorithm.

Ensemble methods

- Bagging
- Boosting
- Stacking

Bagging and Pasting

- When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating).
  - training instances may be sampled several times
- When sampling is performed without replacement, it is called pasting.



Bagging

- Create ensembles by "*bootstrap aggregation*", i.e., repeatedly randomly resampling the training data (Brieman, 1996).

- Bootstrap: draw *n* items from the training dataset D with replacement

- Bagging
  - Train *M* learners on *M* bootstrap samples
  - Combine outputs by voting (e.g., majority vote)

- Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees and neural networks) whose output can change dramatically when the training data is slightly changed.

Bagging - Bootstrap aggregating

- Given a standard training set D of size n

- For $i = 1 .. M$
  - Draw a sample of size $n^* < n$ from D uniformly and with replacement
  - Learn classifier C_i
- Final classifier is a vote of $C_1 .. C_M$
- Increases classifier stability/reduces variance
- Bagging can be made parallel, so it scales well.

Bagging (Bootstrap aggregating)

- Take M bootstrap samples (with replacement)
- Train M different classifiers on these bootstrap samples
- For a new query, let all classifiers predict and take an average (or majority vote)
- If the classifiers make independent errors, then their ensemble can improve performance.
- Stated differently: the variance in the prediction is reduced (we don’t suffer from the random errors that a single classifier is bound to make).

Boosting

- Boosting refers to any Ensemble method that can combine several weak learners into a strong learner.
- Train predictors sequentially, each trying to correct its predecessor.
- Many boosting algorithms, most based on decision trees
  - Adboost
  - GBRT
  - XGBoost
  - LightGBM
  - …

Strong and Weak Learners

- Strong Learner →Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily accurate*

- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is more accurate than random guessing

Boosting

- Weak Learner: only needs to generate a hypothesis with a training accuracy greater than 0.5, i.e., < 50% error over any distribution

- Learners
    - Strong learners are very difficult to construct
    - Constructing weaker Learners is relatively easy

- Questions: Can a set of **weak learners** create a single **strong learner** ?
- YES ☺

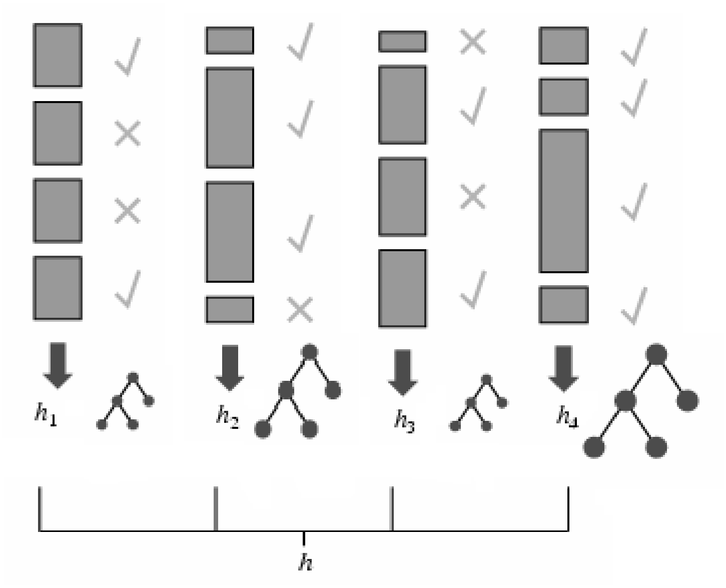
Boost weak classifiers to a strong learner

Boosting

- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than 0.5 (Schapire, 1990).
- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).
- Key Insights
  - Instead of sampling (as in bagging) re-weigh examples!
  - Examples are given weights. At each iteration, a new hypothesis is learned (weak learner) and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.
  - Final classification based on weighted vote of weak classifiers

Adaptive Boosting

- Each rectangle corresponds to an example, with weight proportional to its height.
- Crosses correspond to misclassified examples.
- Size of decision tree indicates the weight of that hypothesis in the final ensemble.



