14 Ensemble Classification

14.1 Introduction

The idea of *ensemble classification* is to learn not just one classifier but a set of classifiers, called an *ensemble* of classifiers, and then to combine their predictions for the classification of unseen instances using some form of voting. This is illustrated in Figure 14.1 below. It is hoped that the ensemble will collectively have a higher level of predictive accuracy than any one of the individual classifiers, but that is not guaranteed.

The term *ensemble learning* is often used to mean the same as ensemble classification, but the former is a more general technique where a set of models is learnt that collectively can be applied to solving a problem of potentially any kind, not just classification.

The individual classifiers in an ensemble are known as *base classifiers*. If the base classifiers are all of the same kind (e.g. decision trees) the ensemble is known as *homogeneous*. Otherwise it is known as *heterogeneous*.

A simple form of ensemble classification algorithm is:

1. Generate N classifiers for a given dataset
2. For an unseen instance X
   a) Compute the predicted classification of X for each of the N classifiers
   b) Select the classification that is most frequently predicted.

This is a *majority voting* model where each time a classifier predicts a particular classification for an unseen instance it counts as one 'vote' for that

M. Bramer, *Principles of Data Mining*, Undergraduate Topics
in Computer Science, DOI 10.1007/978-1-4471-7307-6_14

classification. With N classifiers in the ensemble there will be a total of N votes and the classification with most votes wins, i.e. is deemed to be the ensemble's prediction of the correct classification.

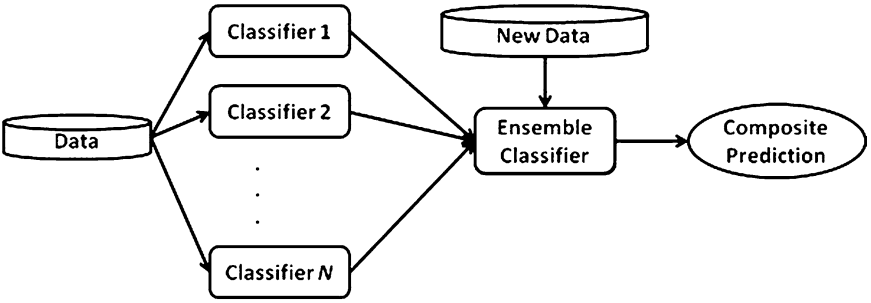

Figure 14.1 Ensemble Classification

The obvious objection to an ensemble classifier approach is that generating N classifiers takes much longer than only one and this additional effort is only justified if the performance of the ensemble is substantially better than that of just a single classifier. There is no guarantee that this will be the case for a given set of test data and far less so for an individual unseen instance, but intuitively it seems reasonable to believe that N classifiers 'working together' have the potential to give better predictive accuracy than one on its own. In practice this is likely to depend on how the classifiers are generated and how their predictions are combined (majority voting or otherwise).

In this chapter we will restrict our attention to the homogeneous case, where all the classifiers are of the same kind, say decision trees. There are several ways in which an ensemble can be formed, for example:

- N trees generated using the same tree generation algorithm, with different parameter settings, all using the same training data.
- N trees generated using the same tree generation algorithm, all with different training data and either with the same or with different parameter settings.
- N trees generated using a variety of different tree generation algorithms, either with the same or with different training data.
- N trees generated using a different subset of the attributes for each one.

If the additional effort needed to generate an ensemble of classifiers is to be worthwhile, the best approach is unlikely to be to generate trees that are all

very similar, as they are all likely to give a very similar 'standard' performance. A better strategy is likely to be to generate trees (or other classifiers) that are diverse, in the hope that some will give much better than 'standard' performance, even if others are much worse. Those in the latter category should not be included in the ensemble; those in the former should be retained. This leads naturally to the idea of generating a large number of classifiers in some random way and then retaining only the best.

Two pioneering pieces of work in this field are the Random Decision Forests system developed by Tin Kam Ho [1] and the Random Forests system of Leo Breiman [2]. Both use the approach of generating a large number of decision trees in a way that has a substantial random element, measuring their performance and then selecting the best trees for the ensemble. To quote Stahl and Bramer [3]: "Ho argues that traditional trees often cannot be grown over a certain level of complexity without risking a loss of generalisation caused by overfitting on the training data. Ho proposes inducing multiple trees in randomly selected subsets of the feature space. He claims that the combined classification will improve, as the individual trees will generalise better on the classification for their subset of the feature space".

