Ensembles

Ensemble methods are techniques that combine multiple machine learning models to improve overall performance. The main idea is that by aggregating the predictions of several models, the ensemble can achieve better accuracy and robustness than any individual model.

With ensemble learning, we would apply multiple datasets to multiple models and then combine their outputs to make a final prediction. **Note:** There is no guarantee that an ensemble will always outperform individual models. To maximimize gain (and the likelihood of gain), the individual models should be diverse and make uncorrelated errors. We should also make sure that they are accurate individually.

Voting

Suppose each classifier in the ensemble outputs a probability that a given instance belongs to a certain class.

Hard voting

In hard voting, each classifier casts a vote for the class with the highest predicted probability, and the class with the most votes is selected as the final prediction.

Soft Voting

In soft voting, the predicted probabilities from each classifier are averaged (or weighted) to obtain a final probability for each class. The class with the highest average probability is then chosen as the final prediction. We could also use weighted mean, median, minimum, maximum, product, etc.

Reason for median: If one model is way off, it won't affect the final prediction as much as the mean would. Reason for maximum: If one model is very confident about a class, we might want to trust that confidence. Reason for minimum: If one model is very unconfident about a class, we might want to avoid that class.

Bagging

Bagging is an ensemble technique that involves training multiple instances of the same model on different subsets of the training data. Each subset is created by randomly sampling the original dataset with replacement (bootstrap sampling). The final prediction is made by aggregating the predictions of all models, typically through majority voting for classification or averaging for regression. **Note:** To increase generalization and maximum diversity, it might be advantageous to train our systems on different training datasets

We do this **with replacement**. Meaning if a sample is selected once, it can be selected again. This means some samples may appear multiple times in the same training set, while others may not appear at all.

Boosting

Boosting is an ensemble technique that focuses on training a sequence of weak learners, where each subsequent. We take the misclassified instances from the previous model and give them higher weights so that the next model focuses more on those difficult cases. The final prediction is made by combining the predictions of all models, often through weighted voting or averaging.

Boosting algorithm

For each iteration:

1. Train a weak learner on the weighted training data.
2. Evaluate the weak learner's performance on the training data.
3. compute the error rate of the weak learner.
4. Update the sample distribution to give more weight to misclassified instances.
5. Compute the weight of the weak learner based on its accuracy.

AdaBoost

Adaptive Boosting (AdaBoost) is a popular boosting algorithm that combines multiple weak learners to create a strong learner. It adjusts the weights of misclassified instances after each iteration, allowing subsequent weak learners to focus more on difficult cases. The final prediction is made through weighted voting, where each weak learner 's vote is weighted by its accuracy.

Get the algorithm from class slides later

Random Forests

Random Forests is an ensemble learning method that combines multiple decision trees to improve predictive performance and reduce overfitting. Each decision tree is trained on a random subset of the training data (using bagging) and a random subset of features (feature bagging). The final prediction is made by aggregating the predictions of all trees, typically through majority voting.

Random forests take an approach of feature bagging where during training, node splits are based on a random set of features.

* Repeat eh following T times to create T trees: * Sample N instances from the training data with replacement (bagging). * For each node in the tree: * Select a random subset of features of size P. P can be set to the ceiling of $\sqrt{d}$ where d is the total number of features. Ceiling means round up to the nearest integer. * Determine the best split based on the selected features. * Grow the tree to its maximum depth without pruning.