Week 9

Data Mining and Knowledge Discovery

Dr John Evans
j.evans8@herts.ac.uk

University of Hertfordshire UH

Plan for today

Ensemble Methods

Methods For Constructing Ensemble Classifiers

Bias vs Variance

Bagging

Random Forests

Recap

- We discussed factors influencing model overfitting:
  - Limited training size.
  - High model complexity.
  - Presence of noise.
- We saw ways to select different models.
- We saw how to use a validation set and how to compute validation error.
- We also saw how to incorporate model complexity:

$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}.$$

- We saw how to evaluate our model. In particular, we saw:
  - The Holdout Method.
  - Cross-validation.

Ensemble Methods

- A popular method of improving classification is to use multiple classifiers and them aggregate the predictions.
- Such a technique is often known as an *ensemble* or *classifier combination* method.

Ensemble Methods

- A popular method of improving classification is to use multiple classifiers and them aggregate the predictions.
- Such a technique is often known as an *ensemble* or *classifier combination* method.
- The basic idea is to take a set of base classifiers, perform classification and then take a vote on the predictions made by each base classifier.
- The final prediction is then chosen by choosing whichever prediction wins the vote.

Example

- Suppose we have 25 binary classifiers, each of which has an error rate of $\epsilon = 0.35$.
- The ensemble classifier predicts the class label of a test example by taking a majority vote on the prediction made by the base classifiers.

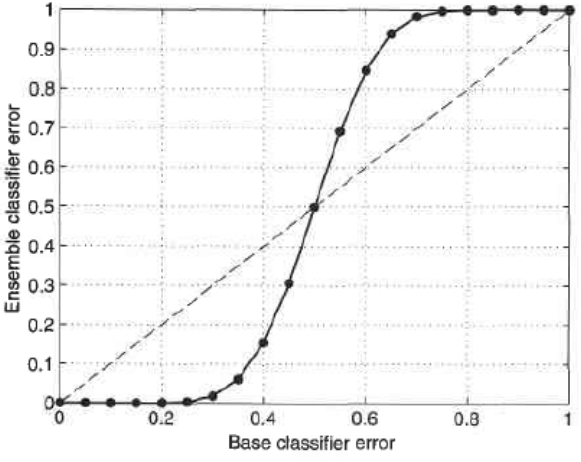
Example

- Suppose we have 25 binary classifiers, each of which has an error rate of $\epsilon = 0.35$.
- The ensemble classifier predicts the class label of a test example by taking a majority vote on the prediction made by the base classifiers.
- We have two options:
  - The base classifiers are identical.
  - The base classifiers are independent, i.e. their errors are uncorrelated.
- In the former case, each base classifier will commit the same mistake and thus the error rate of the ensemble will remain at 0.35.

Example

- Suppose we have 25 binary classifiers, each of which has an error rate of $\epsilon = 0.35$.
- The ensemble classifier predicts the class label of a test example by taking a majority vote on the prediction made by the base classifiers.
- We have two options:
  - The base classifiers are identical.
  - The base classifiers are independent, i.e. their errors are uncorrelated.
- In the former case, each base classifier will commit the same mistake and thus the error rate of the ensemble will remain at 0.35.
- For the latter case, the ensemble will predict incorrectly only if more than half the base classifiers predict incorrectly.
- In this case $e_{ensemble} = \sum_{i=13}^{25} \epsilon^i (1-\epsilon)^{25-i} = 0.06$.

Ensemble Classifier Error



Previous figure explained

- In the previous figure, we see the ensemble error rate ($\epsilon_{ensemble}$) for different base classifier rates (ϵ).
- Again, this is for an ensemble of 25 binary classifiers.

Previous figure explained

- In the previous figure, we see the ensemble error rate ($\epsilon_{ensemble}$) for different base classifier rates (ϵ).
- Again, this is for an ensemble of 25 binary classifiers.
- The diagonal line represents the case in which the base classifiers are identical.
- The solid line represents the case in which base classifiers are independent.

Previous figure explained

- In the previous figure, we see the ensemble error rate ($\epsilon_{ensemble}$) for different base classifier rates (ϵ).
- Again, this is for an ensemble of 25 binary classifiers.
- The diagonal line represents the case in which the base classifiers are identical.
- The solid line represents the case in which base classifiers are independent.

