

Combination of classifiers

Nicolás García-Pedrajas

Computational Intelligence and Bioinformatics Research Group

November 3, 2021



Table of contents

Output coding

Ensembles of classifiers

- Bagging and related methods

- Boosting and related methods

Other methods

- Stacking

- Cascading

- Mixture of experts

Summary



Output coding



Dealing with more than two classes

- ➡ Some classifiers, such as SVM, only address two-class problems
- ➡ Other classifiers might be more effective in two-class problems
- ➡ How to deal with N -class problems?
- ➡ Transform a N class problem into M , $M \geq N$ two-class problems
- ➡ Output coding
 - ✓ One – vs – all
 - ✓ One – vs – one
 - ✓ Error correcting output codes



One-vs-all (OVA)

- ➡ Transform a N -class problem into N binary problems
- ➡ For classifier T
 - ✓ Positive class: T
 - ✓ Negative class: $\{1, 2, \dots, T-1, T+1, \dots, N\}$
- ➡ Testing stage
 - ✓ Output of N classifiers
 - ✓ Classifier with highest output assigns the class
- ➡ Performance
 - ✓ Usually considered the worst
 - ✓ Not always (Rifkin and Klautau, 2004)



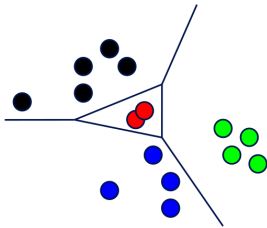
One-vs-all

- ➡ Assumption: Each class can be separated from all the rest using a binary classifier in the hypothesis space
- ➡ Not always possible to learn
- ➡ No theoretical justification
- ➡ Also: need to make sure the range of all classifiers is the same (for the argmax)
- ➡ Note: in high dimensional spaces, it's likely that things are separable

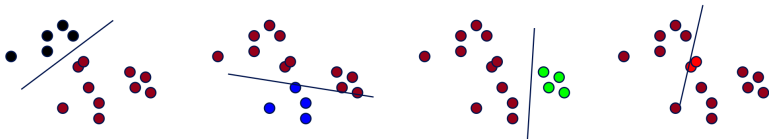


One-vs-all

Actual problem



Decomposition



One-vs-one (OVO) or all-vs-all (AVA)

- ➡ Transform a N -class problem into $N(N - 1)/2$ two-class problems
- ➡ For every pair of classes (i, j) train a classifier to distinguish i from j , disregarding the remaining classes
- ➡ For testing
 - ✓ Each classifier (i, j) cast a vote for either i or j
 - ✓ Majority voting
- ➡ Problems
 - ✓ Some classifiers are trained with few instances
 - ✓ Incompetent classifiers
- ➡ Pros
 - ✓ Faster
 - ✓ Usually performs well



Error correcting output codes (ECOC)(Dietterich and Bakiri, 1995)

- ➡ Based on correcting codes
- ➡ Codifies each class using a codeword
- ➡ Each column of the coding matrix is a classifier

Coding matrix

Class	Classifier 1	Classifier 2	Classifier 3
1	1	1	-1
2	-1	1	-1
3	1	-1	1
4	1	1	1
5	1	1	-1
6	-1	-1	1

- ➡ Sparse matrix vs. Dense matrix
- ➡ Can generalize previous methods



Error correcting output codes

- ➡ Testing stage
- ➡ Outputs of each classifier: 0/1
 - ✓ Instance assigned to class with minimum Hamming distance
- ➡ Output of each classifier: Real
 - ✓ Instance assigned to classifier with minimum Euclidean distance
- ➡ Both methods perform very similar
- ➡ Binary outputs
 - ✓ Less info, less sensitive to noise or a bad classifier
- ➡ Continuous outputs
 - ✓ More info, more sensitive to noise or a bad classifier



Sparse matrices vs. dense matrices

Coding dense matrix

Class	Classifier 1	Classifier 2	Classifier 3
1	1	1	-1
2	-1	1	-1
3	1	-1	1
4	1	1	1
5	1	1	-1
6	-1	-1	1

Coding sparse matrix

Class	Classifier 1	Classifier 2	Classifier 3
1	0	1	-1
2	-1	1	-1
3	0	-1	0
4	1	1	1
5	0	1	-1
6	-1	0	1

