بازشناسی الگو

درس ۱۰

ترکیب طبقه‌بندی‌کننده‌ها

Combining Classifiers

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

http://courses.fouladi.ir/pr

ترکیب طبقه‌بندی‌کننده‌ها

۱

مقدمه

❖ Combining Classifiers

The basic philosophy behind the combination of different classifiers lies in the fact that even the "best" classifier fails in some patterns that other classifiers may classify correctly. Combining classifiers aims at exploiting this complementary information residing in the various classifiers.

Thus, one designs different optimal classifiers and then combines the results with a specific rule.

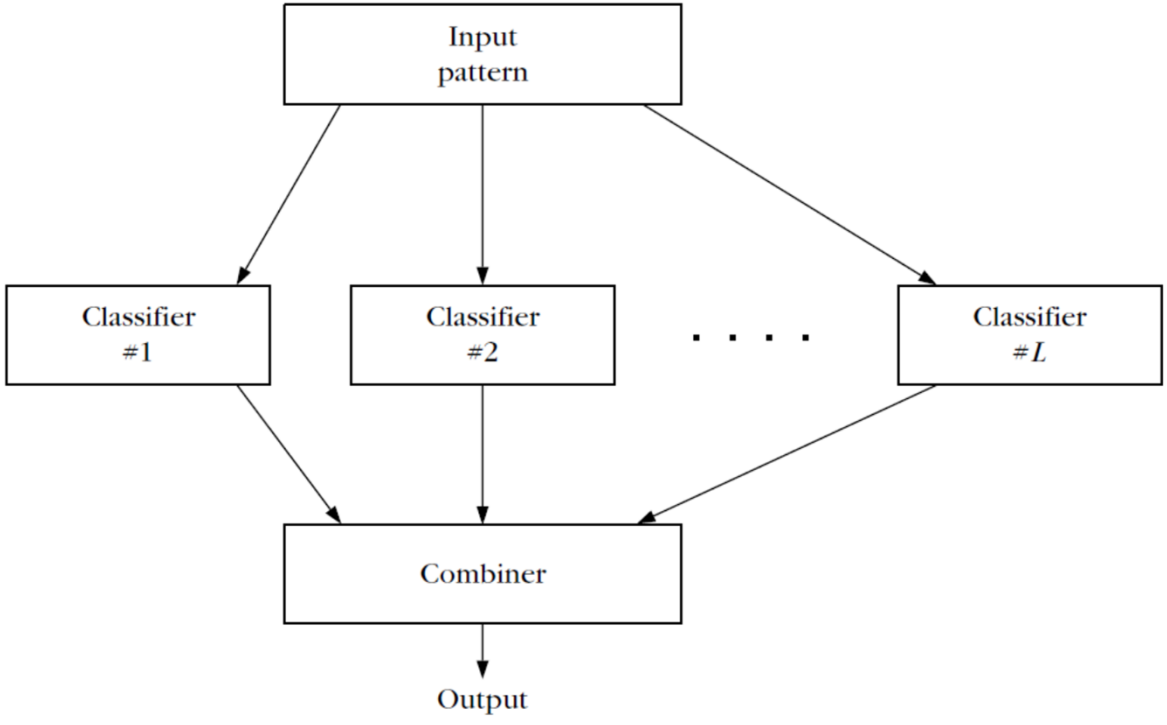


FIGURE 4.29

L classifiers are combined in order to provide the final decision for an input pattern. The individual classifiers may operate in the same or in different feature spaces.

- Assume that each of the, say, L designed classifiers provides at its output the posterior probabilities:

$$P(\omega_i \mid \underline{x}), i = 1,\ 2,\ ...,\ M$$

- Product Rule: Assign $\underline{x}$ to the class ω_i:

$$i = \arg\max_k \prod_{j=1}^{L} P_j(\omega_k \mid \underline{x})$$

where $P_j(\omega_k \mid \underline{x})$ is the respective posterior probability of the j^{th} classifier.

- Sum Rule: Assign $\underline{x}$ to the class ω_i:

$$i = \arg\max_k \sum_{j=1}^{L} P_j(\omega_k \mid \underline{x})$$

- Majority Voting Rule: Assign $\underline{x}$ to the class for which there is a consensus or when at least ℓ_c of the classifiers agree on the class label of $\underline{x}$ where:

$$\ell_c = \begin{cases} \frac{L}{2}+1, & L \text{ even} \\ \frac{L+1}{2}, & L \text{ odd} \end{cases}$$

otherwise the decision is rejection, that is no decision is taken.

Thus, correct decision is made if the majority of the classifiers agree on the correct label, and wrong decision if the majority agrees in the wrong label.

➢ **Dependent** or **non-Dependent** classifiers?

- Although there are not general theoretical results, experimental evidence has shown that the more independent in their decision the classifiers are, the higher the expectation should be for obtaining improved results after combination.

- **However, there is no guarantee that combining classifiers results in better performance compared to the "best" one among the classifiers.**

➢ Towards Independence: A number of Scenarios.

- Train the individual classifiers using different training data points.

  To this end, choose among a number of possibilities:

  – Bootstrapping: This is a popular technique to combine unstable classifiers such as decision trees.
  (Bagging belongs to this category of combination.)

  – Stacking: Train the **combiner** with data points that have been excluded from the set used to train the individual classifiers.

  – Use different subspaces to train individual classifiers:
  According to the method, each individual classifier operates in a different feature subspace. That is, use different features for each classifier.

➢ Remarks:

- The **majority voting** and the **summation** schemes rank among the most popular combination schemes.

- Training individual classifiers in different subspaces seems to lead to substantially **better improvements** compared to classifiers operating in the same subspace.

- Besides the above three rules (product, sum, majority), other alternatives are also possible, such as to use the median value of the outputs of individual classifiers.

