6

Ensemble Pruning

6.1 What is Ensemble Pruning

Given a set of trained individual learners, rather than combining all of them, **ensemble pruning** tries to select a subset of individual learners to comprise the ensemble.

An apparent advantage of ensemble pruning is to obtain ensembles with smaller sizes; this reduces the storage resources required for storing the ensembles and the computational resources required for calculating outputs of individual learners, and thus improves efficiency. There is another benefit, that is, the generalization performance of the pruned ensemble may be even better than the ensemble consisting of all the given individual learners.

The first study on ensemble pruning is possibly [Margineantu and Dietterich, 1997] which tried to prune boosted ensembles. Tamon and Xiang [2000], however, showed that boosting pruning is intractable even to approximate. Instead of pruning ensembles generated by sequential methods, Zhou et al. [2002b] tried to prune ensembles generated by parallel methods such as Bagging, and showed that the pruning can lead to smaller ensembles with better generalization performance. Later, most ensemble pruning studies were devoted to parallel ensemble methods. Caruana et al. [2004] showed that pruning parallel heterogeneous ensembles comprising different types of individual learners is better than taking the original heterogeneous ensembles. In [Zhou et al., 2002b] the pruning of parallel ensembles was called **selective ensemble**; while in [Caruana et al., 2004], the pruning of parallel heterogeneous ensembles was called **ensemble selection**. In this chapter we put all of them under the umbrella of *ensemble pruning*.

Originally, ensemble pruning was defined for the setting where *the individual learners have already been generated*, and no more individual learners will be generated from training data during the pruning process. Notice that traditional **sequential ensemble methods** will discard some individual learners during their training process, but that is not ensemble pruning. These methods typically generate individual learners one by one; once an individual learner is generated, a *sanity check* is applied and the individual learner will be discarded if it can not pass the check. Such a sanity check is

important to ensure the validity of sequential ensembles and prevent them from growing infinitely. For example, in AdaBoost an individual learner will be discarded if its accuracy is below 0.5; however, AdaBoost is not an ensemble pruning method, and boosting pruning [Margineantu and Dietterich, 1997] tries to reduce the number of individual learners after the boosting procedure has stopped and no more individual learners will be generated. It is noteworthy that some recent studies have extended ensemble pruning to all steps of ensemble construction, and individual learners may be pruned even before all individual learners have been generated. Nevertheless, an essential difference between ensemble pruning and sequential ensemble methods remains: for sequential ensemble methods, an individual learner would not be excluded once it is added into the ensemble; while for ensemble pruning methods, any individual learners may be excluded, even for the ones which have been kept in the ensemble for a long time.

Ensemble pruning can be viewed as a special kind of **Stacking**. As introduced in Chapter 4, Stacking tries to apply a meta-learner to combine the individual learners, while the ensemble pruning procedure can be viewed as a special meta-learner. Also, recall that as mentioned in Chapter 4, if we do not worry about how the individual learners are generated, then different ensemble methods can be regarded as different implementations of weighted combination; from this aspect, ensemble pruning can be regarded as a procedure which sets the weights on some learners to zero.

6.2 Many Could be Better Than All

In order to show that it is possible to get a smaller yet better ensemble through ensemble pruning, this section introduces Zhou et al. [2002b]'s analyses.

We start from the regression setting on which the analysis is easier. Suppose there are N individual learners $h_1, \ldots, h_N$ available, and thus the final ensemble size $T \leq N$. Without loss of generality, assume that the learners are combined via weighted averaging according to (4.9) and the weights are constrained by (4.10). For simplicity, assume that equal weights are used, and thus, from (4.11) we have the generalization error of the ensemble as

$$err = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij}/N^2, \tag{6.1}$$

where C_{ij} is defined in (4.12) and measures the correlation between h_i and h_j. If the kth individual learner is excluded from the ensemble, the gener-

alization error of the pruned ensemble is

$$err' = \sum_{\substack{i=1 \\ i \neq k}}^{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} C_{ij}/(N-1)^2. \tag{6.2}$$

By comparing (6.1) and (6.2), we get the condition under which err is not smaller than err', implying that the pruned ensemble is better than the all-member ensemble, that is,

$$(2N-1) \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} \leq 2N^2 \sum_{\substack{i=1 \\ i \neq k}}^{N} C_{ik} + N^2 C_{kk}. \tag{6.3}$$

(6.3) usually holds in practice since the individual learners are often highly correlated. For an extreme example, when all the individual learners are duplicates, (6.3) indicates that the ensemble size can be reduced without sacrificing generalization ability. The simple analysis above shows that in regression, given a number of individual learners, ensembling some instead of all of them may be better.