Key Takeaway: ensemble classifiers performs worse than the base classifiers only when ϵ is larger than 0.5, i.e. when the base classifiers are performing worse than random guessing.

Summary

We have two necessary conditions for an ensemble classifier to perform better than a single classifier:

1. The base classifiers should be independent of each other; and
2. The base classifiers should perform better than random guessing.

Summary

We have two necessary conditions for an ensemble classifier to perform better than a single classifier:

1. The base classifiers should be independent of each other; and
2. The base classifiers should perform better than random guessing.

N.B. In practice, it is difficult to ensure total independence among the base classifiers but this is the ideal. Regardless, even when the base classifiers are somewhat correlated, ensemble methods have been shown to improve classification accuracies.

Method 1 : Manipulate the training set

There are several methods for constructing ensemble classifiers based upon manipulating different aspects of the base classifiers or dataset under investigation.

Method 1 : Manipulate the training set

There are several methods for constructing ensemble classifiers based upon manipulating different aspects of the base classifiers or dataset under investigation.

- **Manipulate the training set**
- In this approach, we take the original data and sample it according to some sampling distribution.
- By repeating this, we obtain multiple training sets and can construct a classifier for each training set.
- In this way, we end up with our ensemble.

Method 1 : Manipulate the training set

There are several methods for constructing ensemble classifiers based upon manipulating different aspects of the base classifiers or dataset under investigation.

- **Manipulate the training set**
- In this approach, we take the original data and sample it according to some sampling distribution.
- By repeating this, we obtain multiple training sets and can construct a classifier for each training set.
- In this way, we end up with our ensemble.
- There are many ways to sample the original data and how we do this will greatly affect the ensemble since this process determines the likelihood of a given data point being chosen for testing.
- Two well-known ensembles that manipulate the training set are *bagging* and *boosting*.

Method 2: Manipulate the input features

- In this approach, a subset of input features is chosen to form each training set.
- This subset can be chosen either randomly or based upon domain expertise.
- Some studies have shown this works well with datasets that contain highly redundant features.
- A classic example of this approach is the *random forest* model which uses decision trees as its base classifiers.

Method 3: Manipulate the class labels

- In this approach, the training data is transformed into a binary class problem by randomly partitioning the class labels into two disjoint subsets A_0, A_1.
- Training examples whose class label belongs to the subset A_i are assigned to class i and the relabelled examples are then used to train a base classifier.
- Repeating this process yields an ensemble of base classifiers is obtained.

Method 3: Manipulate the class labels

- In this approach, the training data is transformed into a binary class problem by randomly partitioning the class labels into two disjoint subsets A_0, A_1.
- Training examples whose class label belongs to the subset A_i are assigned to class i and the relabelled examples are then used to train a base classifier.
- Repeating this process yields an ensemble of base classifiers is obtained.
- When a test example is presented, each base classifier C_i is used to predict its class label.
- If the test example is predicted as class 0, then all the classes that belong to A_0 will receive a vote. Likewise for class 1 and A_1.
- At the end, the votes are tallied and the class which receives the highest vote is assigned to the test example.

Method 3: Manipulate the class labels

- In this approach, the training data is transformed into a binary class problem by randomly partitioning the class labels into two disjoint subsets A_0, A_1.
- Training examples whose class label belongs to the subset A_i are assigned to class i and the relabelled examples are then used to train a base classifier.
- Repeating this process yields an ensemble of base classifiers is obtained.
- When a test example is presented, each base classifier C_i is used to predict its class label.
- If the test example is predicted as class 0, then all the classes that belong to A_0 will receive a vote. Likewise for class 1 and A_1.
- At the end, the votes are tallied and the class which receives the highest vote is assigned to the test example.
- This method works better when the number of classes is sufficiently large.
- An example of this approach is the *error-correcting output coding* method.

Method 4: Manipulate the learning algorithm

- In this approach, the learning algorithm itself is manipulated in such a way that applying the algorithm several times on the same data will result in the construction of different classifiers.
- Note that not all learning algorithms can be manipulated in this way.