0 means ignore



ECOC generalizes one-vs-all

OVA coding matrix

Class	1 vs. rest	2 vs. rest	3 vs. rest	4 vs. rest	5 vs. rest	6 vs. rest
1	1	-1	-1	-1	-1	-1
2	-1	1	-1	-1	-1	-1
3	-1	-1	1	-1	-1	-1
4	-1	-1	-1	1	-1	-1
5	-1	-1	-1	-1	1	-1
6	-1	-1	-1	-1	-1	1

ECOC generalizes one-vs-one

OVO coding matrix

Class	1 vs. 2	1 vs. 3	1 vs. 4	2 vs. 3	2 vs. 4	3 vs. 4
1	1	1	1	0	0	0
2	-1	0	0	1	1	0
3	0	-1	0	-1	0	1
4	0	0	-1	0	-1	-1

Ensembles of classifiers



Ensembles of classifiers

- ➡ Definition: “An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples”(Dietterich, 2000)
- ➡ Construct a set of classifiers from the training data
- ➡ Predict class label of previously unseen records by aggregating predictions made by multiple classifiers



Ensembles of classifier

Basic idea: Learn a set of classifiers (experts) and to allow them to vote

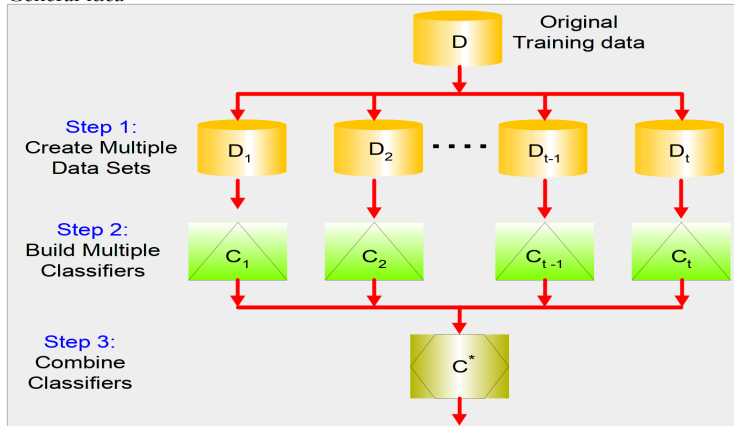
Advantage: improvement in predictive accuracy.

Disadvantage: it is difficult to understand an ensemble of classifiers.



Ensembles of classifier

General idea



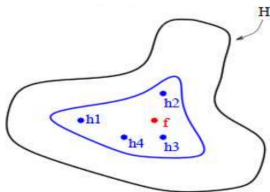
Why do ensembles work?(Dietterich, 2000)

- ➡ Dietterich (2000) showed that ensembles overcome three problems:
 - ✓ The **Statistical Problem** arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!
 - ✓ The **Computational Problem** arises when the learning algorithm cannot guarantee finding the best hypothesis
 - ✓ The **Representational Problem** arises when the hypothesis space does not contain any good approximation of the target class(es)
- ➡ The statistical problem and computational problem result in the variance component of the error of the classifiers
- ➡ The representational problem results in the bias component of the error of the classifiers

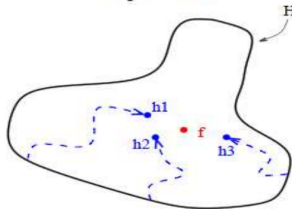


Why do ensembles work?(Dietterich, 2000)

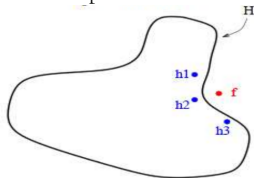
Statistical



Computational



Representational



Why do ensembles work?

- ➡ Suppose there are 25 base classifiers
 - ✓ Each classifier has error rate, $\epsilon = 0.35$
 - ✓ Assume classifiers are independent
 - ✓ Probability that the ensemble classifier makes a wrong prediction

$$\sum_{i=1}^{25} \binom{25}{i} \epsilon^i (1 - \epsilon)^{25-i} = 0.06$$

- ➡ The problem is: Classifiers are not independent



Taxonomy of ensembles of classifiers

- ▶ There are many taxonomies. Rokach (2009) proposed four dimensions:
 - ✓ Training set - How the training sets are generated
 - Independent vs. dependent (cooperative) methods
 - ✓ Inducer – The basic learning methods
 - Homogeneous vs. heterogeneous ensembles
 - ✓ Ensemble generator – This component is responsible for generating the diverse classifiers.
 - ✓ Combiner - The combiner is responsible for combining the classifications of the various classifiers.