Construct Weak Classifiers

- Using Different Data Distribution
  - Start with uniform weighting
  - During each step of learning
    - Increase weights of the examples which are not correctly learned by the weak learner
    - Decrease weights of the examples which are correctly learned by the weak learner
- Idea
  - Focus on difficult examples which are not correctly classified in the previous steps

Combine Weak Classifiers

- Weighted Voting
    - Construct strong classifier by weighted voting of the weak classifiers
- Idea
    - Better weak classifier gets a larger weight
    - Iteratively add weak classifiers
        - Increase accuracy of the combined classifier through minimization of a cost function

Adaptive Boosting: High Level Description

```
C =0; /* counter*/
M = m; /* number of hypotheses to generate*/

1 Set same weight for all the examples  (typically each example has weight = 1/N, N is the size of data);

2 While (C < M)
        2.1 Increase counter C by 1.
        2.2 Generate hypothesis  $h_C$ .
        2.3 Increase the weight of the misclassified examples in  hypothesis $h_C$
3 Weighted majority combination of all M hypotheses (weights according to how well it performed on the training set).
```

Many variants depending on how to set the weights and how to combine the hypotheses. ADABOOST → quite popular!!!!

Performance of Adaboost

- Learner = Hypothesis = Classifier

- Weak Learner: < 50% error over any distribution

- M number of hypothesis in the ensemble.

- If the input learning is a Weak Learner, then ADABOOST will return a hypothesis that classifies the training data perfectly for a large enough M, boosting the accuracy of the original learning algorithm on the training data.

- Strong Classifier: thresholded linear combination of weak learner outputs.