It is interesting to study the difference between ensemble pruning and *sequential ensemble methods* based on (6.3). As above, let N denote the upper bound of the final ensemble size. Suppose the sequential ensemble method employs the sanity check that the new individual learner h_k ($1 < k \leq N$) will be kept if the ensemble consisting of $h_1, \ldots, h_k$ is better than the ensemble consisted of $h_1, \ldots, h_{k-1}$ on mean squared error. Then, h_k will be discarded if [Perrone and Cooper, 1993]

$$(2k-1) \sum_{i=1}^{k-1} \sum_{j=1}^{k-1} C_{ij} \leq 2(k-1)^2 \sum_{i=1}^{k-1} C_{ik} + (k-1)^2 C_{kk}. \tag{6.4}$$

Comparing (6.3) and (6.4) it is easy to see the following. Firstly, ensemble pruning methods consider the correlation among *all* the individual learners while sequential ensemble methods consider only the correlation between the new individual learner and previously generated ones. For example, assume $N = 100$ and $k = 10$; sequential ensemble methods consider only the correlation between $h_1, \ldots, h_{10}$, while ensemble pruning methods consider the correlations between $h_1, \ldots, h_{100}$. Secondly, when (6.4) holds, sequential ensemble methods will only discard h_k, but $h_1, \ldots, h_{k-1}$ won't be discarded; while any classifier in $h_1, \ldots, h_N$ may be discarded by ensemble pruning methods when (6.3) holds.

Notice that the analysis from (6.1) to (6.4) only applies to regression. Since supervised learning includes regression and classification, analysis of classification setting is needed for a unified result. Again let N denote the number of available individual classifiers, and thus the final ensemble size

$T \leq N$. Without loss of generality, consider binary classification with labels $\{-1, +1\}$, and assume that the learners are combined via majority voting introduced in Section 4.3.1 and ties are broken arbitrarily. Given m training instances, the expected output on these instances is $(f_1, \ldots, f_m)^\top$ where f_j is the ground-truth of the jth instance, and the prediction made by the ith classifier h_i on these instances is $(h_{i1}, \ldots, h_{im})^\top$ where h_{ij} is the prediction on the jth instance. Since $f_j, h_{ij} \in \{-1, +1\}$, it is obvious that h_i correctly classifies the jth instance when $h_{ij} f_j = +1$. Thus, the error of the ith classifier on these m instances is

$$err(h_i) = \frac{1}{m} \sum_{j=1}^{m} \eta(h_{ij} f_j) \,, \tag{6.5}$$

where $\eta(\cdot)$ is a function defined as

$$\eta(x) = \begin{cases} 1 & \text{if } x = -1 \\ 0.5 & \text{if } x = 0 \\ 0 & \text{if } x = 1 \end{cases} \tag{6.6}$$

Let $\boldsymbol{s} = (s_1, \ldots, s_m)^\top$ where $s_j = \sum_{i=1}^{N} h_{ij}$. The output of the all-member ensemble on the jth instance is

$$H_j = \texttt{sign}(s_j) \,. \tag{6.7}$$

It is obvious that $H_j \in \{-1, 0, +1\}$. The prediction of the all-member ensemble on the jth instance is correct when $H_j f_j = +1$ and wrong when $H_j f_j = -1$, while $H_j f_j = 0$ corresponds to a tie. Thus, the error of the all-member ensemble is

$$err = \frac{1}{m} \sum_{j=1}^{m} \eta(H_j f_j) \,. \tag{6.8}$$

Now, suppose the kth individual classifier is excluded from the ensemble. The prediction made by the pruned ensemble on the jth instance is

$$H'_j = \texttt{sign}(s_j - h_{kj}) \,, \tag{6.9}$$

and the error of the pruned ensemble is

$$err' = \frac{1}{m} \sum_{j=1}^{m} \eta(H'_j f_j) \,. \tag{6.10}$$

Then, by comparing (6.8) and (6.10), we get the condition under which err is not smaller than err', implying that the pruned ensemble is better than the all-member ensemble; that is,

$$\sum_{j=1}^{m} \left(\eta\left(\texttt{sign}\left(s_j\right) f_j\right) - \eta\left(\texttt{sign}\left(s_j - h_{kj}\right) f_j\right) \right) \geq 0 \,. \tag{6.11}$$

Since the exclusion of the kth individual classifier will not change the output of the ensemble if $|s_j| > 1$, and based on the property that

$$\eta(\mathtt{sign}(x)) - \eta(\mathtt{sign}(x-y)) = -\frac{1}{2}\mathtt{sign}(x+y)\,, \tag{6.12}$$

the condition for the kth individual classifier to be pruned is

$$\sum_{j\in\{\arg_j |s_j|\leq 1\}} \mathtt{sign}((s_j + h_{kj})f_j) \leq 0\,. \tag{6.13}$$

(6.13) usually holds in practice since the individual classifiers are often highly correlated. For an extreme example, when all the individual classifiers are duplicates, (6.13) indicates that the ensemble size can be reduced without sacrificing generalization ability.