Method 4: Manipulate the learning algorithm

- In this approach, the learning algorithm itself is manipulated in such a way that applying the algorithm several times on the same data will result in the construction of different classifiers.
- Note that not all learning algorithms can be manipulated in this way.
- Two examples of this include artificial neural networks and decision trees.
- For the former, there can be a change to its network topology, or to the initial weights of the links between neurons.
- For the latter, we can inject randomness into the tree-growing procedure, such as randomly choosing one of the top k attributes for splitting, as opposed to choosing the best splitting attribute at each node.

What happens once an ensemble has been learned?

- Once an ensemble of classifiers has been learned, a test example $\mathbf{x}$ is classified by combining the predictions made by the base classifiers $C_i(\mathbf{x})$:

$$C_*(\mathbf{x}) = f(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x})),$$

  where f is the function that combines the ensemble responses.
- A simple approach is to simply take a majority vote.
- Alternatives exist such as taking a weighted majority vote in which the weight of a base classifier denotes its accuracy or relevance.

What happens once an ensemble has been learned?

- Once an ensemble of classifiers has been learned, a test example $\mathbf{x}$ is classified by combining the predictions made by the base classifiers $C_i(\mathbf{x})$:

$$C_*(\mathbf{x}) = f(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x})),$$

  where f is the function that combines the ensemble responses.
- A simple approach is to simply take a majority vote.
- Alternatives exist such as taking a weighted majority vote in which the weight of a base classifier denotes its accuracy or relevance.

N.B. Ensemble methods show the most improvement when used with *unstable classifiers*; that is, with base classifiers that are sensitive to minor perturbations in the training set, because of high model complexity.

Bias vs Variance

- The previous point relates to the trade-off between *bias* and *variance*.
- Unstable classifiers often have low bias in finding the optimal decision boundary, but their predictions have a high variance for minor changes in the training set or model selection.

Bias vs Variance

- The previous point relates to the trade-off between *bias* and *variance*.
- Unstable classifiers often have low bias in finding the optimal decision boundary, but their predictions have a high variance for minor changes in the training set or model selection.
- The process or formally analysing the generalisation error of a predictive model is called the *bias-variance decomposition*.
- We consider the intuition of this by discussing an analogue of a regression problem.

A regression problem

- Suppose we have a cannon at position $\mathbf{x}$ which fires projectiles at a target y.
- The target corresponds to the desired output at a test instance.
- The starting position corresponds to its observed attributes.
- The projectile represents the model used for predicting the target using the observed attributes.

A regression problem

- Suppose we have a cannon at position $\mathbf{x}$ which fires projectiles at a target y.
- The target corresponds to the desired output at a test instance.
- The starting position corresponds to its observed attributes.
- The projectile represents the model used for predicting the target using the observed attributes.
- Let $\hat{y}$ denote the point where the projectile hits the ground, which is analogous of the prediction of the model.
- We want our predictions to be as close to the true target as possible. However, different trajectories of projectiles are possible based on differences in the training data or in the approach used for model selection.

Our regression problem continued

- We observe a *variance* in the predictions $\hat{y}$ over different runs of projectile.
- Furthermore, the target in our example is not fixed. Instead, it has some freedom to move around and this results in a *noise* component in the true target.
- This can be understood as the non-deterministic nature of the output variable, where the same set of attributes can have different output values.

Our regression problem continued

- We observe a *variance* in the predictions $\hat{y}$ over different runs of projectile.
- Furthermore, the target in our example is not fixed. Instead, it has some freedom to move around and this results in a *noise* component in the true target.
- This can be understood as the non-deterministic nature of the output variable, where the same set of attributes can have different output values.
- We now let $\hat{y}_{avg}$ represent the average prediction of the projectile over multiple runs, and y_{avg} denote the average target value. The *bias* of the model is then given by $\hat{y}_{avg} - \hat{y}_{avg}$.

Our regression problem continued

- We observe a *variance* in the predictions $\hat{y}$ over different runs of projectile.
- Furthermore, the target in our example is not fixed. Instead, it has some freedom to move around and this results in a *noise* component in the true target.
- This can be understood as the non-deterministic nature of the output variable, where the same set of attributes can have different output values.
- We now let $\hat{y}_{avg}$ represent the average prediction of the projectile over multiple runs, and y_{avg} denote the average target value. The *bias* of the model is then given by $\hat{y}_{avg} - \hat{y}_{avg}$.
- In this way, we can have models which produce high variance but low bias by having a wider spread of projectiles but in which the average is close to our target. This is typical of overfitting.
- Alternatively, we can have a model which presents low variance but high bias. In this case, our projectiles are landing in roughly the same location (low spread), but far away from our target. This is typical of underfitting.

