CS 60050
Machine Learning

Ensemble Learning

Some slides taken from course materials of Tan, Steinbach, Kumar

Basic idea

- Concordet's jury theorem (1785):
  - Imagine that a group of people has to select between two choices (from which only one is correct).
  - They vote independently, and the probability that each of them votes correctly is p.
  - The votes are combined by the majority rule.
  - Let m denote the probability that the majority vote is correct.
  - If $p>0.5$ then $m \rightarrow 1$ as the number of votes goes to infinity

Basic idea

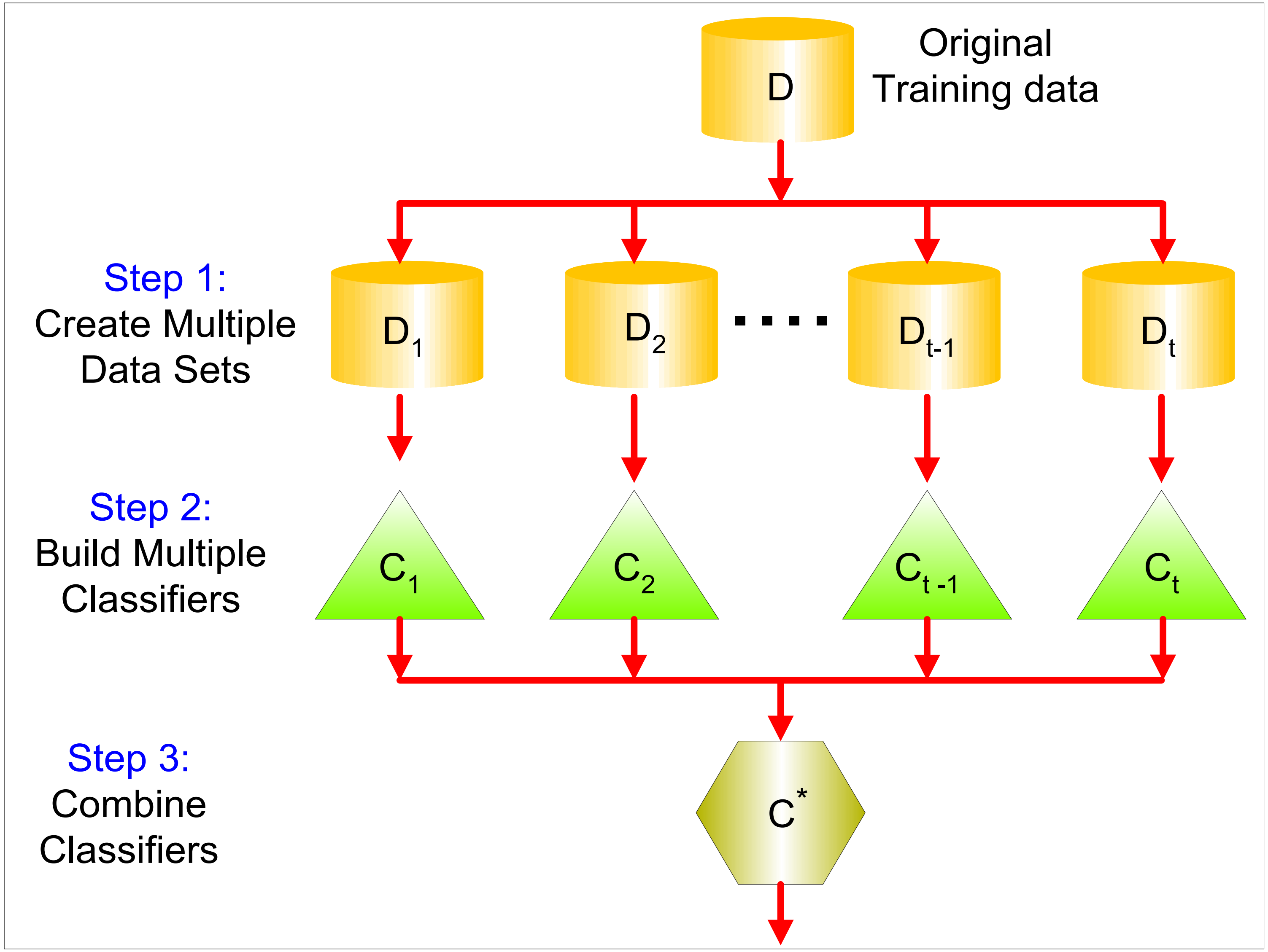
- So the crowd is more clever than the individuals under some assumptions
  - Each individual must be correct with p>0.5 (better than random guessing)
  - Their should make independent decisions

- How can we apply this idea in machine learning?