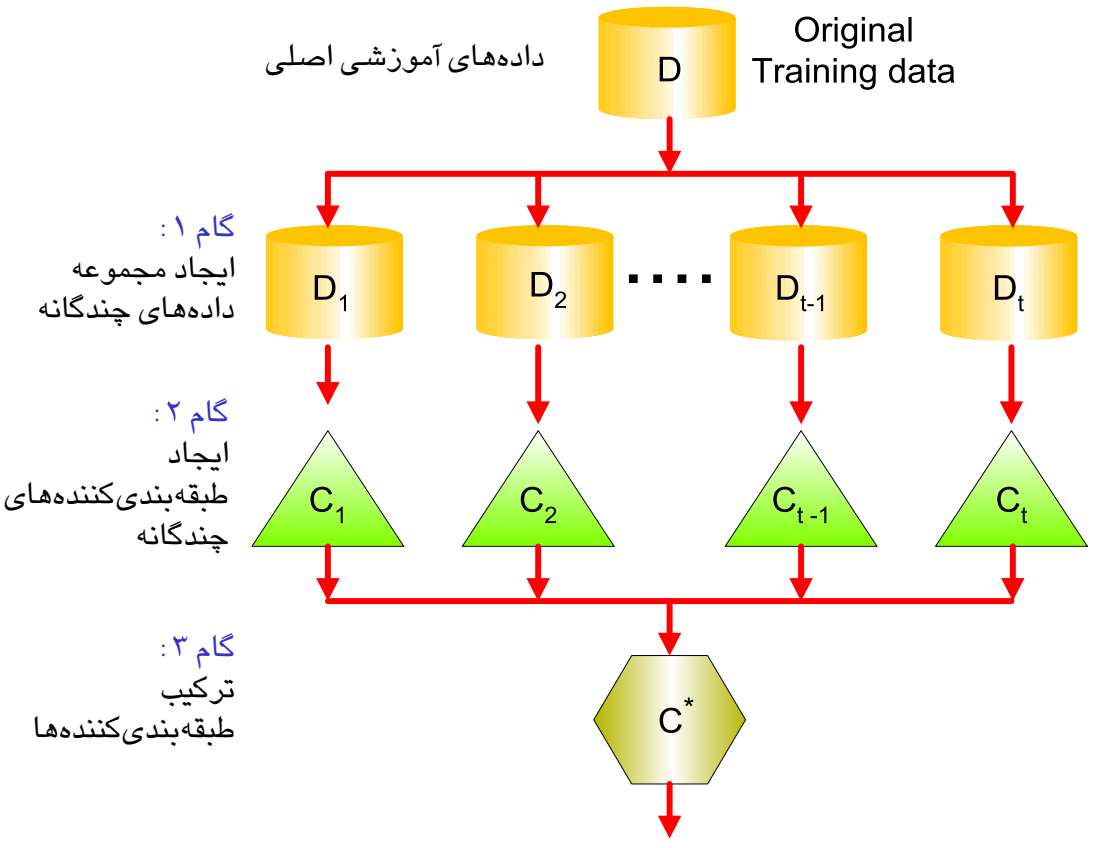
ترکیب طبقه‌بندی‌کننده‌ها

۲

یادگیری دسته‌جمعی

یادگیری دسته‌جمعی

ENSEMBLE LEARNING



ترکیب طبقه‌بندی‌کننده‌ها

قواعد ترکیب ثابت

FIXED COMBINATION RULES

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

مثال:

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.3
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.024

ترکیب طبقه‌بندی‌کننده‌ها

چرا کار می‌کند؟ (مثال)

WHY DOES IT WORK?

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

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

ترکیب طبقه‌بندی‌کننده‌ها

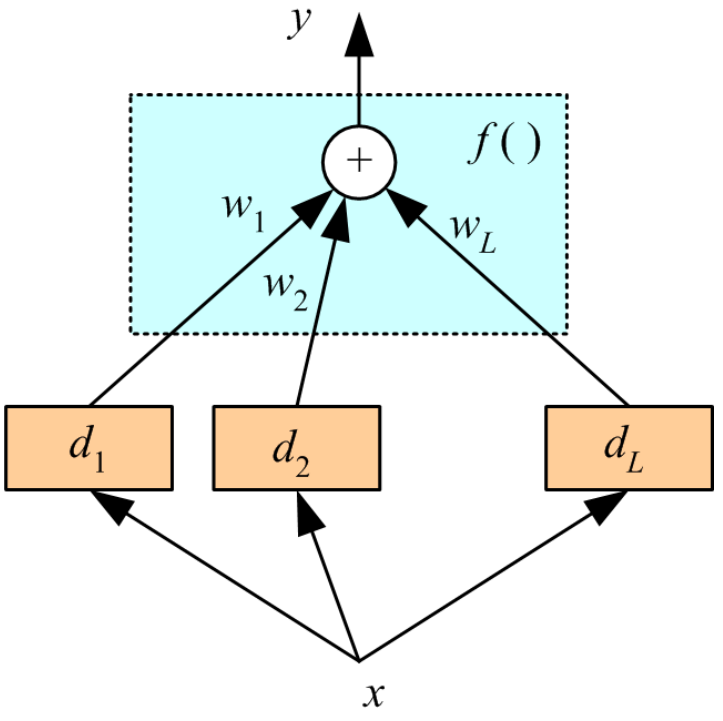
۳

روی‌کردهای ساده برای ترکیب طبقه‌بندی کننده‌ها

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد رأی‌دهی

VOTING



ترکیب خطی:

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$

طبقه‌بندی:

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

ترکیب طبقه‌بندی‌کننده‌ها

چرا کار می‌کند؟ (توضیح با روی‌کرد بیزی)

WHY DOES IT WORK?

از دیدگاه بیزی داریم:

$$P(C_i \mid x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i \mid x, \mathcal{M}_j) P(\mathcal{M}_j)$$

اگر d_jها مستقل باشند، داریم:

$$\text{Var}(y) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \cdot \text{Var}(d_j) = \frac{1}{L} \text{Var}(d_j)$$

بایاس تغییر نمی‌کند،
اما واریانس با ضریب L کاهش می‌یابد.