Bringing this back to classification

- In the context of classification, it can be shown that the generalisation error of a classification model m can be decomposed into terms involving the bias, variance and noise components of the model:

$$gen.error(m) = c_1 \times noise + bias(m) + c_2 \times variance(m),$$

where c_1, c_2 are constants that depend upon the characteristics of training and test sets.

N.B. While the noise term is intrinsic to the target class, the bias and variance terms depend upon the choice of the classification model.

Bias, variance, overfitting and underfitting

- The bias of the model represents how close the average prediction of the model is to the average target. Models that are able to learn complex decision boundaries (e.g. k- nearest neighbour and multi-layer ANN) generally show low bias.

Bias, variance, overfitting and underfitting

- The bias of the model represents how close the average prediction of the model is to the average target. Models that are able to learn complex decision boundaries (e.g. k- nearest neighbour and multi-layer ANN) generally show low bias.
- The variance of a model captures the stability of its predictions in response to minor perturbations in the training set or the model selection approach.

Bias, variance, overfitting and underfitting

- The bias of the model represents how close the average prediction of the model is to the average target. Models that are able to learn complex decision boundaries (e.g. k- nearest neighbour and multi-layer ANN) generally show low bias.
- The variance of a model captures the stability of its predictions in response to minor perturbations in the training set or the model selection approach.
- We say that a model shows better generalisation performance if it has a *lower bias* and *lower variance*.
- However, if the complexity of a model is high but the training size is small, we generally expect to see a lower bias but higher variance (overfitting).
- An overly simplistic model may show a lower variance but would suffer from a higher bias (underfitting).
- The trade-off between bias and variance provides a useful way for interpreting the effects of underfitting and overfitting on generalisation performance.

Why this matters to ensemble methods

- The bias-variance trade-off can be used to explain why ensemble learning improves the generalisation performance of unstable classifiers.
- If a base classifier shows low bias but high variance, it can become susceptible to overfitting, as even a small change in the training set will result in different predictions.
- However, by combining the responses of multiple base classifiers, we can expect to reduce the overall variance.
- Hence, ensemble learning methods show better performance, primarily by lowering the variance in the predictions, although they can also help reduce the bias.
- One of the simplest approaches for combining predictions and reducing their variance is to compute their average. This forms the basis of the bagging method.

Bagging

- Bagging (also known as *bootstrap aggregating*) is a method of repeatedly sampling (with replacement) a data set according to a uniform probability distribution in such a way that each bootstrap sample has the same size as the original data.

Bagging

- Bagging (also known as *bootstrap aggregating*) is a method of repeatedly sampling (with replacement) a data set according to a uniform probability distribution in such a way that each bootstrap sample has the same size as the original data.
- On average, a bootstrap sample D_i contains approximately 63% of the original training data.
- To see why this is, consider the probability of a sample not being selected in D_i. We take N runs through the data (our original data has N data points) and so the probability of this is $(1 - \frac{1}{N})^N$.
- Thus, the probability of each data point being sampled is $1 - (1 - \frac{1}{N})^N$.
- If N is sufficiently large then this probability converges to $1 - \frac{1}{e} \approx 0.632$.

Bagging

- Bagging (also known as *bootstrap aggregating*) is a method of repeatedly sampling (with replacement) a data set according to a uniform probability distribution in such a way that each bootstrap sample has the same size as the original data.
- On average, a bootstrap sample D_i contains approximately 63% of the original training data.
- To see why this is, consider the probability of a sample not being selected in D_i. We take N runs through the data (our original data has N data points) and so the probability of this is $(1 - \frac{1}{N})^N$.
- Thus, the probability of each data point being sampled is $1 - (1 - \frac{1}{N})^N$.
- If N is sufficiently large then this probability converges to $1 - \frac{1}{e} \approx 0.632$.
- On the following slide, δ refers to the Kronecker delta such that $\delta(-) = 1$ if its argument is true, and 0 otherwise.