Strong vs. weak learners

- Strong learner: we seek to produce one classifier for which the classification error can be made arbitrarily small
- Weak learner: a classifier which is just better than random guessing
- Ensemble learning: instead of creating one strong classifier, we create a large set of weak classifiers, then we combine their outputs into one final decision
  - According to Concordet's theorem, under proper conditions the ensemble model can attain an error rate that is close to zero
  - While creating a lot of weak classifiers is hopefully a much easier task than to create one strong classifier

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent, i.e., their errors are uncorrelated
- Ensemble classifier: majority vote on the predictions made by the base classifiers
  - Probability that the ensemble classifier makes a wrong prediction:

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

Conditions for effective ensemble

- Two conditions for an ensemble classifier to perform better than a single classifier:

  – The base classifiers should be independent of each other (in practice, ensemble works even when base classifiers are slightly correlated)

  – The base classifiers should perform better than a classifier that performs random guessing

How to produce diverse classifiers?

- We can combine *different learning algorithms* (“hybridization”)
  - E.g. train a Neural Network, an SVM, a k-NN,… over the same data, and then combine their output

- We can combine the same learning algorithm trained several times over the same data
  - Works only if there is some random factor in the training method
  - E.g. neural networks trained with *different random initialization*

- We can combine the same learning algorithm trained over *different subsets of the training data*
  - We can also try using *different subsets of the features*

Types of ensemble methods

- Manipulate the training instances
  - Create multiple training sets by re-sampling original data by some sampling distribution
  - Build one classifier from each training set
  - Examples: Bagging, Boosting

- Manipulate input features
  - Choose a subset of input features to form each training set, either randomly or based on domain expertise
  - Build one classifier from each training set
  - Example: Random Forest

Types of ensemble methods

- Manipulate the learning algorithm
  - Some learning algorithms can give models that vary based on parameter settings, even when trained on same training data
  - Train different models on same training data, and consider ensembles of the different models
  - Example: decision trees can give various models if randomness is introduced in tree growing process

- Hybrid methods (combinations of the above types of methods) can also be used

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?

- We will focus on
  - Bagging
  - Boosting

Bagging

Bagging

- Bagging = **B**ootstrap + **agg**regat**ing**
- Sampling with replacement to generate different training sets from the original training set

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

- Build classifier on each bootstrap sample
- Classification - aggregate the base learners by taking their average (using uniform weights for each classifiers), or by majority voting

Bagging: bootstrap resampling

- Suppose we have a training set with *n* records

- Bootstrap resampling takes random samples from the original set with replacement
  - Randomness required to obtain different training sets for different rounds of resampling
  - Replacement required to create training sets of size *n* from the original data set of size *n*

Bagging algorithm

Model generation

```
Let n be the number of instances in the training data
For each of t iterations:
      Sample n instances from training set
             (with replacement)
      Apply learning algorithm to the sample
      Store resulting model
```

Classification

```
For each of the t models:
      Predict class of instance using model
Return class that is predicted most often
```

When is bagging effective?