These dimensions are partly orthogonal



Methods for Independently Constructing Ensembles

- ➡ One way to force a learning algorithm to construct multiple hypotheses is to run the algorithm several times with a mechanism to provide diversity
- ➡ This idea is used in the following methods:
 - ✓ Randomness Injection
 - ✓ Bagging
 - ✓ Feature sampling
 - ✓ Error-Correcting Output Coding (already visited)



Randomization Injection

- ➡ Combine several classifiers of the same type with the same dataset
- ➡ Inject some randomization into a standard learning algorithm (usually easy):
 - ✓ Neural network: random initial weights
 - ✓ Decision tree: when splitting, choose one of the top N attributes at random (uniformly)
- ➡ First ensembles were of this type



Bagging

- ▶ Employs simplest way of combining predictions that belong to the same type
- ▶ Combining can be realized with voting or averaging
- ▶ Each model receives equal weight
- ▶ Common version of bagging:
 - ✓ Sample several training sets of size n (instead of just having one training set of size n)
 - ✓ Build a classifier for each training set
 - ✓ Combine the classifier's predictions (more follows on this...)
- ▶ This improves performance in almost all cases if learning scheme is unstable (i.e. decision trees)



Bagging algorithm

Listing: Training stage

```
Let  $n$  be the size of the training set.  
For each of  $t$  iterations:  
    Sample  $n$  instances with replacement from the training set.  
    Apply the learning algorithm to the sample.  
    Store the resulting classifier.
```

Listing: Testing stage (majority voting)

```
For each of the  $t$  classifiers:  
    Predict class of instance using classifier.  
Return class that was predicted most often.
```



Why does Bagging work?

- ➡ Bagging reduces variance by voting/averaging, thus reducing the overall expected error
 - ✓ In the case of classification there are pathological situations where the overall error might increase
 - ✓ Usually, the more classifiers the better
- ➡ Variance term of the error is usually smaller



Feature sampling

- ➡ Key idea: Provide a different subset of the input features in each call of the learning algorithm
- ➡ Random subspace method(Ho, 1998)
 - ✓ Each classifier is trained with a subset of features (original paper: 50% of them)
 - ✓ Combination is carried out using majority voting
 - ✓ Performance is usually very problem dependent



Random forest

- ➡ Ensemble specifically designed for decision trees
- ➡ Enforces diversity (less focused on accuracy)
 - ✓ Large ensemble (typically 100/500 trees)
- ➡ Introduces randomness on each node calculation
 - ✓ Bootstrap sampling of instances on each node
 - ✓ Random sampling of features on each node



Random forest

1. For $b = 1$ to B :
 - (a) Draw a **bootstrap sample** \mathbf{Z}^* of size N from the training data.
 - (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
 - i. Select **m variables at random** from the p variables.
 - ii. Pick the best variable/split-point among the m .
 - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

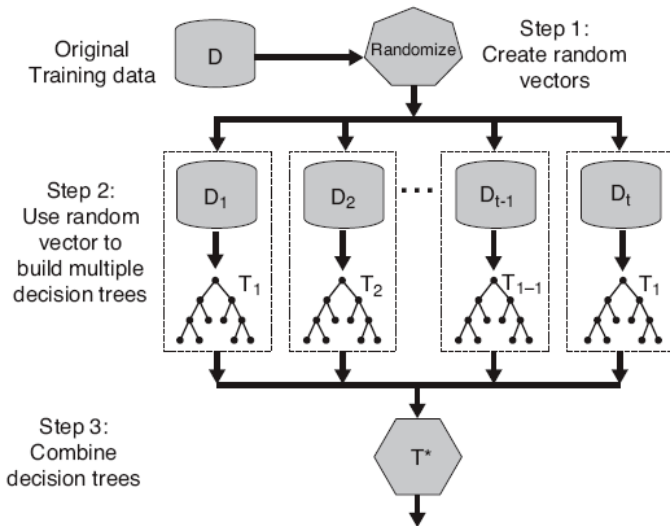
To make a prediction at a new point x :

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the b th random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$.



Random forest



Coordinated Construction of Ensembles

Key idea

To learn complementary classifiers so that instance classification is realized by taking an weighted sum of the classifiers.

