This is lecture 2 of 2 in a series on collaborative filtering and ensemble methods

What is an ensemble

- A suite of simple methods
  - “Rules of thumb”
  - Easier than finding a single advanced algorithm
  - The power of many
  - Remember, collective group intelligence is normally higher than the best individual
- Machine learning algorithms that construct a set of classifiers
  - Classification/prediction based on voting

A suite of simple methods makes up an ensemble. Basically, finding simple methods or rules of thumb can be a lot easier than finding a single, advanced algorithm for accurate predictions and classifications. Remember the previous lecture, where we looked at the wisdom of crowds, and the power of many. We saw that in many cases, the combined intelligence of a group is higher than the best individual. The same principle applies here, as ensembles are machine learning algorithms that construct a set of classifiers, and their individual predictions are combined through a voting system.

Multiple methods

- Bayesian averaging
- Bagging
- Random forest
- Boosting
  - Gradient boosting
  - AdaBoost

There are many ways to create an ensemble, and we will take a look at some of these in the next slides. Some common methods include Bayesian averaging, bagging, random forest, and boosting.

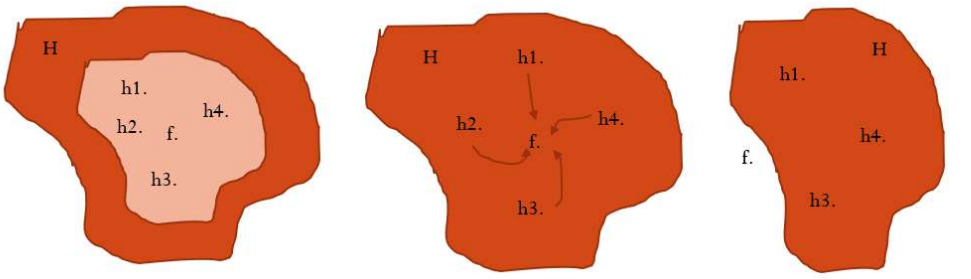
Repetition: classifiers and learners

- Perceptrons
- K-classifiers
- Decision trees
- Regressions
- Simple Bayesian methods
- ++

Now for a quick repetition on classifiers and learners. You remember from previous that we have classes like perceptrons, K-classifiers, decision trees, regressions and simple Bayesian, probability based methods. These are what we call individual classifiers.

Strengths of an ensemble

- Why an ensemble may work better than a single classifier



Statistical Computational Representational

- *H is the set of all classifiers*

Just as with people, combining individual classifiers into an ensemble can be beneficial in three ways: Statistical, computational, and representational. Statistically speaking, some classifiers are more likely to give better predictions from the start by random chance, and on average, the combined output will be closer to the actual value. Similarly, dividing the load on multiple simple classifiers can speed up execution time, compared to one big, heavy algorithm. It is also easier to represent complex problems if we split them up into smaller parts for each classifier in our ensemble.

The origin of ensemble methods

- Roots in crowdsourcing
  - Committees, majority vote and unweighted averages
- Simple ensembles
  - Simple average
  - Simple combination of learners/classifiers
- Example: Committee approach:

$$\hat{y} = \frac{1}{N_h} \sum_{hi} \sigma i * hi(x),\ \sigma i \in [0,1]$$

The concept of ensemble methods has its origin in crowdsourcing. For example, in the real world, we have committees, majority voting and unweighted averages. Like the example from the previous lecture, when guessing an unknown number, we assume that every individual has the same probability of guessing the correct answer.

It is possible to simply take the average prediction of every classifier, or do some other basic combination. For example, using a committee approach, we simply sum up every prediction and take its average. As we have

seen, this usually leads to higher accuracy, but we can do even better. With classifiers in ensembles, we usually have some idea of the performance of each model. More about this later.