Through combining the analyses on both regression and classification (i.e., (6.3) and (6.13)), we get the theorem of MCBTA ("*many could be better than all*") [Zhou et al., 2002b], which indicates that for supervised learning, given a set of individual learners, it may be better to ensemble some instead of all these individual learners.

6.3 Categorization of Pruning Methods

Notice that, simply pruning individual learners with poor performance may not lead to a good pruned ensemble. Generally, it is better to keep some accurate individuals together with some not-that-good but complementary individuals. Furthermore, notice that neither (6.3) nor (6.13) provides practical solutions to ensemble pruning since the required computation is usually intractable even when there is only one output in regression and two classes in classification. Indeed, the central problem of ensemble pruning research is how to design practical algorithms leading to smaller ensembles without sacrificing or even improving the generalization performance contrasting to all-member ensembles.

During the past decade, many effective ensemble pruning methods have been proposed. Roughly speaking, those methods can be classified into three categories [Tsoumakas et al., 2009]:

- **Ordering-based pruning**. Those methods try to order the individual learners according to some criterion, and only the learners in the front-part will be put into the final ensemble. Though they work in a sequential style, it is noteworthy that they are quite different from sequential ensemble methods (e.g., AdaBoost) since all the available individual learners are given in advance and no more individual learners will be generated in the pruning process; moreover, any individual learner, not just the latest generated one, may be pruned.

- **Clustering-based pruning**. Those methods try to identify a number of representative *prototype* individual learners to constitute the final ensemble. Usually, a clustering process is employed to partition the individual learners into a number of groups, where individual learners in the same group behave similarly while different groups have large diversity. Then, the prototypes of clusters are put into the final ensemble.

- **Optimization-based pruning**. Those methods formulate the ensemble pruning problem as an optimization problem which aims to find the subset of individual learners that maximizes or minimizes an objective related to the generalization ability of the final ensemble. Many optimization techniques have been used, e.g., heuristic optimization methods, mathematical programming methods, etc.

It is obvious that the boundaries between different categories are not crisp, and there are methods that can be put into more than one category. In particular, though there are many early studies on *pure* ordering-based or clustering-based pruning methods, along with the explosively increasing exploitation of optimization techniques in machine learning, recent ordering-based and clustering-based pruning methods become closer to optimization-based methods.

6.4 Ordering-Based Pruning

Ordering-based pruning originated from Margineantu and Dietterich [1997]'s work on *boosting pruning*. Later, most efforts were devoted to pruning ensembles generated by parallel ensemble methods.

Given N individual learners $h_1, \ldots, h_N$, suppose they are combined sequentially in a random order, the generalization error of the ensemble generally decreases monotonically as the ensemble size increases, and approaches an asymptotic constant error. It has been found that [Martínez-Muñoz and Suárez, 2006], however, if an appropriate ordering is devised, the ensemble error generally reaches a minimum with intermediate ensemble size and this minimum is often lower than the asymptotic error, as shown in Figure 6.1. Hence, ensemble pruning can be realized by ordering the N individual learners and then putting the front T individual learners into the final ensemble.

It is generally hard to decide the best T value, but fortunately there are usually many T values that will lead to better performance than the all-member ensemble, and at least the T value can be tuned on training data. A more crucial problem is how to order the individual learners appropri-

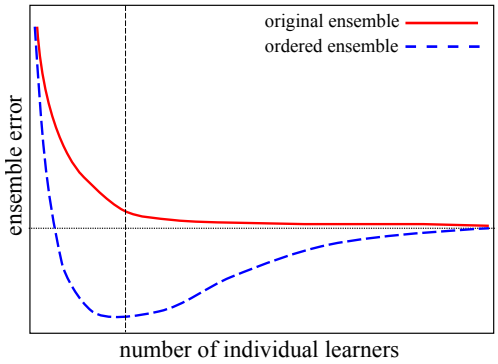

FIGURE 6.1: Illustration of error curves of the original ensemble (aggregated in random order) and ordered ensemble.

ately. During the past decade, many ordering strategies have been proposed. Most of them consider both the accuracy and diversity of individual learners, and a validation data set V with size $|V|$ is usually used (when there are not sufficient data, the training data set D or its sub-samples can be used as validation data). In the following we introduce some representative ordering-based pruning methods.