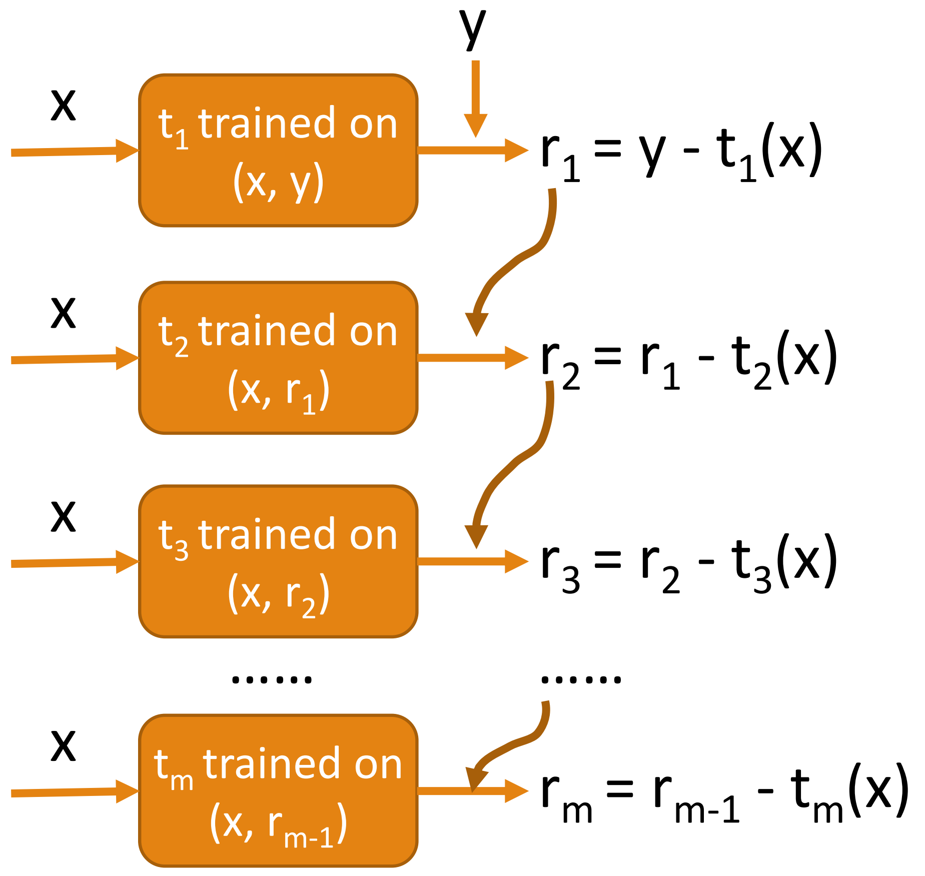
AdaBoost Summary

- Train classifiers (e.g. decision trees) in a sequence.
- A new classifier should focus on those cases which were incorrectly classified in the last round.
- Combine the classifiers by letting them vote on the final prediction (like bagging).
- Each classifier is “weak” but the ensemble is “strong.”

Gradient Boosting

- Like AdaBoost, Gradient Boosting (GB) works by sequentially adding predictors to an ensemble, each one correcting its predecessor
- Unlike AdaBoost to tweak the instance weights at every iteration, GB tries to fit the new predictor to the residual errors made by the previous predictor

- Gradient Boosted Regression Trees (GBRT)