➡ Basic method: Boosting



Boosting

- ➡ Iterative procedure: new models are influenced by performance of previously built ones
 - ✓ New model is encouraged to become expert for instances classified incorrectly by earlier models
 - ✓ Intuitive justification: models should be experts that complement each other
- ➡ Also uses voting/averaging but models are weighted according to their performance (why?)
- ➡ There are many variants of this algorithm



Boosting

- ➡ An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - ✓ Each instance is assigned a weight that measures its probability of being sampled
 - ✓ Initially, all N records are assigned equal weights, $1/N$
 - ✓ Records that are wrongly classified will have their weights increased
 - ✓ Records that are classified correctly will have their weights decreased
- ➡ Unlike bagging, weights change at the end of boosting round



AdaBoost

- ➡ Most common method of boosting
- ➡ The ensemble is formed by T classifiers: C_1, C_2, \dots, C_T
- ➡ Error of each classifier is

$$\epsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(\mathbf{x}_j) \neq y_j)$$

where w_j is the weight of instance \mathbf{x}_j , and $\delta(p)$ is 1 if predicate p is true

- ➡ Each classifier is weighted

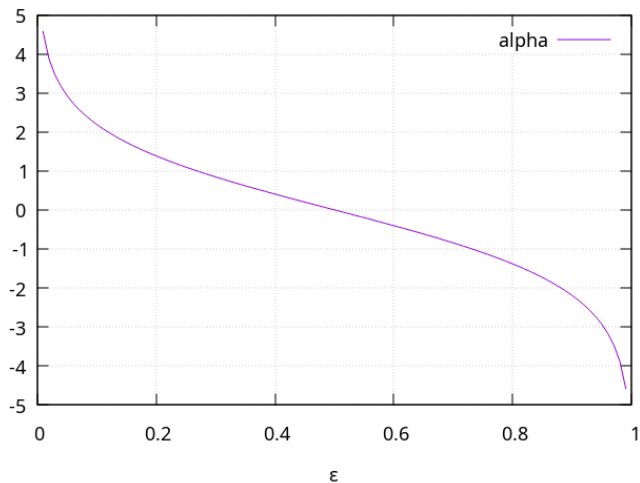
$$\alpha_i = \frac{1}{2} \log \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$

- ➡ Weights can be fed to the classifiers or used to sample the dataset



AdaBoost

The weight of a classifier depends on its accuracy



AdaBoost

Weight update

$$w_i^{(j+1)} = w_i^{(j-1)} \begin{cases} \exp^{-\alpha_j}, & \text{if } C_j(\mathbf{x}_i) = y_i \\ \exp^{\alpha_j}, & \text{if } C_j(\mathbf{x}_i) \neq y_i \end{cases}$$

After update \mathbf{w} is normalized to module 1

If any intermediate rounds produce error rate higher than 50% process is stopped. In a common variant the weights are reverted back to $1/N$ and the resampling procedure is repeated

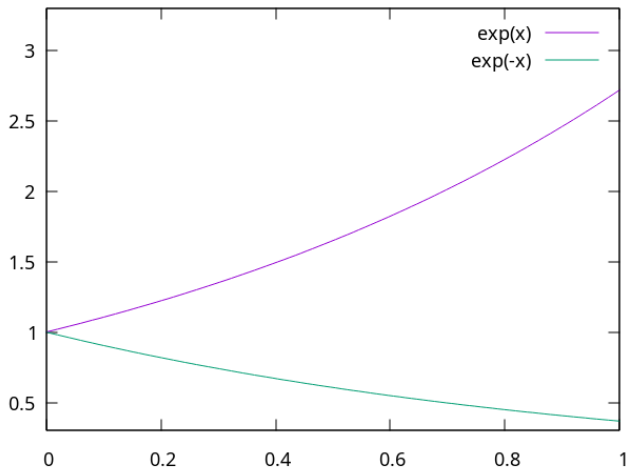
Classification:

$$C^*(\mathbf{x}) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(\mathbf{x}) = y)$$



AdaBoost

Weight update depends on the accuracy of the classifier



Discrete AdaBoost ($h_t : \mathbb{X} \rightarrow \{-1, 1\}$)(Webb, 2000)