Reduce-Error Pruning [Margineantu and Dietterich, 1997]. This method starts with the individual learner whose validation error is the smallest. Then, the remaining individual learners are sequentially put into the ensemble, such that the validation error of the resulting ensemble is as small as possible in each round. This procedure is greedy, and therefore, after obtaining the top T individual learners, Margineantu and Dietterich [1997] used Backfitting [Freidman and Stuetzle, 1981] search to improve the ensemble. In each round, it tries to replace one of the already-selected individual learners with an unselected individual learner which could reduce the ensemble error. This process repeats until none of the individual learners can be replaced, or the pre-set maximum number of learning rounds is reached. It is easy to see that Backfitting is time-consuming. Moreover, it was reported [Martínez-Muñoz and Suárez, 2006] that Backfitting could not improve the generalization ability significantly for parallel ensemble methods such as Bagging.

Kappa Pruning [Margineantu and Dietterich, 1997]. This method assumes that all the individual learners have similar performance, and uses the κ_p diversity measure introduced in Section 5.3.1 to calculate the diversity of every pair of individual learners on the validation set. It starts with the pair with the smallest κ_p (i.e., the largest diversity among the given individual learners), and then selects pairs of learners in ascending order of κ_p. Finally, the top T individual learners are put into the ensemble. A variant

method was proposed by Martínez-Muñoz et al. [2009] later, through replacing the pairwise κ_p diversity measure by the interrater agreement diversity measure κ introduced in Section 5.3.2. The variant method still starts with the pair of individual learners that are with the smallest κ value. Then, at the tth round, it calculates the κ value between each unselected individual learner and the current ensemble H_{t-1}, and takes the individual learner with the smallest κ value to construct the ensemble H_t of size t. The variant method often leads to smaller ensemble error. However, it is computationally much more expensive than the original Kappa pruning. Banfield et al. [2005] proposed another variant method that starts with the all-member ensemble and iteratively removes the individual learner with the largest average κ value.

Kappa-Error Diagram Pruning [Margineantu and Dietterich, 1997]. This method is based on the **kappa-error diagram** introduced in Section 5.3.3. It constructs the *convex hull* of the points in the diagram, which can be regarded as a summary of the entire diagram and includes both the most accurate and the most diverse pairs of individual learners. The pruned ensemble consists of any individual learner that appears in a pair corresponding to a point on the convex hull. From the definition of the kappa-error diagram it is easy to see that this pruning method simultaneously considers the accuracy as well as diversity of individual learners.

Complementariness Pruning [Martínez-Muñoz and Suárez, 2004]. This method favors the inclusion of individual learners that are complementary to the current ensemble. It starts with the individual learner whose validation error is the smallest. Then, at the tth round, given the ensemble H_{t-1} of size $t-1$, the complementariness pruning method adds the individual learner h_t which satisfies

$$h_t = \arg\max_{h_k} \sum_{(\boldsymbol{x},y)\in V} \mathbb{I}\left(h_k(\boldsymbol{x}) = y \text{ and } H_{t-1}(\boldsymbol{x}) \neq y\right), \tag{6.14}$$

where V is the validation data set, and h_k is picked up from unselected individual learners.

Margin Distance Pruning [Martínez-Muñoz and Suárez, 2004]. This method defines a **signature vector** for each individual learner. For example, the signature vector $\boldsymbol{c}^{(k)}$ of the kth individual learner h_k is a $|V|$-dimensional vector where the ith element is

$$\boldsymbol{c}_i^{(k)} = 2\mathbb{I}\left(h_k\left(\boldsymbol{x}_i\right) = y_i\right) - 1, \tag{6.15}$$

where $(\boldsymbol{x}_i, y_i) \in V$. Obviously, $\boldsymbol{c}_i^{(k)} = 1$ if and only if h_k classifies $\boldsymbol{x}_i$ correctly and -1 otherwise. The performance of the ensemble can be characterized by the average of $\boldsymbol{c}^{(k)}$'s, i.e., $\bar{\boldsymbol{c}} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{c}^{(k)}$. The ith instance is correctly classified by the ensemble if the ith element of $\bar{\boldsymbol{c}}$ is positive, and the value

of $|\bar{c}_i|$ is the *margin* on the ith instance. If an ensemble correctly classifies all the instances in V, the vector $\bar{\boldsymbol{c}}$ will lie in the first-quadrant of the $|V|$-dimensional hyperspace, that is, every element of $\bar{\boldsymbol{c}}$ is positive. Consequently, the goal is to select the ensemble whose signature vector is near an objective position in the first-quadrant. Suppose the objective position is the point $\boldsymbol{o}$ with equal elements, i.e., $\boldsymbol{o}_i = p\ (i = 1, \ldots, |V|; 0 < p < 1)$. In practice, the value of p is usually a small value (e.g., $p \in (0.05, 0.25)$). The individual learner to be selected is the one which can reduce the distance between $\bar{\boldsymbol{c}}$ and $\boldsymbol{o}$ to the most.