Bagging algorithm

Algorithm:	Bagging algorithm
Step 1	Let k be the number of bootstrap samples;

Step 2	**for** $i = 1$ **do**
	Create a bootstrap sample of size N, D_i;
	Train a base classifier C_i on the bootstrap sample D_i;
	end for

Step 3	$C_*(\mathbf{x}) = argmax_y \sum_i \delta(C_i(\mathbf{x}) = y)$.

Example

Consider the following dataset:

x	1	2	3	4	5	6	7	8	9	10
y	1	1	1	-1	-1	-1	-1	1	1	1

- We have a one-dimensional attribute x and a class label y.
- We suppose we only use a one-level binary decision tree, with a test condition $x \leq k$, where k is a split point chosen to minimise the entropy of the leaf nodes.
- Such a tree is known as a *decision stump*.

Example continued

- Without bagging, the best decision stump we can produce splits the instances at either $x \leq 3.5$ or $x \leq 7.5$.
- Either way, the accuracy of the tree is at most 70%.

Example continued

- Without bagging, the best decision stump we can produce splits the instances at either $x \leq 3.5$ or $x \leq 7.5$.
- Either way, the accuracy of the tree is at most 70%.
- Now suppose we apply the bagging procedure on the data set using 10 bootstrap samples. These are as follows, where on the right we have described the decision stump being used for each round:

Bagging round 1

x	1	2	2	3	4	4	5	6	9	9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 3.5 \quad \Rightarrow \quad y = 1$

$x > 3.5 \quad \Rightarrow \quad y = -1$

Example continued

Bagging round 2

x	1	2	3	4	5	8	9	10	10	10
y	1	1	1	-1	-1	1	1	1	1	1

$x \leq 6.5 \Rightarrow y = 1$

$x > 6.5 \Rightarrow y = 1$

Bagging round 3

x	1	2	3	4	4	5	7	7	8	9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 3.5 \Rightarrow y = 1$

$x > 3.5 \Rightarrow y = -1$

Bagging round 4

x	1	1	2	4	4	5	5	7	8	9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 3 \Rightarrow y = 1$

$x > 3 \Rightarrow y = -1$

Example continued

Bagging round 5

x	1	1	2	5	6	6	6	10	10	10
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 3.5 \Rightarrow y = 1$

$x > 3.5 \Rightarrow y = -1$

Bagging round 6

x	2	4	5	6	7	7	7	8	9	10
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 7.5 \Rightarrow y = -1$

$x > 7.5 \Rightarrow y = 1$

Bagging round 7

x	1	4	4	6	7	8	9	9	9	10
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 7.5 \Rightarrow y = -1$

$x > 7.5 \Rightarrow y = 1$

Example continued

Bagging round 8

x	1	2	5	5	5	7	7	8	9	10
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 7.5 \Rightarrow y = -1$

$x > 7.5 \Rightarrow y = 1$

Bagging round 9

x	1	3	4	4	6	7	7	8	10	10
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 7.5 \Rightarrow y = -1$

$x > 7.5 \Rightarrow y = 1$

Bagging round 10

x	1	1	1	1	3	3	8	8	9	9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.5 \Rightarrow y = -1$

$x > 0.5 \Rightarrow y = 1$

Example continued

- We then classify the entire data given above by taking a majority vote among the predictions made by each base classifier.
- The results of the predictions are shown in the table on the following slide.
- Since the class labels are ± 1, taking the majority vote is equivalent to summing up the predicted values of y and examining the resulting sign.
- This is given in the second to last row.
- We note that the ensemble classifier perfectly classifies all 10 examples in the original data.

Example concluded

Round	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$	$x = 7$	$x = 8$	$x = 9$	$x = 10$
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
sum	2	2	2	-6	-6	-6	-6	2	2	2
sign	1	1	1	-1	-1	-1	-1	1	1	1
true class	1	1	1	-1	-1	-1	-1	1	1	1

Bagging summary

- Overall, bagging improves generalisation error by reducing the variance of the base classifiers.
- We note that the performance of bagging depends upon the stability of the base classifier.
- If a base classifier is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data.
- However, if a base classifier is stable (i.e. robust to minor perturbations in the training set), then the error of the ensemble is primarily caused by bias in the base classifier.
- In this situation, bagging may not be able to improve the performance of the base classifiers significantly. It may even degrade the classifier's performance because the effective size of each training set is about 37% smaller than the original data.