Given: $(x_1, y_1), \dots, (x_m, y_m), x_i \in \mathbb{X}, y_i \in \{-1, 1\}$

Initialize weights $w_1^{(i)} = 1/m$

For $t = 1, \dots, T$

1. Call WeakLearn, which returns the weak classifier $h_t : \mathbb{X} \rightarrow \{-1, 1\}$ with minimum error w.r.t. distribution \mathbf{w}_t
2. Compute error $\epsilon_t = \frac{\sum_i w_t^{(i)} I(y_i \neq h_t(x_i))}{\sum_i w_i}$, and obtain $\alpha_t = \log(\frac{1-\epsilon_t}{\epsilon_t})$
3. Update weights

$$w_{t+1}^{(i)} = \frac{w_t^{(i)} \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor chosen so that \mathbf{w}_{t+1} is a distribution

Output the strong classifier

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Variants of AdaBoost

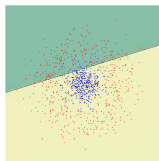
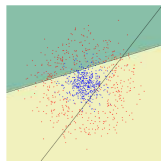
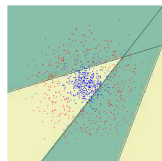
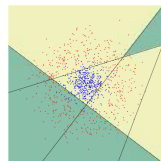
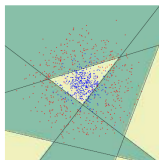
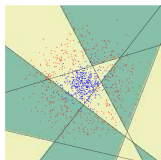
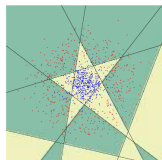
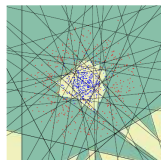
- ➡ AdaBoost.M1 for multi-class problems ($h_t : \mathbb{X} \rightarrow \{0, 1, \dots, k\}$)

$$w_{t+1}^{(i)} = w_t^{(i)} \cdot \exp(\alpha_t I(y_i \neq h_t(\mathbf{x}_i)))$$

- ➡ AdaBoost.M2 for multi-label problems ($h_t : \mathbb{X} \rightarrow [0, 1]^k$)
- ➡ AdaBoost.R for real valued outputs ($h_t : \mathbb{X} \rightarrow [0, 1]$)



AdaBoost at work

 $t = 1$  $t = 2$  $t = 3$  $t = 4$  $t = 5$  $t = 6$  $t = 7$  $t = 40$

Boosting

- ➡ Boosting can be applied without weights using re-sampling with probability determined by weights
- ➡ Boosting decreases exponentially the training error in the number of iterations
- ➡ Boosting works well if base classifiers are not too complex and their error does not become too large too quickly
- ➡ Boosting reduces the bias component of the error of simple classifiers



Bagging, boosting and bias/variance error

- Bagging: Reduces variance of the error
- Boosting: Reduces both bias and variance of the error

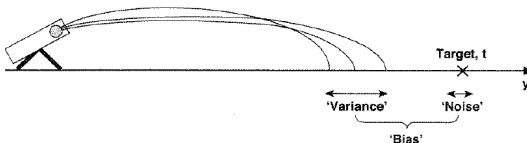
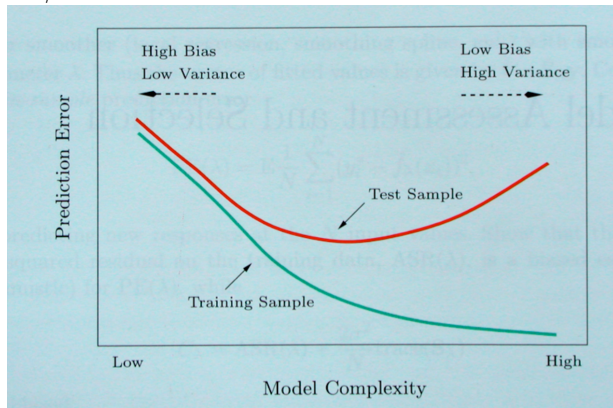


Figure 5.32. Bias-variance decomposition.