Orientation Pruning [Martínez-Muñoz and Suárez, 2006]. This method uses the signature vector defined as above. It orders the individual learners increasingly according to the angles between the corresponding signature vectors and the *reference direction*, denoted as $\boldsymbol{c}_{ref}$, which is the projection of the first-quadrant diagonal onto the hyperplane defined by the signature vector $\bar{\boldsymbol{c}}$ of the all-member ensemble.

Boosting-Based Pruning [Martínez-Muñoz and Suárez, 2007]. This method uses AdaBoost to determine the order of the individual learners. It is similar to the AdaBoost algorithm except that in each round, rather than generating a base learner from the training data, the individual learner with the lowest weighted validation error is selected from the given individual learners. When the weighted error is larger than 0.5, the **Boosting with restart** strategy is used, that is, the weights are reset and another individual learner is selected. Notice that, the weights are used in the ordering process, while Martínez-Muñoz and Suárez [2007] reported that there is no significant difference for the pruned ensemble to make prediction with/without the weights.

Reinforcement Learning Pruning [Partalas et al., 2009]. This method models the ensemble pruning problem as an episodic task. Given N individual learners, it assumes that there is an agent which takes N sequential actions each corresponding to either including the individual learner h_k in the final ensemble or not. Then, the Q-learning algorithm [Watkins and Dayan, 1992], a famous **reinforcement learning** technique, is applied to solve an optimal policy of choosing the individual learners.

6.5 Clustering-Based Pruning

An intuitive idea to ensemble pruning is to identify some *prototype* individual learners that are representative yet diverse among the given individual learners, and then use only these prototypes to constitute the ensemble. This category of methods is called as clustering-based pruning because

the most straightforward way to identify the prototypes is to use clustering techniques.

Generally, clustering-based pruning methods work in two steps. In the first step, the individual learners are grouped into a number of clusters. Different clustering techniques have been exploited for this purpose. For example, Giacinto et al. [2000] used **hierarchical agglomerative clustering** and regarded the probability that the individual learners do not make coincident validation errors as the distance; Lazarevic and Obradovic [2001] used k**-means clustering** based on Euclidean distance; Bakker and Heskes [2003] used **deterministic annealing** for clustering; etc.

In the second step, prototype individual learners are selected from the clusters. Different strategies have been developed. For example, Giacinto et al. [2000] selected from each cluster the learner which is the most distant to other clusters; Lazarevic and Obradovic [2001] iteratively removed individual learners from the least to the most accurate inside each cluster until the accuracy of the ensemble starts to decrease; Bakker and Heskes [2003] selected the centroid of each cluster; etc.

6.6 Optimization-Based Pruning

Optimization-based pruning originated from [Zhou et al., 2002b] which employs a **genetic algorithm** [Goldberg, 1989] to select individual learners for the pruned ensemble. Later, many other optimization techniques, including heuristic optimization, mathematical programming and probabilistic methods have been exploited. This section introduces several representative methods.

6.6.1 Heuristic Optimization Pruning

Recognizing that the theoretically optimal solution to weighted combination in (4.14) is infeasible in practice, Zhou et al. [2002b] regarded the ensemble pruning problem as an optimization task and proposed a practical method GASEN.

The basic idea is to associate each individual learner with a weight that could characterize the goodness of including the individual learner in the final ensemble. Given N individual learners, the weights can be organized as an N-dimensional weight vector, where small elements in the weight vector suggest that the corresponding individual learners should be excluded. Thus, one weight vector corresponds to one solution to ensemble pruning. In GASEN, a set of weight vectors are randomly initialized at first. Then, a genetic algorithm is applied to the *population* of weight vectors, where the

fitness of each weight vector is calculated based on the corresponding ensemble performance on validation data. The pruned ensemble is obtained by decoding the optimal weight vector evolved from the genetic algorithm, and excluding individual learners associated with small weights.

There are different GASEN implementations, by using different coding schemes or different genetic operators. For example, Zhou et al. [2002b] used a *floating coding* scheme, while Zhou and Tang [2003] used a *bit coding* scheme which directly takes 0-1 weights and avoids the problem of setting an appropriate threshold to decide which individual learner should be excluded.

In addition to genetic algorithms [Coelho et al., 2003], many other heuristic optimization techniques have been used in ensemble pruning. For example, greedy hill-climbing [Caruana et al., 2004], artificial immune algorithms [Castro et al., 2005, Zhang et al., 2005], case similarity search [Coyle and Smyth, 2006], etc.

6.6.2 Mathematical Programming Pruning

One deficiency of heuristic optimization is the lack of solid theoretical foundations. Along with the great success of using mathematical programming in machine learning, ensemble pruning methods based on mathematical programming optimization have been proposed.