اگر d_jها وابسته باشند، خطا با همبستگی مثبت افزایش می‌یابد:

$$\text{Var}(y) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \text{Var}(d_j) + 2\sum_j \sum_{i<j} Cov(d_i, d_j)\right]$$

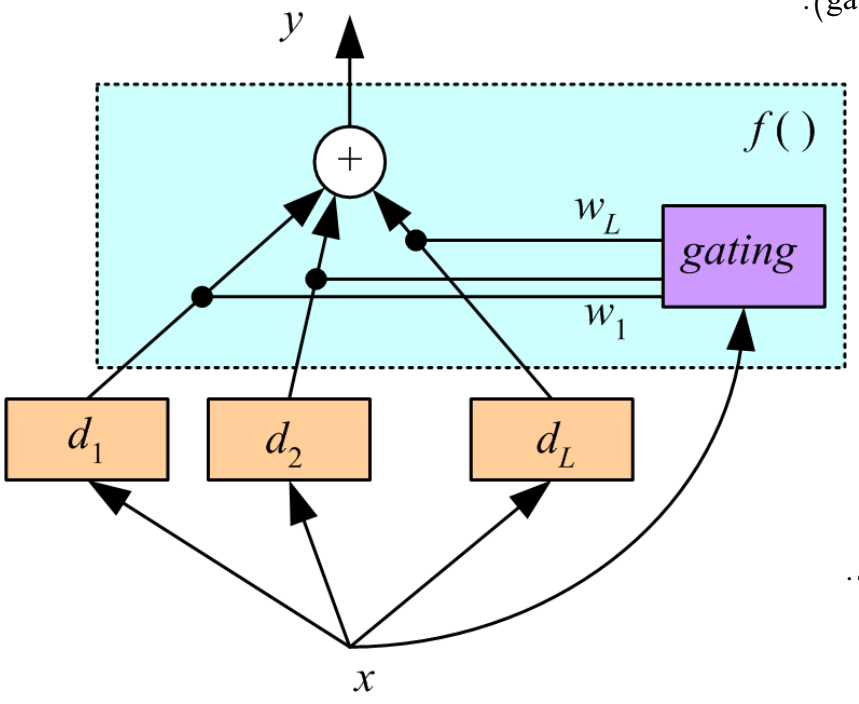
ترکیب طبقه‌بندی‌کننده‌ها

مخلوط خبره‌ها

MIXTURE OF EXPERTS

شیوه‌ای از رأی‌دهی
که وزن‌ها وابسته به ورودی هستند (gating).

$$y = \sum_{j=1}^{L} w_j d_j$$



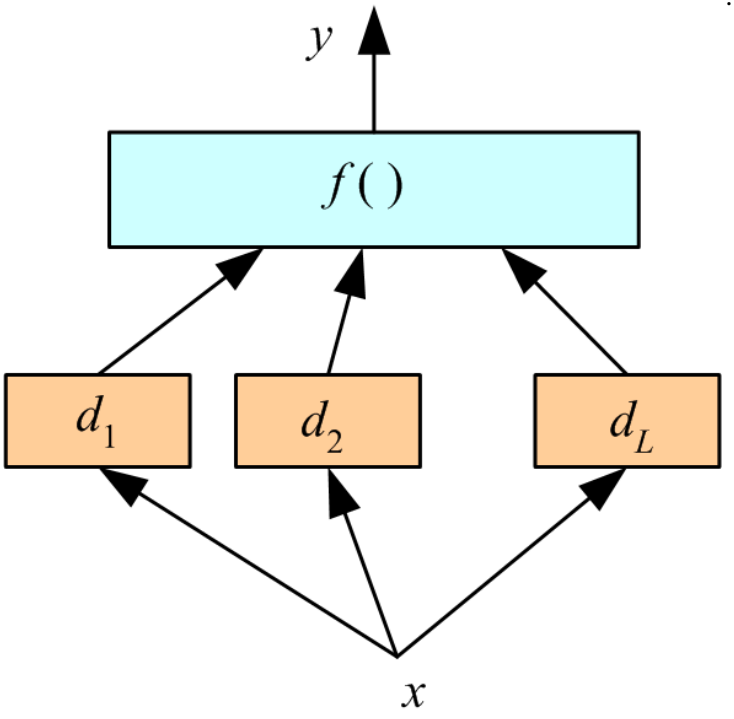
خبره‌ها یا gating می‌تواند غیرخطی باشد.

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «استکینگ» (پشته‌گذاری)

STACKING APPROACH

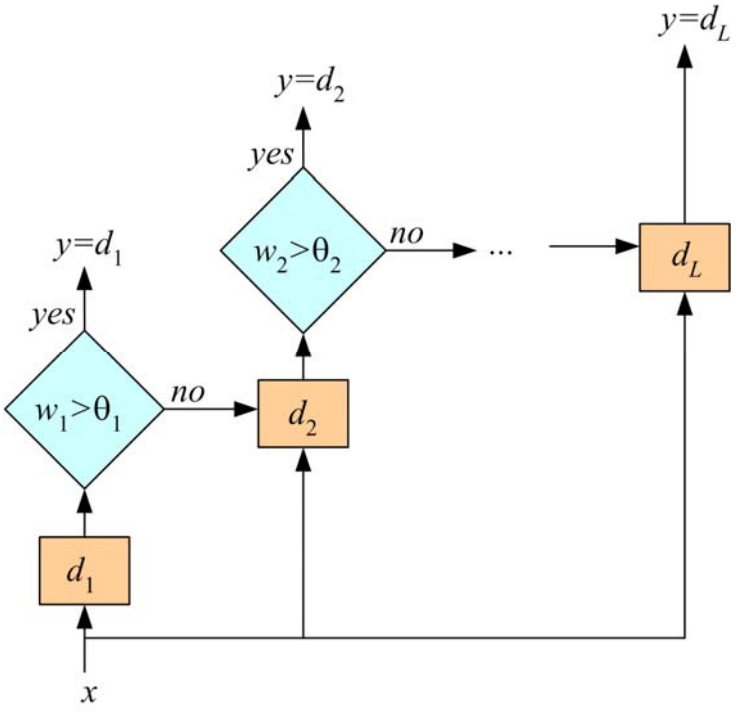
تابع ترکیب‌گر f() خودش یک یادگیرنده‌ی دیگر است.



ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «کسکیدینگ» (آبشارگذاری)

CASCADİNG APPROACH



از d_j تنها در صورتی استفاده می‌شود که قبلی d_{j-1} مطمئن نباشد.

یادگیرنده‌ها، به‌ترتیب افزایش پیچیدگی در آبشار کار گذاشته می‌شوند.