Ho's work introduced the idea of making a random selection of the attributes to use when generating each classifier. Breiman added to this by introducing a technique known as *bagging* for generating multiple different but related training sets from a single set of training data, with the aim of reducing overfitting and improving classification accuracy [4].

Naturally this is computationally expensive to do. Ho's and Breiman's papers are both important contributions to the field and are well worth studying in detail. However as usual there are many other ways of implementing the same general ideas once they have been set out and the description given in this chapter is our own.

To develop the idea of basing an ensemble on random classifiers further we need:

- A means of generating a large number of classifiers (say 100) in a random fashion and
- A way of measuring the performance of each one.

The final step is to choose all those that meet some criterion to include in an ensemble. There are several ways of doing this. For example we may select, say, the 10 classifiers with the best performance or all the classifiers with performance above some threshold of accuracy.

14.2 Estimating the Performance of a Classifier

Elsewhere in this book we have described the standard methodology for developing and estimating the performance of a classifier: divide the available data into a training set and a test set, use the training set to develop the classifier and then use the test set to estimate how the classifier will perform on (genuinely) unseen data.

For an ensemble classifier the procedure requires an extra dataset called a *validation dataset* associated with each classifier. The method is as follows:

1. Divide the available data into a test set and the remainder.
2. For each candidate classifier
   a) Divide the remaining data from step (1) into training data and validation data in some suitable fashion.
   b) Generate a classifier using the training data.
   c) Run the classifier against the validation data to give an estimate of its performance.
3. Use the performance estimates to find the best classifiers, e.g. all those with predictive accuracy greater than a specified percentage or perhaps the best X. If the number of classifiers remaining from this step is M, together they comprise an ensemble of size M.
4. Use the ensemble to classify each of the instances in the test set selected at step (1) and use the result as an estimate of the performance of the ensemble on (genuinely) unseen data.

The method used to predict the classification of unseen instances at step (4) is generally to use each of the M classifiers independently and then to combine their 'votes' for the correct classification (see Section 14.5 below).

How many classifiers to use in an ensemble is a matter for experiment, but to take advantage of the opportunity given by introducing a random element into the classifier generation process, i.e. that some particularly good classifiers will emerge by chance, a reasonable number might be, say, 100 with perhaps the best 10 chosen to form the ensemble itself.

14.3 Selecting a Different Training Set for Each Classifier

One problem that arises when implementing step (2)(a) of the algorithm in the previous section 'divide the remainder of the data into training data and validation data, in some suitable fashion' is how best to do this a large number of times, each giving a different division into the two datasets.

An approach to this which was implemented by Breiman [4] in a different context and later used in his Random Forests system is called *bagging*. (Bagging is short for 'bootstrap aggregating', but the significance of this term will not be explained here.)

Let us assume that the data described as the 'remaining data' in the last section, i.e. all the available data less instances removed to form a test set, comprises N instances. The bagging method is then as follows, applied to form each candidate classifier in turn.

- Randomly select N instances, one-by-one, at each stage selecting from the full set of instances (we call this *sampling with replacement*). This will lead to a training set of N instances in which inevitably some of the instances will appear more than once, perhaps several times, and others will not appear at all.
- There are likely to be many instances left unselected by this process. Collect them together to form a validation set.

It is extremely unlikely that sampling with replacement N times from a collection of N instances will lead to each instance being selected exactly once. The opposite extreme, where a single instance happens to be selected N times, is also extremely unlikely. To see what is likely to happen in the usual case we start by asking what is the probability that a particular instance in the 'remaining data' is never picked.

The probability of a particular instance being selected at the first 'pick' is $1/N$, so the probability that it is not selected is $1-1/N$. Each of the N picks is independent of the others, as all N instances are available for picking each time, so the probability of a particular instance never being picked as the training set of N instances is assembled is $(1-1/N)^N$. As N becomes large, this value can be proved to become extremely close to the value $1/e$ (mathematicians call this its *limiting value*). The symbol e represents a well-known 'mathematical constant' with the value 2.71828. Thus the limiting value is $1/e = 0.368$. The value of $(1-1/N)^N$ approximates this value to two decimal places for values of N as small as 64.