- The ensemble model is almost always better than the base learners if the base learners are unstable
  - Unstable learners: a small change in the training data may cause a large change in the learnt model
  - Unstable: Neural networks, Decision Trees
  - Stable: SVM, k nearest neighbor
- Bagging with stable learners not a good idea

Boosting

Boosting: basic idea

- An iterative procedure that proceeds in rounds
  - Generate a series of base learners which complement each other
  - For this, we will force each learner to focus on the mistakes of the previous learner

- Adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all *n* records are assigned equal weights
  - Unlike bagging, weights given to records may change at the end of each boosting round

Boosting: resampling

- Records that are wrongly classified will have their weights increased for next round
- Records that are classified correctly will have their weights decreased for next round

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

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Bootstrap: learning the ensemble

- In each round
  - Sample a training set considering the weights assigned to each record
  - Train learner on the sampled training set
  - Apply learner on the whole training set
  - Note performance of learner, and which records correctly / wrongly classified
  - Adjust weights of records for next round

- If performance of learner in a certain round is too bad, certain special steps can be taken

Bootstrap: aggregation step

- Combine decisions of base learners obtained in different rounds

- Boosting attempts to make the aggregation process more intelligent:
  - Aggregate the base learners using weighted voting
  - Importance / weight of base learners: The learners which had better performance will get a larger weight than those whose performance was not good

Implementations of boosting

- Many implementations of boosting, differing in:
  - How weights of records are updated after every round
  - How importance of base learners is measured
  - How the decisions of base learners are aggregated

- A popular implementation: AdaBoost (Adaptive Boosting)

AdaBoost

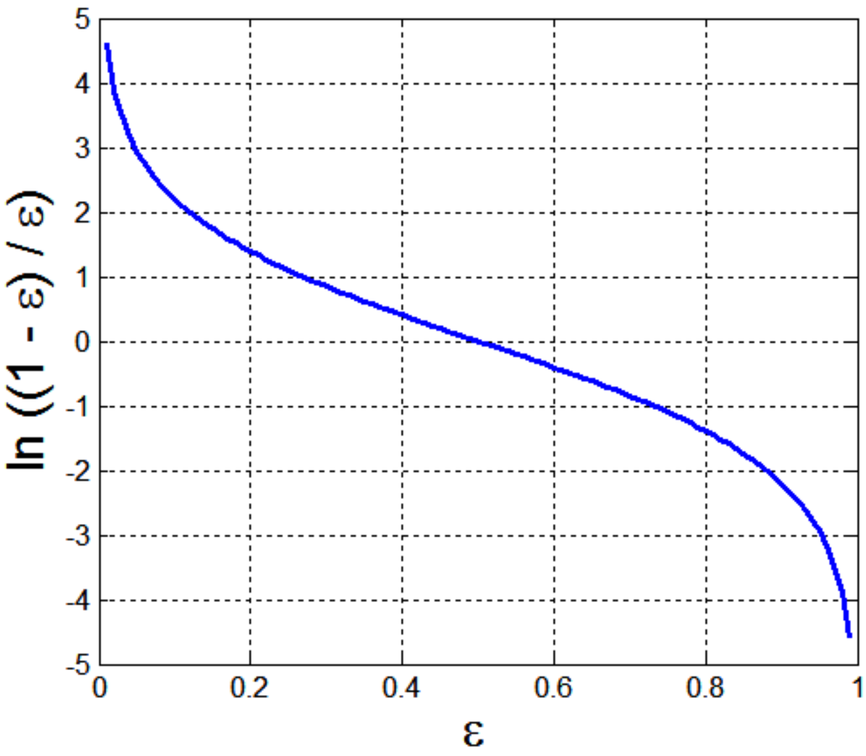
- Base classifiers: $C_1, C_2, \ldots, C_T$

- Error rate of classifier C_i where w_j is weight of record j:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of classifier C_i:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



AdaBoost

- Weight update - how weight of w_i is updated for round j+1, after round j:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

Weight decreased if classification correct

Weight increased if classification incorrect

- Normalization such that sum of all weights in a particular round equals 1
- If any intermediate round produces error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated

AdaBoost

- Classification:

$$C^*(x) = \arg\max_{y} \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

- Choose that class y which maximizes the weighted vote
- Learners C_j weighted according to importance α_j

Bagging vs. Boosting

Bagging vs. Boosting

- Both are usually effective in learning ensemble classifiers that are better than base classifiers
- On average, boosting results in better classification accuracy than bagging
- But boosting is particularly subject to overfitting if training set has significant amount of noise
  - Bagging gives equal weightage to all records
  - Boosting gives higher weightage to those records that are difficult to classify (which may be noise)
- Bagging is easy to parallelize, but boosting is not