ترکیب طبقه‌بندی‌کننده‌ها

کدهای تصحیح‌کننده‌ی خروجی (۱ از ۲)

ERROR-CORRECTİNG OUTPUT CODES

- K classes; L problems
- Code matrix W codes classes in terms of learners

- **One per class**

  $L = K$

$$\mathrm{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

- **Pairwise**

  $L = K\,(K-1)\,/\,2$

$$\mathrm{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

ترکیب طبقه‌بندی‌کننده‌ها

کدهای تصحیح‌کننده‌ی خروجی (۲ از ۲)

ERROR-CORRECTİNG OUTPUT CODES

- **Full code**
  $L = 2^{(K-1)} - 1$

$$W = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{bmatrix}$$

- With reasonable L, find W such that the Hamming distance between rows and columns are maximized.
- Voting scheme

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

- Subproblems may be more difficult than one-per-K

ترکیب طبقه‌بندی‌کننده‌ها

۴

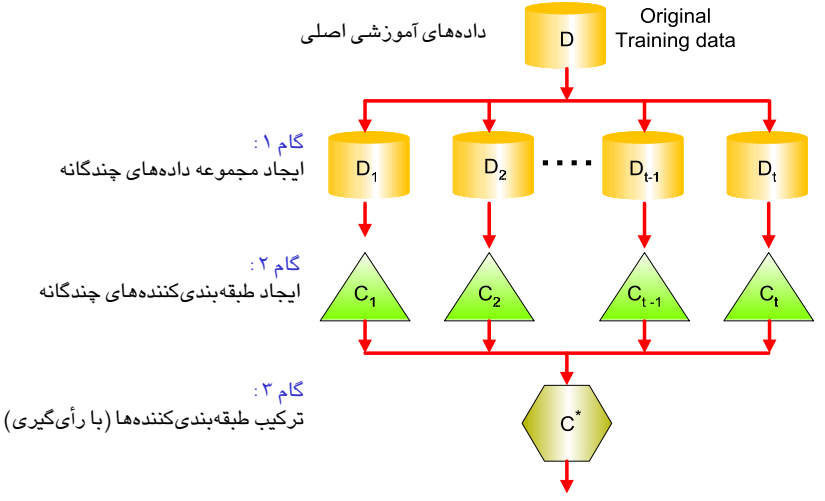
روی‌کرد «بگینگ» برای ترکیب طبقه‌بندی کننده‌ها

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بگینگ»

BAGGING

از خودراه‌اندازی (bootstrapping) برای تولید L مجموعه‌ی آموزشی استفاده می‌کنیم
و با هر یک، یک طبقه‌بندی‌کننده‌ی پایه (base classifier) را آموزش می‌دهیم.
برای ترکیب طبقه‌بندی‌کننده‌ها از رأی‌گیری (Voting) استفاده می‌کنیم.
(میانگین‌گیری یا میانه‌گیری با رگرسیون)



الگوریتم‌های ناپایدار از بگینگ سود می‌برند.

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بگینگ»: مثال

BAGGING

از نمونه‌برداری با جایگذاری استفاده می‌کنیم:

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

بر روی هر نمونه‌ی bootstrap، یک طبقه‌بندی‌کننده می‌سازیم.

احتمال انتخاب هر نمونه عبارت است از:

$$\left(1-\frac{1}{n}\right)^{n}$$

ترکیب طبقه‌بندی‌کننده‌ها

۵

روی‌کرد «بوستینگ» برای ترکیب طبقه‌بندی کننده‌ها

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بوستینگ»

BOOSTING

«بوستینگ» یک روال تکراری برای تغییر وفقی توزیع داده‌های آموزشی
با تمرکز بیشتر بر روی داده های طبقه‌بندی شده به‌صورت نادرست است.

در ابتدا، به همه‌ی N داده، وزن‌های مساوی نسبت داده می‌شود.
بر خلاف «بگینگ»، وزن‌ها می‌توانند در انتهای یک دور «بوستینگ» تغییر کنند.

داده‌هایی که نادرست طبقه‌بندی شده‌اند، وزن‌شان افزایش خواهد یافت.
داده‌هایی که درست طبقه‌بندی شده‌اند، وزن‌شان کاهش خواهد یافت.

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بوستینگ»: مثال

BOOSTING

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

مثال 4، به‌سختی طبقه‌بندی می‌شود،
پس وزن آن افزایش می‌یابد،
بنابراین، احتمال انتخاب آن در دورهای بعدی بیشتر می‌شود.

ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بوستینگ»: الگوریتم آدابوست

ADABOOST ALGORITHM

D_1= initial dataset with equal weights

FOR i = 1 **to** k **DO**

Learn new classifier C_i;

Compute α_i (classifier's importance);

Update example weights;

Create new training set D_{i+1} (using weighted sampling)

END FOR

Construct Ensemble which uses C_i weighted by α_i (i = 1, k)

ترکیب طبقه‌بندی‌کننده‌ها

رویکرد «بوستینگ»: الگوریتم آدابوست: مثال (۱ از ۲)

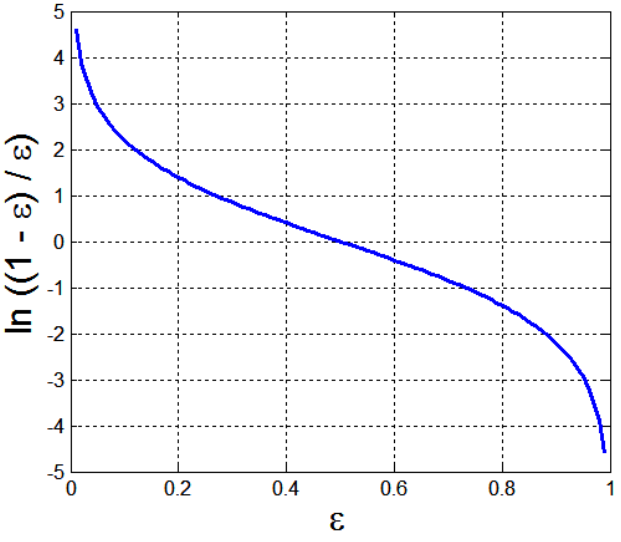
ADABOOST ALGORITHM

- Base classifiers: $C_1, C_2, \ldots, C_T$
- Error rate (weights add up to 1):

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بوستینگ»: الگوریتم آدابوست: مثال (۲ از ۲)

ADABOOST ALGORITHM

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} e^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ e^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

افزایش وزن در صورت نادرست‌بودن طبقه‌بندی
(افزایش متناسب با اهمیت طبقه‌بندی‌کننده‌ها صورت می‌گیرد.)

اگر هر یک از دورهای میانی، نرخ خطایی بالاتر از ۵۰% تولید کنند،
وزن‌ها به $1/n$ برمی‌گردند و روال نمونه‌برداری مجدد، تکرار می‌شود.