Random forests

- Finally, we discuss random forests. These attempt to improve the generalisation performance by constructing an ensemble of *decorrelated* decision trees.
- We do not want correlated trees since all of these trees would sort data in the same way. Consequently, there would be no advantage to this method over a single decision tree.

Random forests

- Finally, we discuss random forests. These attempt to improve the generalisation performance by constructing an ensemble of *decorrelated* decision trees.
- We do not want correlated trees since all of these trees would sort data in the same way. Consequently, there would be no advantage to this method over a single decision tree.
- This is where bagging comes in to play by using a different bootstrap sample of the training data for learning decision trees.
- However, there is a key distinguishing feature of random forests in that at every internal node of a tree, the best splitting criterion is chosen from among a small set of randomly selected attributes.
- In this way, random forests construct ensembles of decision trees by not only manipulating training instances (by using bootstrap samples similar to bagging), but also the input attributes (by using different subsets of attributes at every internal node).

The algorithm

Given a training set D consisting of n instances and d attributes, the basic procedure of training a random forest classifier is as follows:

1. Construct a bootstrap sample D_i of the training set by randomly sampling n instances (with replacement) from D.
2. Use D_i to learn a decision tree T_i, as follows:
   - At every internal node of T_i, randomly sample a set of p attributes.
   - Choose an attribute from this subset that shows the maximum reduction in an impurity measure for splitting.
   - Repeat this procedure until every leaf is pure, i.e. contains only instances from a single class.

Once an ensemble of decision trees has been constructed, their average prediction (majority vote) on a test instance is used as the final prediction of the random forest.

Lack or correlation

- The decision trees involved in a random forest are *unpruned*. They are allowed to grow to their largest possible size until every leaf is pure. Hence, the base classifiers of a random forest represent unstable classifiers that have low bias but high variance because of their large size.

Lack or correlation

- The decision trees involved in a random forest are *unpruned*. They are allowed to grow to their largest possible size until every leaf is pure. Hence, the base classifiers of a random forest represent unstable classifiers that have low bias but high variance because of their large size.
- A **key** property of the base classifiers learned in random forests is the lack of correlation among their model parameters and test predictions.
- This can be attributed to the use of an independently sampled data set D_i for learning every decision tree T_i, similar to the bagging approach.
- However, random forests have the additional advantage of choosing a splitting criterion at every internal node using a different (and randomly selected) subset of attributes.
- This property significantly helps in breaking the correlation structure, if any, among the decision trees T_i.

Random forests encourage decorrelated trees

- Consider a training set involving a large number of attributes where only a small subset of attributes are strong predictors of the target class (the rest are weak indicators).
- For such a training set, even if we consider different bootstrap samples D_i for learning T_i, we would mostly be choosing the same attributes for splitting at internal nodes, because the weak attributes would be largely overlooked when compared with the strong predictors.
- This can result in a considerable correlation among the trees.

Random forests encourage decorrelated trees

- Consider a training set involving a large number of attributes where only a small subset of attributes are strong predictors of the target class (the rest are weak indicators).
- For such a training set, even if we consider different bootstrap samples D_i for learning T_i, we would mostly be choosing the same attributes for splitting at internal nodes, because the weak attributes would be largely overlooked when compared with the strong predictors.
- This can result in a considerable correlation among the trees.
- However, if we restrict the choice of attributes at every internal node to a random subset of attributes, then we can ensure the selection of both strong and weak predictors, thus promoting diversity among the trees.
- This principle is utilised by random forests for creating decorrelated decision trees.

Benefits to decorrelated decision trees

- By aggregating the predictions of an ensemble of strong and decorrelated decision trees, random forests are able to reduce the variance of the trees without negatively impacting their low bias.
- This makes random forests quite robust to overfitting.
- Additionally, because of their ability to consider only a small subset of attributes at every internal node, random forests are computationally fast and robust even in high-dimensional settings.

How do we choose the number of attributes to be selected at each node?

How do we choose the number of attributes to be selected at each node?

- This hyperparameter (labelled p) of the random forest classifier is often key as a small value of p can reduce the correlation among the classifiers, but may also reduce their strength.
- In contrast, a large value can improve their strength but may result in correlated trees similar to bagging.

How do we choose the number of attributes to be selected at each node?