6.6.2.1 SDP Relaxation

Zhang et al. [2006] formulated ensemble pruning as a quadratic integer programming problem. Since finding the optimal solution is computationally infeasible, they provided an approximate solution by **Semi-Definite Programming** (SDP).

First, given N individual classifiers and m training instances, Zhang et al. [2006] recorded the errors in the matrix $\mathbf{P}$ as

$$P_{ij} = \begin{cases} 0 & \text{if } h_j \text{ classifies } \boldsymbol{x}_i \text{ correctly} \\ 1 & \text{otherwise.} \end{cases} \tag{6.16}$$

Let $\mathbf{G} = \mathbf{P}^\top \mathbf{P}$. Then, the diagonal element G_{ii} is the number of mistakes made by h_i, and the off-diagonal element G_{ij} is the number of co-occurred mistakes of h_i and h_j. The matrix elements are normalized according to

$$\tilde{G}_{ij} = \begin{cases} \frac{G_{ij}}{m} & i = j \\ \frac{1}{2}\left(\frac{G_{ij}}{G_{ii}} + \frac{G_{ji}}{G_{jj}}\right) & i \neq j\,. \end{cases} \tag{6.17}$$

Thus, $\sum_{i=1}^{N} \tilde{G}_{ii}$ measures the overall performance of the individual classifiers, $\sum_{i,j=1;i\neq j}^{N} \tilde{G}_{ij}$ measures the diversity, and a combination of these two terms $\sum_{i,j=1}^{N} \tilde{G}_{ij}$ is a good approximation of the ensemble error.

Consequently, the ensemble pruning problem is formulated as the quadratic integer programming problem

$$\min_{\boldsymbol{z}} \ \boldsymbol{z}^\top \tilde{\mathbf{G}} \boldsymbol{z} \quad \text{s.t.} \ \sum_{i=1}^{N} z_i = T, \ z_i \in \{0,1\}\,, \tag{6.18}$$

where the binary variable z_i represents whether the ith classifier h_i is included in the ensemble, and T is the size of the pruned ensemble.

(6.18) is a standard 0-1 optimization problem, which is generally NP-hard. However, let $v_i = 2z_i - 1 \in \{-1, 1\}$,

$$\mathbf{V} = \boldsymbol{v}\boldsymbol{v}^\top, \ \ \mathbf{H} = \begin{pmatrix} \mathbf{1}^\top \tilde{\mathbf{G}} \mathbf{1} & \mathbf{1}^\top \tilde{\mathbf{G}} \\ \tilde{\mathbf{G}} \mathbf{1} & \tilde{\mathbf{G}} \end{pmatrix}, \text{ and } \mathbf{D} = \begin{pmatrix} N & \mathbf{1}^\top \\ \mathbf{1} & \mathbf{I} \end{pmatrix}, \tag{6.19}$$

where $\mathbf{1}$ is all-one column vector and $\mathbf{I}$ is identify matrix, then (6.18) can be rewritten as the equivalent formulation [Zhang et al., 2006]

$$\begin{aligned} \min_{\mathbf{V}} \ & \mathbf{H} \otimes \mathbf{V} \\ \text{s.t.} \ & \mathbf{D} \otimes \mathbf{V} = 4T, \ \mathrm{diag}(\mathbf{V}) = \mathbf{1}, \ \mathbf{V} \succeq 0 \\ & \mathrm{rank}(\mathbf{V}) = 1\,, \end{aligned} \tag{6.20}$$

where $\mathbf{A} \otimes \mathbf{B} = \sum_{ij} A_{ij} B_{ij}$. Then, by dropping the rank constraint, it is relaxed to the following convex SDP problem which can be solved in polynomial time [Zhang et al., 2006]

$$\begin{aligned} \min_{\mathbf{V}} \ & \mathbf{H} \otimes \mathbf{V} \\ \text{s.t.} \ & \mathbf{D} \otimes \mathbf{V} = 4T, \ \mathrm{diag}(\mathbf{V}) = \mathbf{1}, \ \mathbf{V} \succeq 0\,. \end{aligned} \tag{6.21}$$

6.6.2.2 ℓ_1-Norm Regularization

Li and Zhou [2009] proposed a regularized selective ensemble method RSE which reduces the ensemble pruning task to a **Quadratic Programming** (QP) problem.

Given N individual classifiers and considering weighted combination, RSE determines the weight vector $\boldsymbol{w} = [w_1, \ldots, w_N]^\top$ by minimizing the regularized risk function

$$R(\boldsymbol{w}) = \lambda V(\boldsymbol{w}) + \Omega(\boldsymbol{w})\,, \tag{6.22}$$

where $V(\boldsymbol{w})$ is the empirical loss which measures the misclassification on training data $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, $\Omega(\boldsymbol{w})$ is the *regularization* term which tries to make the final classifier smooth and simple, and λ is a *regularization* parameter which trades off the minimization of $V(\boldsymbol{w})$ and $\Omega(\boldsymbol{w})$.