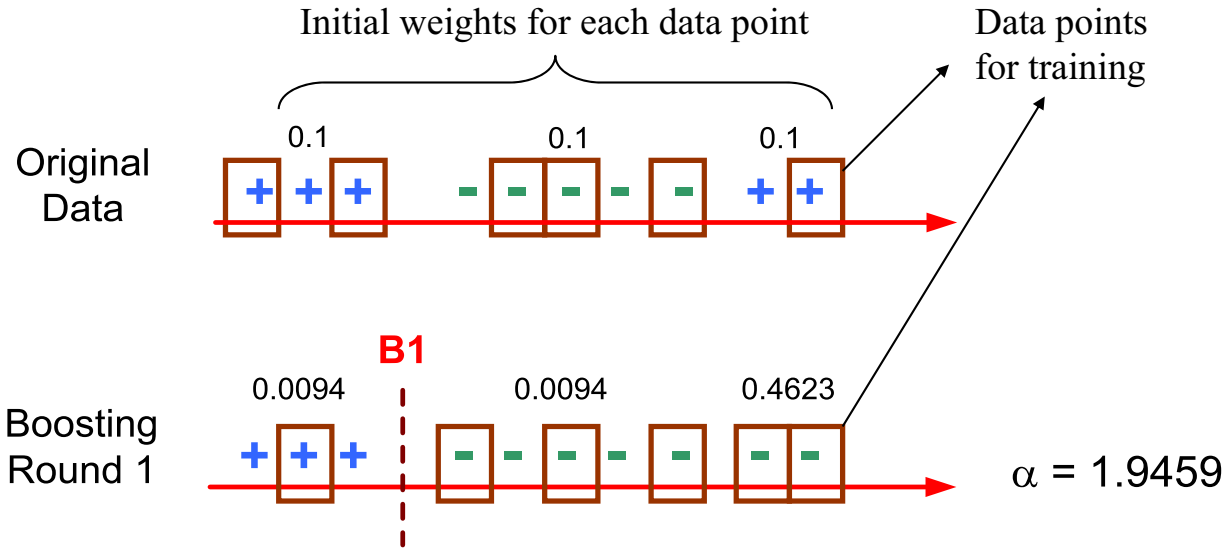
طبقه‌بندی (α_j اهمیت طبقه‌بندی‌کننده‌ی j برای کل مجموعه داده است):

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

ترکیب طبقه‌بندی‌کننده‌ها

رویکرد «بوستینگ»: الگوریتم آدابوست: مثال مصور (۱ از ۲)

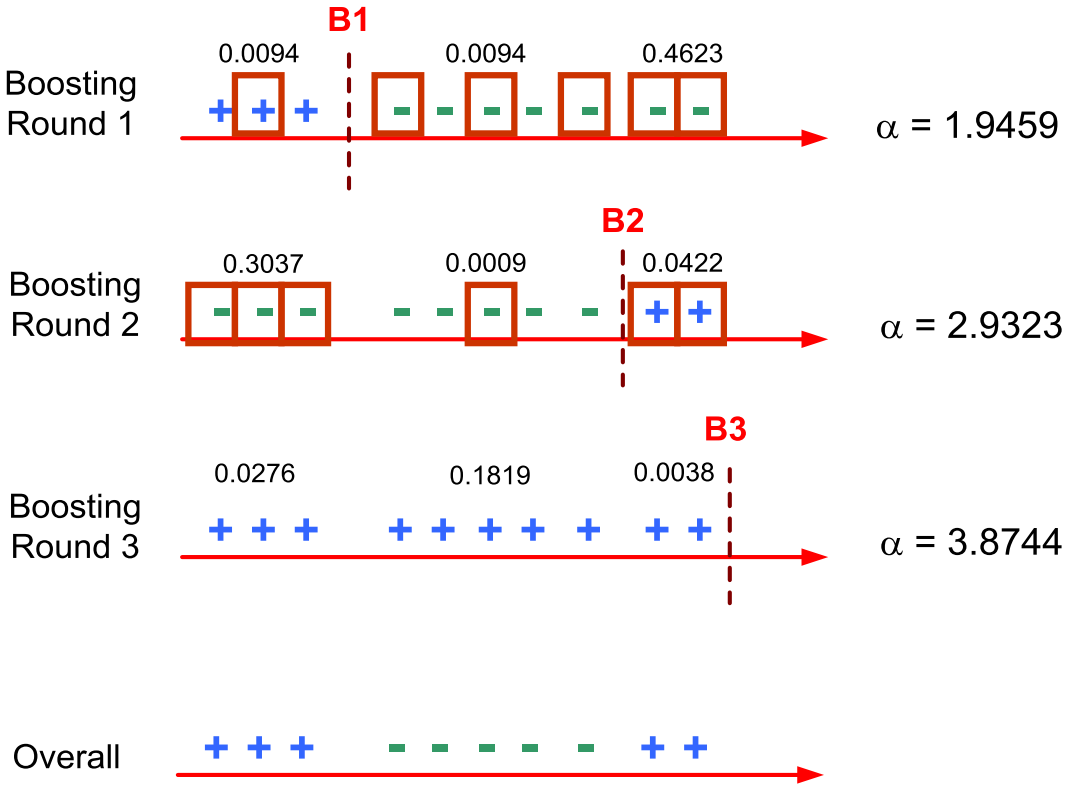
ADABOOST ALGORITHM



ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بوستینگ»: الگوریتم آدابوست: مثال مصور (۲ از ۲)

ADABOOST ALGORITHM



ترکیب طبقه‌بندی‌کننده‌ها

روی‌کرد «بوستینگ»: الگوریتم آدابوست: شبه‌کد

ADABOOST ALGORITHM

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$$Z_j \leftarrow \sum_t p_{j+1}^t; \quad p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^L \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

❖ The Boosting Approach

- **The origins:** Is it possible a weak learning algorithm (one that performs slightly better than a random guessing) to be boosted into a strong algorithm? (Villiant 1984).

- The procedure to achieve it:

  - Adopt a weak classifier known as the base classifier.

  - Employing the base classifier, design a series of classifiers, in a hierarchical fashion, each time employing a different weighting of the training samples. Emphasis in the weighting is given on the hardest samples, i.e., the ones that keep "failing".

  - Combine the hierarchically designed classifiers by a weighted average procedure.

➢ The AdaBoost Algorithm.