Bagging, boosting and bias/variance error

Bias/variance tradeoff



Success of ensembles

- ▀ Individual members must be accurate
 - ✓ Accuracy must be enforced
- ▀ Individual members must be different
 - ✓ Diversity must be enforced
- ▀ Diversity/accuracy are in conflict
 - ✓ Members of the ensemble must be diverse when they are wrong



Diversity vs. accuracy

➡ Two key aspects

- ✓ An ensemble of classifiers must be more accurate than any of its individual members
- ✓ The individual classifiers composing an ensemble must be accurate and diverse
 - An “accurate” classifier is one that has an error rate better than random when guessing new examples
 - Two classifiers are diverse if **they make different errors** on new data points

➡ Accuracy and diversity are in conflict



Other methods

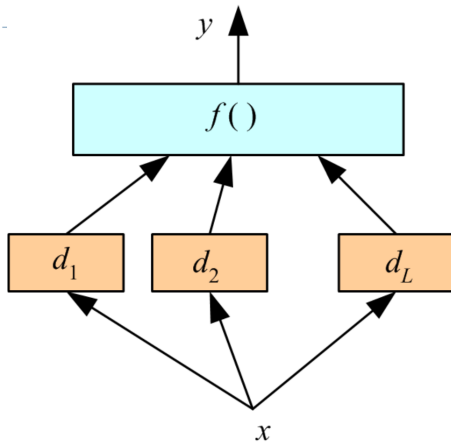


Stacking

- ▶ In stacked generalization, the combiner $f()$ is another learner and is not restricted to being a linear combination as in voting
- ▶ Uses meta learner instead of voting to combine predictions of base learners
 - ✓ Predictions of base learners (level-0 models) are used as input for meta learner (level-1 model)
 - ✓ Base learners usually different learning schemes
- ▶ Hard to analyze theoretically: “black magic”



Stacking

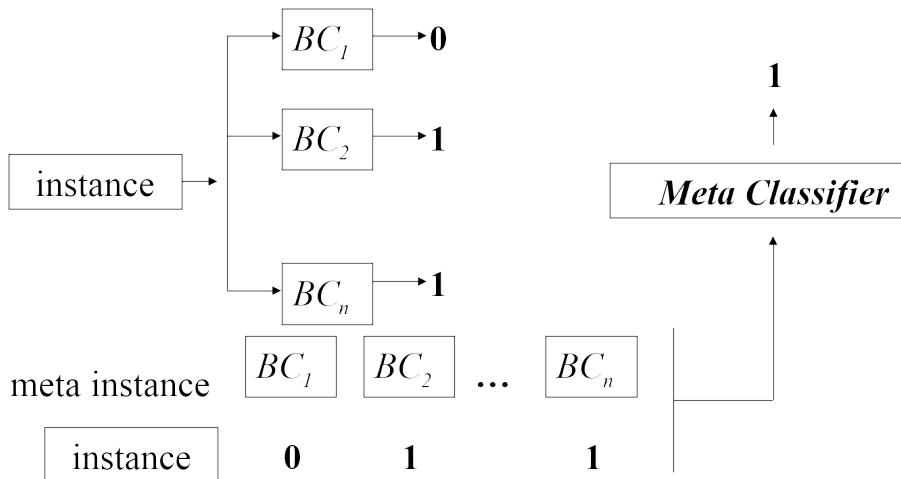


Stacking

- ➡ The combiner system should learn how the base learners make errors
- ➡ Stacking is a means of estimating and correcting for the biases of the base-learners.
- ➡ Therefore, the combiner should be trained on data unused in training the base-learners
 - ✓ Otherwise the best classifier is always preferred



Stacking



Stacking

- ➡ Predictions on training data can't be used to generate data for level-1 model.
 Level-0 classifier that better fit training data will be chosen by the level-1 model
- ➡ k -fold cross-validation-like scheme is employed

<i>train</i>	<i>train</i>	<i>test</i>
<i>train</i>	<i>test</i>	<i>train</i>
<i>test</i>	<i>train</i>	<i>train</i>
<i>test</i>	<i>test</i>	<i>test</i>

Meta Data

Stacking

- ➡ If base learners can output probabilities it is better to use those as input to meta learner
- ➡ Which algorithm to use to generate meta learner?
 - ✓ In principle, any learning scheme can be applied
 - ✓ David Wolpert: “relatively global, smooth” model
 - Base learners do most of the work
 - Reduces risk of overfitting