By using the *hinge loss* and *graph Laplacian* regularizer as the empirical loss and *regularization* term, respectively, the problem is formulated as [Li

and Zhou, 2009]

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \mathbf{P}\mathbf{L}\mathbf{P}^\top \boldsymbol{w} + \lambda \sum_{i=1}^{m} \max(0, 1 - y_i \boldsymbol{p}_i^\top \boldsymbol{w}) \\ \text{s.t.} \quad & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0} \end{aligned} \tag{6.23}$$

where $\boldsymbol{p}_i = (h_1(\boldsymbol{x}_i), \ldots, h_N(\boldsymbol{x}_i))^\top$ encodes the predictions of individual classifiers on $\boldsymbol{x}_i$, $\mathbf{P} \in \{-1, +1\}^{N \times m}$ is the *prediction matrix* which collects predictions of all individual classifiers on all training instances, where $P_{ij} = h_i(\boldsymbol{x}_j)$. $\mathbf{L}$ is the *normalized graph Laplacian* of the neighborhood graph G of the training data. Denote the weighted adjacency matrix of G by $\mathbf{W}$, and $\mathbf{D}$ is a diagonal matrix where $D_{ii} = \sum_{j=1}^{m} W_{ij}$. Then, $\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}$.

By introducing slack variables $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\top$, (6.23) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \boldsymbol{w}^\top \mathbf{P}\mathbf{L}\mathbf{P}^\top \boldsymbol{w} + \lambda \, \mathbf{1}^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & y_i \boldsymbol{p}_i^\top \boldsymbol{w} + \xi_i \geq 1, \quad i = 1, \ldots, m \\ & \mathbf{1}^\top \boldsymbol{w} = 1, \quad \boldsymbol{w} \geq \mathbf{0}, \quad \boldsymbol{\xi} \geq \mathbf{0} \, . \end{aligned} \tag{6.24}$$

Obviously, (6.24) is a standard QP problem that can be efficiently solved by existing optimization packages.

Notice that $\mathbf{1}^\top \boldsymbol{w} = 1, \boldsymbol{w} \geq \mathbf{0}$ is a ℓ_1-norm constraint on the weights $\boldsymbol{w}$. The ℓ_1-norm is a sparsity-inducing constraint which will force some w_i's to be zero, and thus, RSE favors ensemble with small sizes and only a subset of the given individual learners will be included in the final ensemble.

Another advantage of RSE is that it naturally fits the **semi-supervised learning** setting due to the use of the graph Laplacian regularizer, hence it can exploit *unlabeled data* to improve ensemble performance. More information on semi-supervised learning will be introduced in Chapter 8.

6.6.3 Probabilistic Pruning

Chen et al. [2006, 2009] proposed a probabilistic pruning method under the Bayesian framework by introducing a sparsity-inducing prior over the combination weights, where the *maximum a posteriori* (MAP) estimation of the weights is obtained by **Expectation Maximization** (EM) [Chen et al., 2006] and **Expectation Propagation** (EP) [Chen et al., 2009], respectively. Due to the sparsity-inducing prior, many of the posteriors of the weights are sharply distributed at zero, and thus many individual learners are excluded from the final ensemble.

Given N individual learners $h_1, \ldots, h_N$, the output vector of the individual learners on the instance $\boldsymbol{x}$ is $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \ldots, h_N(\boldsymbol{x}))^\top$. The output of

the all-member ensemble is $H(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{h}(\boldsymbol{x})$, where $\boldsymbol{w} = [w_1, \ldots, w_N]^\top$ is a non-negative weight vector, $w_i \geq 0$.

To make the weight vector $\boldsymbol{w}$ sparse and nonnegative, a left-truncated *Gaussian prior* is introduced to each weight w_i [Chen et al., 2006], that is,

$$p(\boldsymbol{w} \mid \boldsymbol{\alpha}) = \prod_{i=1}^{N} p(w_i \mid \alpha_i) = \prod_{i=1}^{N} \mathcal{N}_t(w_i \mid 0, \alpha_i^{-1}), \tag{6.25}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^\top$ is the inverse variance of weight vector $\boldsymbol{w}$ and $\mathcal{N}_t(w_i \mid 0, \alpha_i^{-1})$ is a left-truncated Gaussian distribution defined as

$$\mathcal{N}_t(w_i \mid 0, \alpha_i^{-1}) = \begin{cases} 2N(w_i \mid 0, \alpha_i^{-1}) & \text{if } w_i \geq 0 \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{6.26}$$