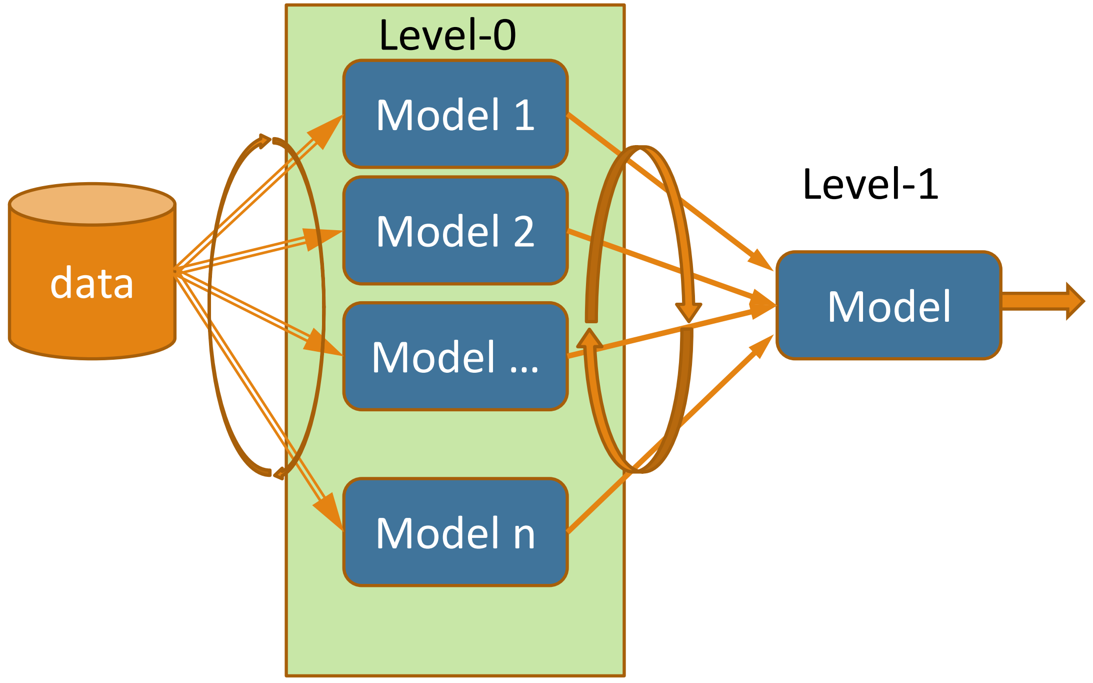


Final prediction: $\hat{y} = \sum_i t_i$

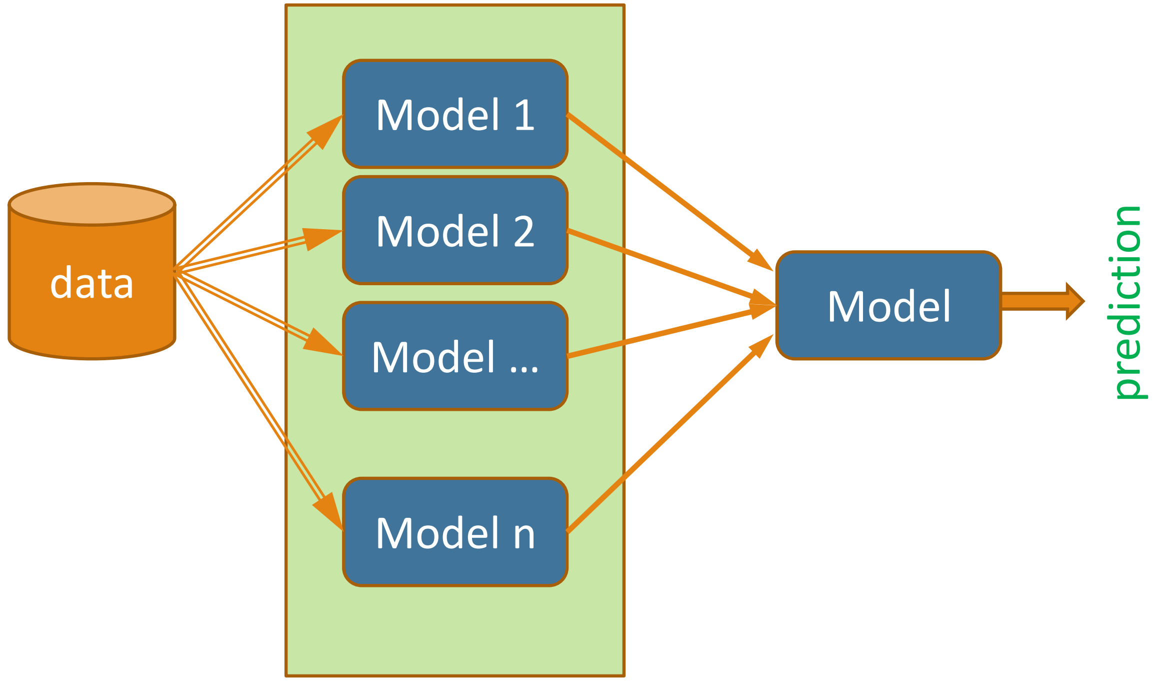
Stacking

- Training
  - Train different models on the same data (level-0 models)
  - Train a new ("level-1") model with the outputs of the level-1 models

- Prediction
  - Given a new unlabelled data sample, input it to each level-0 model
  - The ensemble prediction is the level-1 model prediction based on level-0 predictions



Use different parameters and/or different ML techniques



Random Forest

- Motivation: reduce error correlation between classifiers
- Main idea: build a larger number of un-pruned decision trees
- Key: using a random selection of features to split on at each node

Random Forest

- Each tree is grown on a bootstrap sample of the training set of ***N*** cases.
  - A number *m* is specified much smaller than the total number of features *M* (e.g. *m* = sqrt(*M*)).
  - At each node, *m* variables are selected at random out of the *M*.
  - The split used is the best split on these *m* variables.

- Ensemble consisting of a bagging of un-pruned decision tree learners with a randomized selection of features at each split.

- Grow many trees on datasets sampled from the original dataset with replacement (a bootstrap sample).
  - Draw K bootstrap samples of a fixed size
  - Grow a DT, randomly sampling a few attributes/dimensions to split on at each internal node

- Average the predictions of the trees for a new query (or take majority vote)

Advantages of random forest

- Error rates compare favorably to Adaboost
- More robust with respect to noise.
- More efficient on large data
- Provides an estimation of the importance of features in determining classification

Reading

- Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems. O'Reilly, 2017. **Chapter 7**.