Since the same calculation applies to all instances and those never selected form the validation dataset for the classifier, it follows that for a reasonably large dataset of 'remaining data' we can expect that the validation dataset will comprise (on average) 36.8% of the instances. It follows that the other 63.2% go into the training set, some of them many times, to make a training set of N instances.

The significance of the training set being 'padded out' to N instances with duplicate values is far from negligible. Depending on the algorithm used, the classifier generated may be substantially different from the one obtained if duplicate values are deleted from the training set, which is a possible alternative approach.

14.4 Selecting a Different Set of Attributes for Each Classifier

One of the ideas introduced in Ho's Random Decision Forests system was that of processing only a subset of the available attributes (he uses the equivalent term 'features'), selected at random for each decision tree. The general idea is that combining classifiers produced by trees generated this way will give greater accuracy than a single classifier as the individual trees will generalise better on their subset of the available features.

One way of selecting attributes at random is just to choose a random subset from the total number available, with a different subset for each classifier. Another more complex approach would be similar to that for selecting instances for a training set in the previous section. If there are a total of N attributes, then N attributes are picked one at a time, in each case from the full collection of N possibilities. The analysis given in the previous section demonstrates that on average approximately 63.2% of the attributes will be selected for each decision tree by this method. In this case, the attributes not selected would simply be discarded. Duplicates of attributes already selected would also be discarded.

A random selection of attributes can be made just once for each decision tree. An alternative would be to make a further random selection at each node of an evolving decision tree from the attributes remaining under consideration at that point.

14.5 Combining Classifications: Alternative Voting Systems

Having constructed an ensemble of N classifiers, how can their predictions of the correct classification of an unseen instance (whether one in the test set or a genuinely unseen instance) best be combined into a single prediction?

The method adopted in both Ho's Random Decision Forests paper and Breiman's Random Forests paper is simply to treat each prediction as a vote for a particular classification, giving a total of N votes, with the prediction collecting the most votes being considered the winner. We will call this approach *majority voting* or *simple majority voting*. As with real-world voting systems for elections it is quite easy to point to possible flaws in this approach.

Classifier	Predicted Class
1	A
2	B
3	A
4	B
5	A
6	C
7	C
8	A
9	C
10	B

Figure 14.2 Predicted Classes for an Ensemble of 10 Classifiers

Figure 14.2 shows a possible situation. Classification A gained 4 votes, against 3 for B and 3 for C and so is 'elected', even though only 4 out of 10 classifiers made that prediction. Winning with a minority of the votes cast may (or may not) be acceptable for elections where the government of a country is at stake. For the purposes of this book, the important question is how reliable a prediction made this way is likely to be – to which the obvious answer is 'not very'.

Figure 14.3 is the same as Figure 14.2 but with an additional column: 'Accuracy'. This shows the predictive accuracy of the classifier on its validation dataset during the ensemble creation process, expressed as a proportion from 0 to 1. All the values are quite high, or the classifier would not have been included in the ensemble, but some are appreciably higher than others.

Classifier	Accuracy	Predicted Class
1	0.65	A
2	0.90	B
3	0.65	A
4	0.85	B
5	0.70	A
6	0.70	C
7	0.90	C
8	0.65	A
9	0.80	C
10	0.95	B
Total	7.75	

Figure 14.3 An Ensemble of Classifiers with Predictive Accuracy Information

We can now adopt a *weighted majority voting* approach, with each vote for a classification weighted by the proportion given in the middle column.

- Now classifier A gains $0.65 + 0.65 + 0.7 + 0.65 = 2.65$ votes.
- Classifier B gains $0.9 + 0.85 + 0.95 = 2.7$ votes.
- Classifier C gains $0.7 + 0.9 + 0.8 = 2.4$ votes.
- The total number of votes available is $0.65 + 0.9 + \ldots + 0.95 = 7.75$.