For regression, it is assumed that the ensemble output is corrupted by a Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ with mean zero and variance σ^2. That is, for each training instance $(\boldsymbol{x}_i, y_i)$, it holds that

$$y_i = \boldsymbol{w}^\top \boldsymbol{h}(\boldsymbol{x}_i) + \epsilon_i \ . \tag{6.27}$$

Assuming *i.i.d.* training data, the likelihood can be expressed as

$$p(\boldsymbol{y} \mid \boldsymbol{w}, \mathbf{X}, \sigma^2) = (2\pi\sigma^2)^{-m/2} \exp\left\{ -\frac{1}{2\sigma^2} \| \boldsymbol{y}^\top - w^\top \mathbf{H} \| \right\} \ , \tag{6.28}$$

where $\boldsymbol{y} = [y_1, \ldots, y_m]^\top$ is the ground-truth output vector, and $\mathbf{H} = [\boldsymbol{h}(\boldsymbol{x}_1), \ldots, \boldsymbol{h}(\boldsymbol{x}_m)]$ is an $N \times m$ matrix which collects all the predictions of individual learners on all the training instances. Consequently, the posterior of $\boldsymbol{w}$ can be written as

$$p(\boldsymbol{w} \mid \mathbf{X}, \boldsymbol{y}, \boldsymbol{\alpha}) \propto \prod_{i=1}^{N} p(w_i \mid \alpha_i) \prod_{i=1}^{m} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \ . \tag{6.29}$$

As defined in (6.26), the prior over $\boldsymbol{w}$ is a left-truncated Gaussian, and therefore, exact Bayesian inference is intractable. However, the EM algorithm or EP algorithm can be employed to generate an MAP solution, leading to an approximation of the sparse weight vector [Chen et al., 2006, 2009].

For classification, the ensemble output is formulated as

$$H(\boldsymbol{x}) = \Phi\left(\boldsymbol{w}^\top \boldsymbol{h}(\boldsymbol{x})\right) , \tag{6.30}$$

where $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(t \mid 0, 1) dt$ is the Gaussian cumulative distribution function. The class label of $\boldsymbol{x}$ is $+1$ if $H(\boldsymbol{x}) \geq 1/2$ and 0 otherwise. As above, the posterior of $\boldsymbol{w}$ can be derived as

$$p(\boldsymbol{w} \mid \mathbf{X}, \boldsymbol{y}, \boldsymbol{\alpha}) \propto \prod_{i=1}^{N} p(w_i \mid \alpha_i) \prod_{i=1}^{m} \Phi(y_i \boldsymbol{w}^\top \boldsymbol{h}(\boldsymbol{x}_i)), \tag{6.31}$$

where both the prior $p(w_i \mid \alpha_i)$ and the likelihood $\Phi(y_i \boldsymbol{w}^\top \boldsymbol{h}(\boldsymbol{x}_i))$ are non-Gaussian, and thus, the EM algorithm or the EP algorithm is used to obtain an MAP estimation of the sparse weight vector [Chen et al., 2006, 2009].

6.7 Further Readings

Tsoumakas et al. [2009] provided a brief review on ensemble pruning methods. Hernández-Lobato et al. [2011] reported a recent empirical study which shows that optimization-based and ordering-based pruning methods, at least for pruning parallel regression ensembles, generally outperform ensembles generated by AdaBoost.R2, Negative Correlation and several other approaches.

In addition to clustering, there are also other approaches for selecting the prototype individual learners, e.g., Tsoumakas et al. [2004, 2005] picked prototype individual learners by using statistical tests to compare their individual performance. Hernández-Lobato et al. [2009] proposed the *instance-based pruning* method, where the individual learners selected for making prediction are determined for each instance separately. Soto et al. [2010] applied the instance-based pruning to pruned ensembles generated by other ensemble pruning methods, yielding the *double pruning* method. A similar idea has been described by Fan et al. [2002].

If each individual learner is viewed as a fancy feature extractor [Kuncheva, 2008, Brown, 2010], it is obvious that ensemble pruning has close relation to **feature selection** [Guyon and Elisseeff, 2003] and new ensemble pruning methods can get inspiration from feature selection techniques. It is worth noting, however, that the different natures of ensemble pruning and feature selection must be considered. For example, in ensemble pruning the individual learners predict the same target and thus have the same physical meaning, while in feature selection the features usually have different physical meanings; the individual learners are usually highly correlated, while this may not be the case for features in feature selection. A breakthrough in computer vision of the last decade, i.e., the *Viola-Jones face detector* [Viola and Jones, 2004], actually can be viewed as a pruning of Harr-feature-based decision stump ensemble, or selection of Harr features by AdaBoost with a cascade architecture.