Construct an optimally designed classifier of the form:

$$f(\underline{x}) = \text{sign}\{F(\underline{x})\}$$

where:

$$F(\underline{x}) = \sum_{k=1}^{K} a_k \varphi(\underline{x}; \underline{\vartheta}_k)$$

where $\varphi(\underline{x}; \underline{\vartheta}_k)$ denotes the base classifier that returns a binary class label:

$$\varphi(\underline{x}; \underline{\vartheta}_k) \in \{-1, 1\}$$

$\underline{\vartheta}$ is a parameter vector.

- The essence of the method.
  Design the series of classifiers:

$$\varphi(\underline{x};\underline{\vartheta}_1), \varphi(\underline{x};\underline{\vartheta}_2), ..., \varphi(\underline{x};\underline{\vartheta}_k)$$

  The parameter vectors

$$\underline{\vartheta}_k, k = 1,\ 2,\ ...,\ K$$

  are optimally computed so as:
  - To minimize the error rate on the training set.
  - Each time, the training samples are re-weighted so that the weight of each sample depends on its history. Hard samples that "insist" on failing to be predicted correctly, by the previously designed classifiers, are more heavily weighted.

- Updating the weights for each sample $\underline{x}_i, i = 1, 2, ..., N$

$$w_i^{(m+1)} = \frac{w_i^m \exp\left(-y_i a_m \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right)\right)}{Z_m}$$

  - Z_m is a normalizing factor common for all samples.
  - $a_m = \frac{1}{2} \ln \frac{1 - P_m}{P_m}$

    where $P_m < 0.5$ (by assumption) is the error rate of the optimal classifier $\varphi\left(\underline{x}; \underline{\vartheta}_m\right)$ at stage m. Thus $\alpha_m > 0$.

  - The term: $\exp\left(-y_i a_m \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right)\right)$

    takes a large value if $y_i \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right) < 0$ (wrong classification) and a small value in the case of correct classification

    $$\{y_i \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right) > 0\}$$

  - The update equation is of a multiplicative nature. That is, successive large values of weights (hard samples) result in larger weight for the next iteration

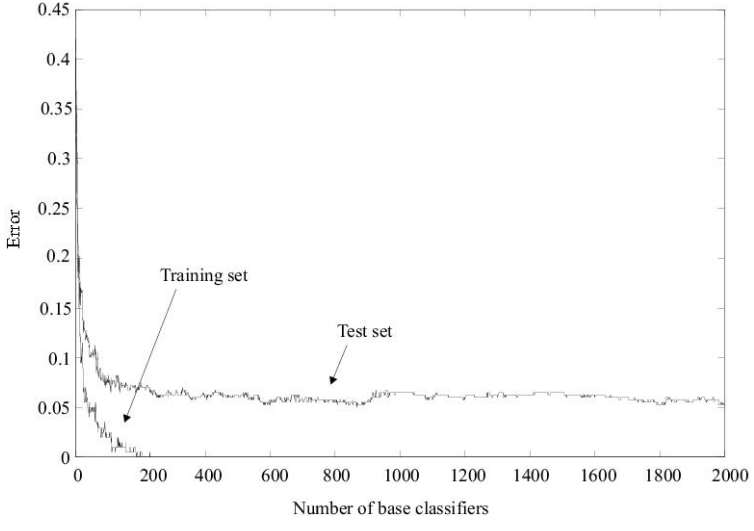
The AdaBoost Algorithm

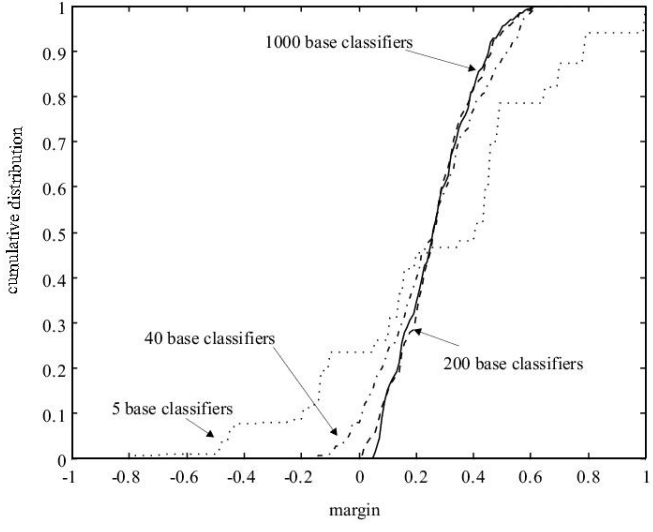
- Initialize: $w_i^{(1)} = \frac{1}{N},\ i = 1, 2 \ldots, N$
- Initialize: $m = 1$
- Repeat
  - Compute optimum $\boldsymbol{\theta}_m$ in $\phi(\cdot; \boldsymbol{\theta}_m)$ by minimizing P_m; (4.135)
  - Compute the optimum P_m; (4.135)
  - $\alpha_m = \frac{1}{2} \ln \frac{1-P_m}{P_m}$
  - $Z_m = 0.0$
  - For $i = 1$ to N
    - $w_i^{(m+1)} = w_i^{(m)} \exp\left(-y_i \alpha_m \phi(\boldsymbol{x}_i; \boldsymbol{\theta}_m)\right)$
    - $Z_m = Z_m + w_i^{(m+1)}$
  - End{For}
  - For $i = 1$ to N
    - $w_i^{(m+1)} = w_i^{(m+1)} / Z_m$
  - End {For}
  - $K = m$
  - $m = m + 1$
- Until a termination criterion is met.
- $f(\cdot) = \text{sign}(\sum_{k=1}^{K} \alpha_k \phi(\cdot, \boldsymbol{\theta}_k))$

➢ Remarks:

- Training error rate tends to zero after a few iterations. The test error levels to some value.

- AdaBoost is greedy in reducing the margin that samples leave from the decision surface.