With this approach classifier B is now the winner. This seems reasonable as it gained the votes of three of the best classifiers, judged by their performance on their validation datasets (which vary from one classifier to another), whereas candidate classifier A gained the votes of four relatively weak classifiers. In this case choosing B as the winning classifier seems justified.

However it is possible to make the situation more complex still. An overall predictive accuracy figure of say 0.85 can conceal considerable variation in performance. We will focus on classifier 4 with overall predictive accuracy of 0.85 and consider a possible confusion matrix for it, assuming there were exactly 1,000 instances in its validation dataset. (Confusion matrices are discussed in Chapter 7.)

From Figure 14.4 we can see that classification B was quite rare in the validation dataset for classifier 4. Of the 100 instances with that classification only 50 were correctly predicted. Even worse, if we look at the 120 times that classification B was predicted by classifier 4, only 50 times was the prediction correct. Now it seems as if giving classifier 4 a weighted value of 0.85 for its

		Predicted Class			Total
		A	*B*	*C*	
Actual Class	*A*	550	30	20	600
	B	20	50	30	100
	C	10	40	250	300
Total		580	120	300	1000

Figure 14.4 Confusion Matrix for Classifier 4

prediction of classification B was far too optimistic. Perhaps it should have been just $50/120 = 0.417$.

Looking at confusion matrices gives us an approach to combining votes from multiple classifiers, which we will call 'track record voting'. For classifier 4, when it predicts class B: 30 times out of 120 the correct classification is A (25%), 50 times out of 120 the correct classification is B (41.7%) and 40 times out of 120 the correct classification is C (33.3%)

We say that a prediction of B by classifier 4 gives votes of 0.25, 0.417 and 0.333 for classifications A, B and C respectively. Note that these figures are all far below the overall predictive accuracy of the classifier (0.85). The explanation is that classifier 4 is very reliable when it predicts class A (correct 550 times out of $580 = 94.8\%$) and class C (correct 250 times out of $300 = 83.3\%$) but very unreliable when it predicts class B (correct only 50 times out of $120 = 41.7\%$).

		Vote for Class			Total
Classifier	Predicted Class	*A*	*B*	*C*	
1	*A*	0.80	0.05	0.15	1.0
2	*B*	0.10	0.80	0.10	1.0
3	*A*	0.75	0.20	0.05	1.0
4	*B*	0.25	0.42	0.33	1.0
5	*A*	0.40	0.20	0.40	1.0
6	*C*	0.05	0.05	0.90	1.0
7	*C*	0.10	0.10	0.80	1.0
8	*A*	0.75	0.20	0.05	1.0
9	*C*	0.10	0.00	0.90	1.0
10	*B*	0.10	0.80	0.10	1.0
Total		3.40	2.82	3.78	10.0

Figure 14.5 Ensemble of Classifiers with Voting Based on 'Track Record'

Figure 14.5 is a revised version of Figure 14.3. Now each classifier again has one vote, which it casts as three proportions. For example classifier 4 predicts class B for the unseen instance under consideration. This produces not a single vote for class B, but a vote split into three parts cast for all three classes A, B and C, in this case the values 0.25, 0.42 and 0.33 respectively. These proportions are derived from the 'Predicted Class B' column of the confusion matrix for classifier 4 (Figure 14.4).

Adding the votes for each of the three classes in Figure 14.5, the winner now (rather surprisingly) is class C, mainly because of the three high votes of 0.9 twice and 0.8.

Which of the three methods illustrated in this section is the most reliable? The first predicted class A, the second class B and the third class C. There is no clear-cut answer to this. The point is that there are a number of ways the votes can be combined in an ensemble classifier rather than just one.

Looking again at Figure 14.5 there are further complications to take into account. Classifier 5, which predicts class A has 'votes' of 0.4, 0.2 and 0.4. This means that for its validation data when it predicted class A, only 40% of the instances were actually of class A, 20% of the instances were class B and 40% of the instances were class C. What credibility can be given to a prediction of class A by that classifier? We can look at the three proportions for classifier 5 as indications of its 'track record' when predicting class A. On that basis there seems no reason at all to trust it and we might consider eliminating that classifier from consideration any time its prediction is A, as well as eliminating classifier 4 when its prediction is class B. However, if we do so, we will have implicitly moved from a 'democratic' model – one classifier, one vote – to something closer to a 'community of experts' approach.