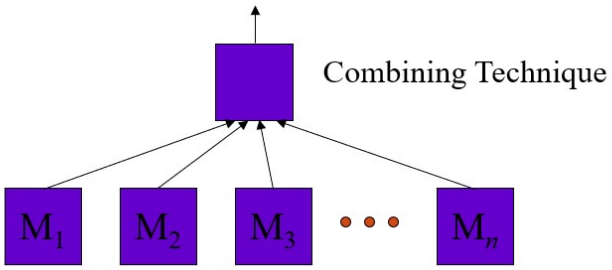
Ensembles using simple statistics

- Multiple diverse models trained on the same problem
  - Outputs are combined to form the final output
  - Overfit can be average out
  - Diverse models can be accurate even if the individuals are weak generalizers

Another benefit of splitting the dataset and combining classifier output is that we can have multiple diverse models trained on the same problem. Their outputs are combined to come up with a final output. This is good because the specific overfit of each learning model can be averaged out. Also, if models are diverse (meaning that we assume uncorrelated errors) then even if the individual models are weak generalizers, the ensemble can be very accurate

Combining outputs

- Many different Ensemble approaches:
  - Stacking, Gating/Mixture of Experts, Bagging, Boosting, Wagging, Mimicking, Heuristic Weighted Voting, Combinations



As we saw earlier, we have many different ensemble approaches, or ways of combining outputs.

Bootstrap Aggregation (Bagging)

- A way to improve machine learning by splitting the data set
  - improves overall accuracy by decreasing variance
- Often used with the same learning algorithm
  - Best for diverse hypotheses based on initial conditions
- A set of *m* learners
  - Same initial parameters
  - Each training set chosen uniformly at random (2/3)
- Does not overfit (compared to boosting)
  - May be more conservative on accuracy improvements

Let's start by taking a look at Bootstrap Aggregation, or Bagging. Basically what we do, is split the dataset in order to improve the accuracy of a machine learning technique. This is achieved due to decreased variance. This approach is often used with the same learning algorithm and thus best for those which tend to give more diverse hypotheses based on initial conditions. This is done by inducing a set of M learners starting with the same initial parameters with each training set chosen uniformly at random with replacement from the original data set. These training sets might be 2/3rds of the

dataset, as we still need to save some separate data for testing. We also give all M hypotheses an equal vote for classifying novel instances. As we have previously discussed, this results in consistent empirical improvement.

One major advantage of bagging compared to boosting is that it does not overfit. However, bagging may also be more conservative overall on accuracy improvements.

It is also possible to use other schemes to improve diversity between learners, like different initial parameters, sampling approaches etc. We could even use different learning algorithms for each classifier. In short – the more diversity the better. Most often bagging is simply implemented with the same learning algorithm and just different training sets.

More advantages of bagging

- Bias
  - Error due to model choice
  - We only see «part of the world»
- Variance
  - Randomness due to data size
- Bias and Variance are expectations:
  - Averaging helps reduce outliers

Further advantages of bagging, and ensemble methods in general, are that we reduce bias. Bias is usually caused due to model choice, and hidden data. In short, we only see part of the whole image. An ensemble of classifiers trained on a diverse dataset are therefore less likely to have bias. Similarly, if each classifier has a high variance (unstable) the aggregated classifier has a smaller variance than each single classifier. The bagging classifier is like an approximation of the true average computed by replacing the probability distribution with bootstrap approximation.

Boosting

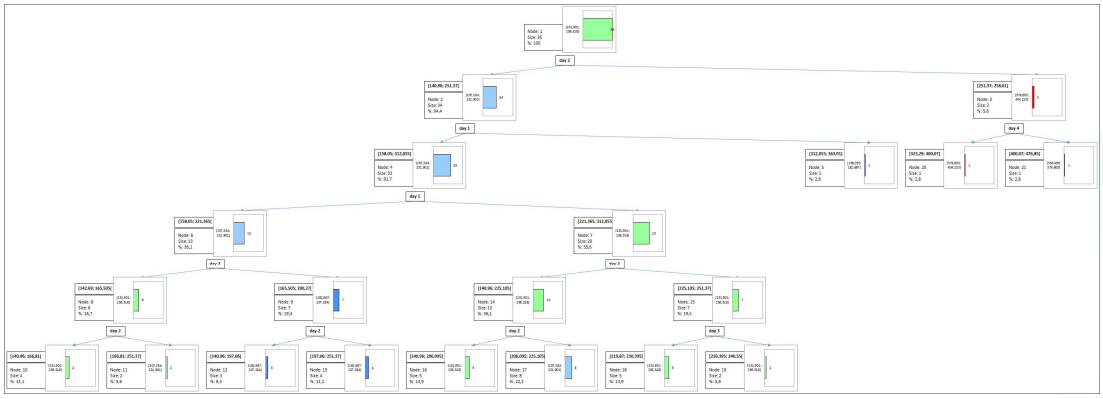
- All models have a weighted vote
  - Vote is scaled based on the model's accuracy
- More «aggressive» than bagging
  - Better than bagging on average
  - May overfit and do worse
    - Could theoretically converge to the training set
  - May be worse than non-ensemble in rare cases
- Many variations
  - Gradient boost, boosted regression tree, AdaBoost etc.