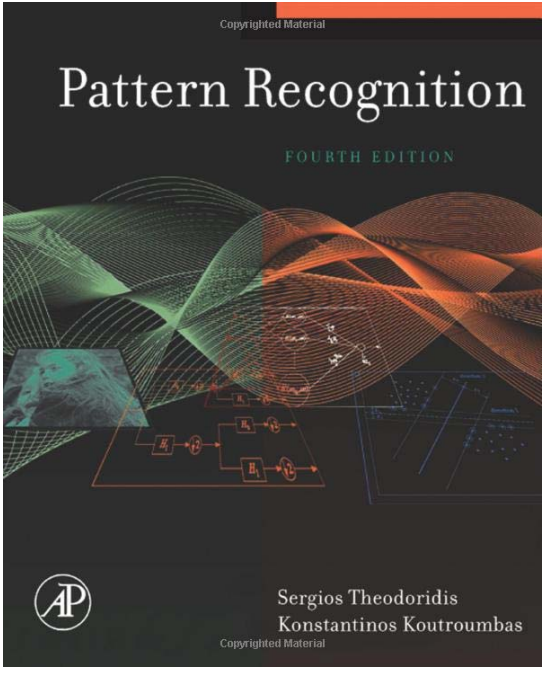


ترکیب طبقه‌بندی‌کننده‌ها

۶

منابع

منبع اصلی



S. Theodoridis, K. Koutroumbas,
Pattern Recognition,
Fourth Edition, Academic Press, 2009.

Chapter 4 (4-21 .. 4-23)

CHAPTER

Nonlinear Classifiers 4

4.1 INTRODUCTION

In the previous chapter we dealt with the design of linear classifiers described by linear discriminant functions (hyperplanes) $g(\boldsymbol{x})$. In the simple two-class case, we saw that the perceptron algorithm computes the weights of the linear function $g(\boldsymbol{x})$, provided that the classes are linearly separable. For nonlinearly separable classes, linear classifiers were optimally designed, for example, by minimizing the squared error. In this chapter we will deal with problems that are not linearly separable and for which the design of a linear classifier, even in an optimal way, does not lead to satisfactory performance. The design of nonlinear classifiers emerges now as an inescapable necessity.

4.2 THE XOR PROBLEM

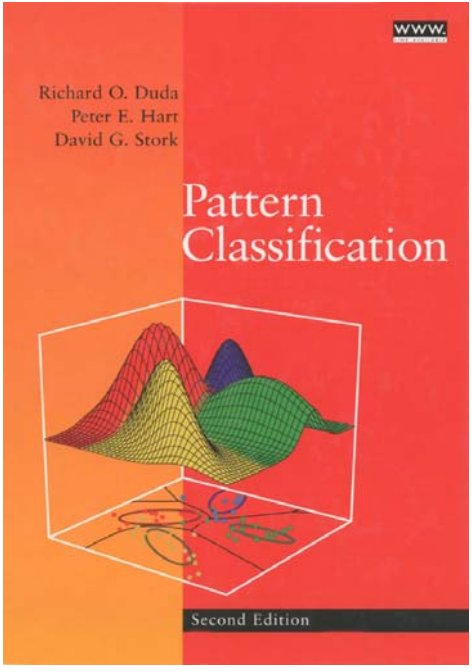
To seek nonlinearly separable problems one does not need to go into complicated situations. The well-known *Exclusive OR (XOR)* Boolean function is a typical example of such a problem. Boolean functions can be interpreted as classification tasks. Indeed, depending on the values of the input binary data $\boldsymbol{x} = [x_1, x_2, \ldots, x_l]^T$, the output is either 0 or 1, and $\boldsymbol{x}$ is classified into one of the two classes $A(1)$ or $B(0)$. The corresponding truth table for the XOR operation is shown in Table 4.1.

Figure 4.1 shows the position of the classes in space. It is apparent from this figure that no single straight line exists that separates the two classes. In contrast, the other two Boolean functions, AND and OR, are linearly separable. The corresponding truth tables for the AND and OR operations are given in Table 4.2 and the respective class positions in the two-dimensional space are shown in Figure 4.2a and 4.2b. Figure 4.3 shows a perceptron, introduced in the previous chapter, with synaptic weights computed so as to realize an OR gate (verify).

Our major concern now is first to tackle the XOR problem and then to extend the procedure to more general cases of nonlinearly separable classes. Our kickoff point will be geometry.

151

منبع کمکی



R.O. Duda, P.E. Hart, and D.G. Stork,
Pattern Classification,
Second Edition, John Wiley & Sons, Inc., 2001.

Chapter 9 (9.7)

CHAPTER
9

ALGORITHM-INDEPENDENT MACHINE LEARNING

9.1 INTRODUCTION

In the previous chapters we have seen many learning algorithms and techniques for pattern recognition. When confronting such a range of algorithms, everyone has wondered at one time or another which one is "best." Of course, some algorithms may be preferred because of their lower computational complexity; others may be preferred because they take into account some prior knowledge of the form of the data (e.g., discrete, continuous, unordered list, string, ...). Nevertheless, there are classification problems for which such issues are of little or no concern, or we wish to compare algorithms that are equivalent in regard to them. In these cases we are left with the question, Are there any reasons to favor one algorithm over another? For instance, given two classifiers that perform equally well on the training set, it is frequently asserted that the *simpler* classifier can be expected to perform better on a test set. But is this version of *Occam's razor* really so evident? Likewise, we frequently prefer or impose *smoothness* on a classifier's decision functions. Do simpler or "smoother" classifiers generalize better, and, if so, why? In this chapter we address these and related questions concerning the foundations and philosophical underpinnings of statistical pattern classification. Now that the reader has intuition and experience with individual algorithms, these issues in the theory of learning may be better understood.

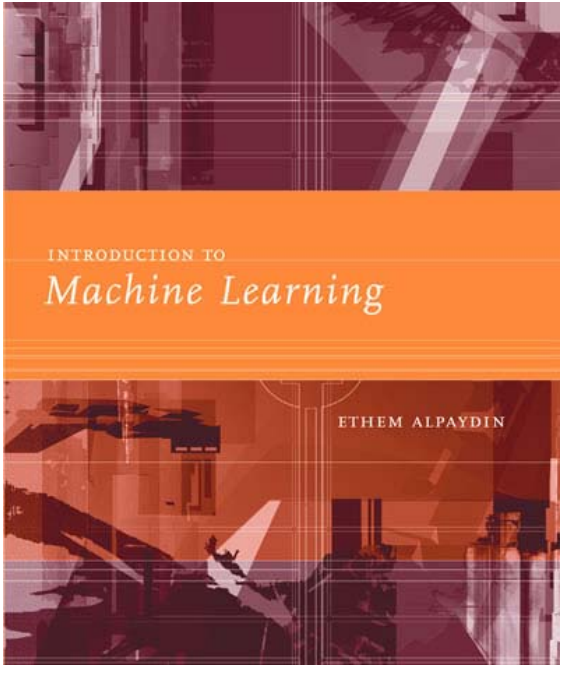
OCCAM'S RAZOR

In some fields there are strict conservation laws and constraint laws—such as the conservation of energy, charge, and momentum in physics, as well as the second law of thermodynamics, which states that the entropy of an isolated system can never decrease. These hold regardless of the number and configuration of the forces at play. Given the usefulness of such laws, we naturally ask, Are there analogous results in pattern recognition, ones that do not depend upon the particular choice of classifier or learning method? Are there any fundamental results that hold regardless of the cleverness of the designer, the number and distribution of the patterns, and the nature of the classification task?