Suppose the 10 classifiers represent 10 medical consultants in a hospital and A, B and C are three treatments to give a patient with a life-threatening condition. The consultants are trying to predict which treatment is most likely to prove effective. Why should anyone trust consultants 4 and 5, with their poor track records when predicting B and A respectively?

By contrast consultant 6, whose prediction is that treatment C will prove the most effective at saving the patient, has a track record of 90% success when making that prediction. The only consultant to compare with consultant 6 is number 9, who also has a track record of 90% success when predicting C. With two such experts making the same choice, who would wish to contradict them? Even the act of counting the votes seems not only pointless but unnecessarily risky, just in case the other eight less successful consultants might happen to outvote the two leading experts.

We could go on elaborating this example but will stop here. Clearly it is possible to look at the question of how best to combine the classifications

produced by the different classifiers in an ensemble in a variety of different ways. Which way is most likely to give a high level of classification accuracy on unseen data? As so often in data mining, only experimentation with different datasets can give us the answer, but whatever the best approach turns out to be for an 'average' dataset, it is most unlikely that a single method will be best for all datasets or for all unseen instances and it is desirable to have a range of options available.

14.6 Parallel Ensemble Classifiers

As mentioned previously, an important practical obstacle to an ensemble classifier approach is the computation time needed to generate N classifiers rather than just one.

One way of dealing with this is to distribute the work around a local area network of personal computers, with each machine responsible for generating one or more classifiers and estimating its performance using a corresponding validation dataset. This general approach is described in Chapter 13 in the context of dealing with a large volume of data, rather than (as here) generating a large number of classifiers.

Depending on the way in which the ensemble is formed (as discussed in Section 14.1) the machines in the network might all make use of the same data in a central location, or all have identical local copies of the data, or they might begin by taking a sample of a common dataset (e.g. when using a bagging approach, as described in Section 14.3).

If we envisage a network of say 10 machines, we might generate 500 classifiers (50 per machine), estimate the performance of each one using its own validation dataset and retain (say) the 50 best. We might then rearrange the locations of the best 50 classifiers so that there are 5 on each of the 10 machines, or possibly we might put them all together on a single machine, if the volume of unseen data that needs to be processed is expected to be small.

The field of Parallel Ensemble Classifiers is a relatively new one, but appears promising. Two papers that give further information are [5] and [6].

14.7 Chapter Summary

This chapter is concerned with ensemble classification, i.e. using a set of classifiers to classify unseen data rather than just a single one. The classifiers in

the ensemble all predict the correct classification of each unseen instance and their predictions are then combined using some form of voting system.

The idea of a random forest of classifiers is introduced and issues relating to the selection of a different training set and/or a different set of attributes from a given dataset when constructing each of the classifiers are discussed.

A number of alternative ways of combining the classifications produced by an ensemble of classifiers are considered. The chapter concludes with a brief discussion of a distributed processing approach to dealing with the large amount of computation often required to generate an ensemble.

14.8 Self-assessment Exercises for Chapter 14

Given the values shown in Figure 14.5:

1. What would be the effect of setting a threshold of 0.5, i.e. discounting any classifier for which the entry in the table (the 'vote') for the predicted class is less than 0.5?
2. What would be the effect of setting a threshold of 0.8?

References

[1] Ho, T. K. (1995). Random decision forests. *International Conference on Document Analysis and Recognition*, *1*, 278.

[2] Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

[3] Stahl, F., & Bramer, M. (2011). Random prism: an alternative to random forests. In *Research and development in intelligent systems XXVIII* (pp. 5–18). Springer.

[4] Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

[5] Stahl, F., May, D., & Bramer, M. (2012). Parallel random prism: a computationally efficient ensemble learner for classification. In *Research and development in intelligent systems XXIX*. Springer.

[6] Panda, B., Herbach, J. S., Basu, S., & Bayardo, R. J. (2009). Planet: massively parallel learning of tree ensembles with mapreduce. *Proceedings of the VLDB Endowment*, *2*, 1426–1437.