Ensemble Learning

Note: Unless otherwise noted all references including images are from the required textbook, Machine Learning: A Probabilistic Perspective by Kevin P. Murphy.

Ensemble Learning

Ensemble learning refers to learning a weighted combination of base models of the form

$$f(y|\mathbf{x}, \boldsymbol{\pi}) = \sum_{m \in \mathcal{M}} w_m f_m(y|\mathbf{x})$$

where the w_m are tunable parameters. Ensemble learning is sometimes called a **committee method**, since each base model f_m gets a weighted "vote."

Ensemble Learning

Clearly ensemble learning is closely related to learning adaptive-basis function models. In fact, one can argue that a neural net is an ensemble method, where f_m represents the mth hidden unit, and w_m are the output layer weights. Also, we can think of boosting as kind of ensemble learning, where the weights on the base models are determined sequentially.

Stacking

An obvious way to estimate the weights is to use

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} w_m f_m(\mathbf{x}))$$

However, this will result in overfitting, with w_m being large for the most complex model.

Stacking

A simple solution to this is to use cross-validation. In particular, we can use the LOOCV estimate

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(\mathbf{x}))$$

where $\hat{f}_m^{-i}(\mathbf{x})$ is the predictor obtained by training on data excluding $(\mathbf{x}_i, y_i)$. This is known as stacking, (Wolpert 1992).

Error-correcting Output Codes

An interesting form of ensemble learning is known as error-correcting output codes or ECOC (Dietterich and Bakiri 1995), which can be used in the context of multi-class classification. The idea is that we are trying to decode a symbol (namely the class label) which has C possible states.

Error-correcting Output Codes

We could use a bit vector of length $B = \lceil \log_2 C \rceil$ to encode the class label, and train B separate binary classifiers to predict each bit.

However, by using more bits, and by designing the codewords to have maximal Hamming distance from each other, we get a method that is more resistant to individual bit-flipping errors (misclassification).

Error-correcting Output Codes

Class	C_1	C_2	C_3	C_4	C_5	C_6	$\cdots$	C_{15}
0	1	1	0	0	0	0	$\cdots$	1
1	0	0	1	1	1	1	$\cdots$	0
					$\vdots$			
9	0	1	1	1	0	0	$\cdots$	0

Part of a 15-bit error-correcting output code for a 10-class problem.
Each row defines a two-class problem.

Error-correcting Output Codes

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

Part of a 15-bit error-correcting output code for a 10-class problem.
Each row defines a two-class problem.

Error-correcting Output Codes

In the example above, we use B = 15 bits to encode a C = 10 class problem. The minimum Hamming distance between any pair of rows is 7. The decoding rule is

$$\hat{c}(\mathbf{x}) = \min_{c} \sum_{b=1}^{B} |C_{cb} - \hat{p}_b(\mathbf{x})|$$

where C_{cb} is the b'th bit of the codeword for class c.