Finally, we will take a look at boosting. As with bagging, all models have a vote, but we don't simply take the average. As I mentioned earlier, we do have some idea of the performance of each individual classifier in our ensemble. We can use this in order to scale each model's vote based on its accuracy on the training set it was trained on. This means that boosting is a more aggressive approach than bagging, and may also be prone to overfitting and worse performance in some cases. In theory, using boosting may even make the ensemble converge to the training set. This may even

lead to worse performance than non-ensemble methods in some rare cases. However, on average, boosting is still better than bagging for most problems.

There are many variations, some examples include gradient boost, boosted regression trees, and AdaBoost.

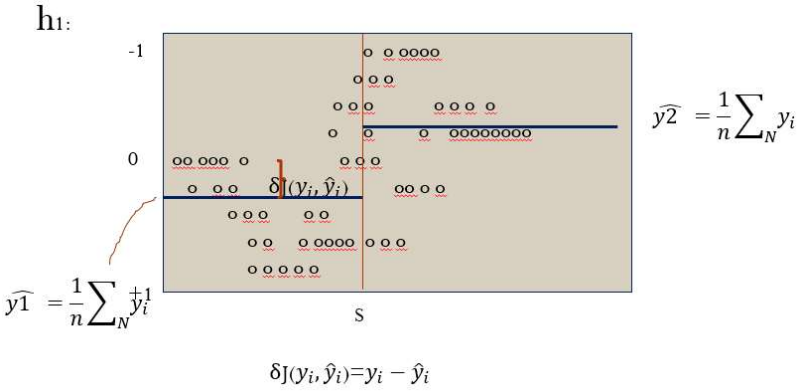
Regression tree



To close off, we will take a quick look at a boosted regression tree. In order to create an accurate prediction, a single tree can become quite unmanageable, like this example for predicting electricity prices:

Example: Boosted regression tree

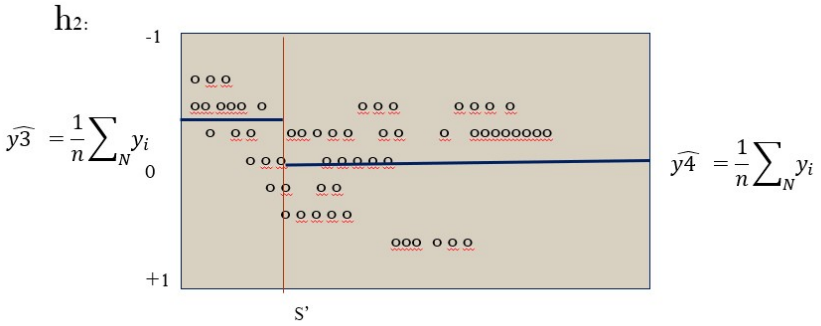
- The first simple learner h1:



To simply show the concept, let's say we have some points we want to classify. We create a simple learner called h1 with two classes: y1 and y2 divided by the point S.

Example: Boosted regression tree

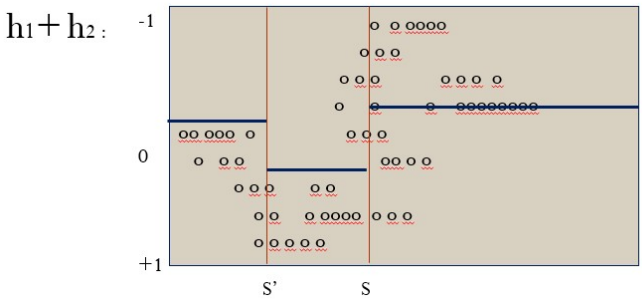
- The second simple learner h2:



We do the same with another tree h2, but we choose a different split point S' as so:

Simple learners combined

- We see that combining the trees gives a more accurate classifier



Finally, we combine the outputs from the two trees, and we get the following: Each tree is still a simple yes/no, but with their outputs combined we get a more accurate approximation of our points.