- This hyperparameter (labelled p) of the random forest classifier is often key as a small value of p can reduce the correlation among the classifiers, but may also reduce their strength.
- In contrast, a large value can improve their strength but may result in correlated trees similar to bagging.
- There are several common suggestions for p in the literature with the following two being quite popular:
  - $p = \sqrt{d}$;
  - $p = \log_2(d) + 1$.

Alternative options

- Another option is to select p by tuning it over a validation set.

Alternative options

- Another option is to select p by tuning it over a validation set.
- Yet another method (which does not require using a separate validation set) involves computing a reliable estimate of the generalisation error rate directly during training.
- This is known as the *out-of-bag (oob)* estimate.
- The oob estimate can be computed for any generic ensemble learning method that builds independent base classifiers using bootstrap samples of the training set, e.g. bagging and random forests.

Random forest summary

- Random forests have been empirically found to provide significant improvements in generalisation performance that are often comparable, if not superior, to the improvements provided by the AdaBoost algorithm.
- Random forests are also more robust to overfitting and run much faster than the AdaBoost algorithm.

Random forest summary

- Random forests have been empirically found to provide significant improvements in generalisation performance that are often comparable, if not superior, to the improvements provided by the AdaBoost algorithm.
- Random forests are also more robust to overfitting and run much faster than the AdaBoost algorithm.
- We conclude with a table which compares the accuracy of a decision tree classifier against three ensemble methods for 24 classic datasets.
- We note that the base classifiers used in each ensemble consisted of 50 decision trees and the classification accuracies are obtained from ten-fold cross-validation.
- Observe that the ensemble classifiers generally outperform a single decision tree classifier on many of the datasets.

Table of comparison

Data Set	Number of (Attributes, Classes, Records)	Decision Tree (%)	Bagging (%)	Boosting (%)	RF (%)
Anneal	(39, 6, 898)	92.09	94.43	95.43	95.43
Australia	(15, 2, 690)	85.51	87.10	85.22	85.80
Auto	(26, 7, 205)	81.95	85.37	85.37	84.39
Breast	(11, 2, 699)	95.14	96.42	97.28	96.14
Cleve	(14, 2, 303)	76.24	81.52	82.18	82.18
Credit	(16, 2, 690)	85.8	86.23	86.09	85.8
Diabetes	(9, 2, 768)	72.40	76.30	73.18	75.13
German	(21, 2, 1000)	70.90	73.40	73.00	74.5
Glass	(10, 7, 214)	67.29	76.17	77.57	78.04
Heart	(14, 2, 270)	80.00	81.48	80.74	83.33
Hepatitis	(20, 2, 155)	81.94	81.29	83.87	83.23
Horse	(23, 2, 368)	85.33	85.87	81.25	85.33
Ionosphere	(35, 2, 351)	89.17	92.02	93.73	93.45
Iris	(5, 3, 150)	94.67	94.67	94.00	93.33
Labor	(17, 2, 57)	78.95	84.21	89.47	84.21
Led7	(8, 10, 3200)	73.34	73.66	73.34	73.06
Lymphography	(19, 4, 148)	77.03	79.05	85.14	82.43
Pima	(9, 2, 768)	74.35	76.69	73.44	77.60
Sonar	(61, 2, 208)	78.85	78.85	84.62	85.58
Tic-tac-toe	(10, 2, 958)	83.72	93.84	98.54	95.82
Vehicle	(19, 4, 846)	71.04	74.11	78.25	74.94
Waveform	(22, 3, 5000)	76.44	83.30	83.90	84.04
Wine	(14, 3, 178)	94.38	96.07	97.75	97.75
Zoo	(17, 7, 101)	93.07	93.07	95.05	97.03

Summary

- We have seen the idea of ensemble methods which combine several base classifiers and then use each classifier's prediction to make a final prediction (often by majority voting).
- We have seen methods for constructing ensembles:
  - Manipulate the training set;
  - Manipulate the input features;
  - Manipulate the class labels;
  - Manipulate the learning algorithm.
- We have seen the trade-off between bias and variance and the effects of underfitting and overfitting on generalisation performance.
- We have seen bagging which repeatedly samples (with replacement) a data set according to a uniform probability distribution in such a way that each bootstrap sample has the same size as the original data.
- We have seen random forests whose base classifiers are decision trees. They manipulate training instances and the input attributes.

Exercise before next time: Read pp. 121-125 of the notes.