Of course it is very valuable to know that there exists a constraint on classifier accuracy, the Bayes error rate, and it is sometimes useful to compare performance

453

منبع کمکی



Ethem Alpaydin,
Introduction to Machine Learning,
The MIT Press, 2004.

Chapter 15

15 *Combining Multiple Learners*

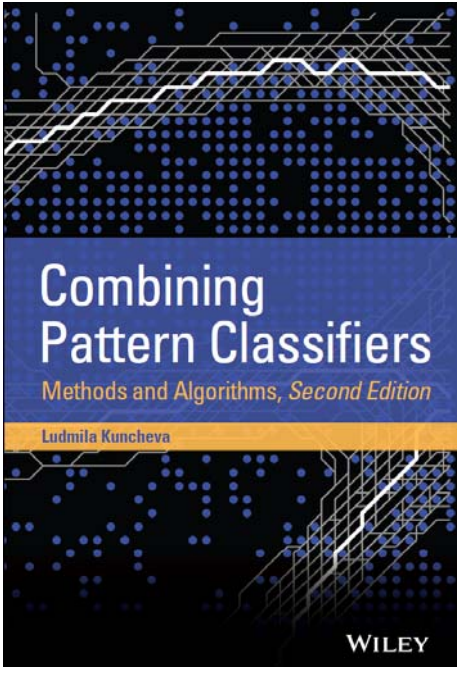
We discussed many different learning algorithms in the previous chapters. Though these are generally successful, no one single algorithm is always the most accurate. Now, we are going to discuss models composed of multiple learners that complement each other so that by combining them, we attain higher accuracy.

15.1 Rationale

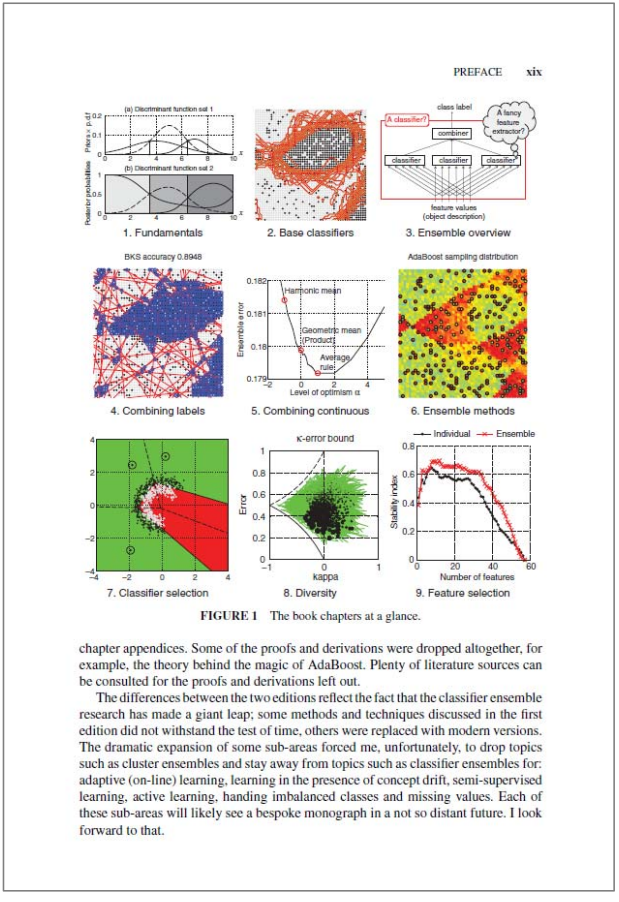
IN ANY APPLICATION, we can use one of several learning algorithms, and with certain algorithms, there are hyperparameters that affect the final learner. For example, in a classification setting, we can use a parametric classifier or a multilayer perceptron, and for example, with a multilayer perceptron, we should also decide on the number of hidden units. The No Free Lunch Theorem states that there is no single learning algorithm that in any domain always induces the most accurate learner. The usual approach is to try many and choose the one that performs the best on a separate validation set, as we discussed in chapter 14.

Each learning algorithm dictates a certain model that comes with a set of assumptions. This inductive bias leads to error if the assumptions do not hold for the data. Learning is an ill-posed problem and with finite data, each algorithm converges to a different solution and fails under different circumstances. The performance of a learner may be fine-tuned to get the highest possible accuracy on a validation set, but this fine-tuning is a complex task and still there are instances on which even the best learner is not accurate enough. The idea is that there may be another learner that is accurate on these. By suitably combining multiple learners then, accuracy can be improved. Recently with computation and mem-

منبع کمکی



Ludmila I. Kuncheva,
Combining Pattern Classifiers: Methods and Algorithms,
Second Edition, John Wiley & Sons, Inc., 2014.



PREFACE xix

1. Fundamentals
2. Base classifiers
3. Ensemble overview
4. Combining labels
5. Combining continuous
6. Ensemble methods
7. Classifier selection
8. Diversity
9. Feature selection

FIGURE 1 The book chapters at a glance.

chapter appendices. Some of the proofs and derivations were dropped altogether, for example, the theory behind the magic of AdaBoost. Plenty of literature sources can be consulted for the proofs and derivations left out.

The differences between the two editions reflect the fact that the classifier ensemble research has made a giant leap; some methods and techniques discussed in the first edition did not withstand the test of time, others were replaced with modern versions. The dramatic expansion of some sub-areas forced me, unfortunately, to drop topics such as cluster ensembles and stay away from topics such as classifier ensembles for: adaptive (on-line) learning, learning in the presence of concept drift, semi-supervised learning, active learning, handing imbalanced classes and missing values. Each of these sub-areas will likely see a bespoke monograph in a not so distant future. I look forward to that.