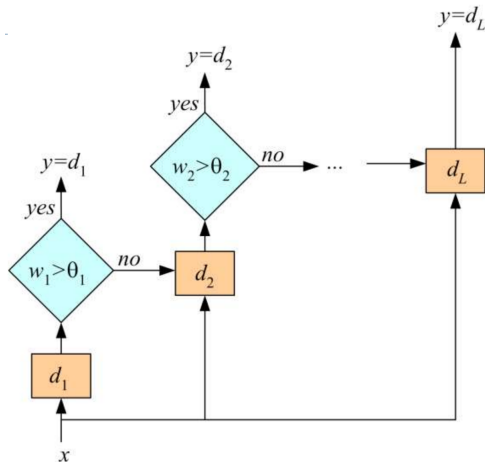


Cascading

- ➡ Cascade learners in order of complexity
- ➡ Use d_j only if preceding ones are not confident



Cascading general structure



Cascading

- ▶ Cascading is a multistage method, and we use d_j only if all preceding learners are not confident
- ▶ Associated with each learner is a confidence w_j such that we say d_j is confident of its output and can be used if $w_j > \theta_j$ (the threshold)
- ▶ Confident: misclassifications as well as the instances for which the posterior is not high enough.
- ▶ Important: The idea is that an early simple classifier handles the majority of instances, and a more complex classifier is used only for a small percentage, so does not significantly increase the overall complexity



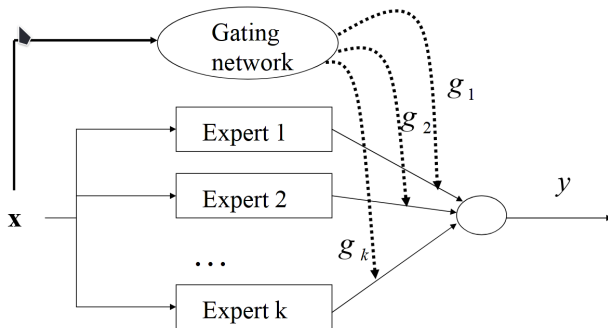
Mixture of experts

- ⇒ Ensemble methods
 - ✓ Use a combination of simpler learners to improve predictions
- ⇒ Mixture of expert model
 - ✓ Covers different input regions with different learners
 - ✓ A “soft” switching between learners



Mixture of experts

Gating networks decide which expert to use



Learning mixture of experts

- ➡ Learning consists of two tasks
 - ✓ Learn the parameters of individual experts
 - ✓ Learn the parameters of the gating network
 - Decides where to make a split
- ➡ **Assume:** gating functions give probabilities

$$0 \leq g_1(x), g_2(x), \dots, g_k(x) \leq 1, \quad \sum_u g_u(x) = 1$$

- ➡ Based on the probability we partition the space
 - ✓ Partitions belongs to different experts



Learning mixture of experts

- ➡ Gradient methods
- ➡ Expectation maximization (EM) algorithm



Summary



Ensembles, a summary

- ➡ Basic idea
 - ✓ Build different “experts”, and let them collaborate
- ➡ Advantages
 - ✓ Improve predictive performance
 - ✓ Other types of classifiers can be directly included
 - ✓ Easy to implement
 - ✓ No too much parameter tuning: Individual members don't need to be that accurate
- ➡ Disadvantages
 - ✓ The combined classifier is not so transparent (black box)
 - ✓ Not a compact representation
 - ✓ Boosting can overfit, specially in presence of noise








Some practical advice

- ▶ If the classifier is unstable (high variance), then apply bagging
- ▶ If the classifier is stable and simple (high bias) then apply boosting
- ▶ If the classifier is stable and complex then apply randomization injection
- ▶ If you have many classes and a binary classifier then try error-correcting codes. If it does not work then use a complex binary classifier



References I

-  Dietterich, T. G. (2000). 'An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization'. In: *Machine Learning* 40, pp. 139–157.
-  Dietterich, T. G. and G. Bakiri (1995). 'Solving multiclass learning problems via error-correcting output codes'. In: *Journal of Artificial Intelligence Research* 2, pp. 263–286.
-  Ho, T. K. (1998). 'The Random Subspace Method for Constructing Decision Forests'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8, pp. 832–844.
-  Rifkin, R. and A. Klautau (2004). 'In Defense of One-Vs-All Classification'. In: *Journal of Machine Learning Research* 5, pp. 101–141.
-  Rokach, L. (2009). 'Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography'. In: *Computational Statistics & Data Analysis* 53.12, pp. 4046–4072.

References II



Webb, G. I. (2000). 'MultiBoosting: A Technique for Combining Boosting and Wagging'. In: *Machine Learning* 40.2, pp. 159–196.