

機械学習 第10回 アンサンブル学習

立命館大学 情報理工学部

福森 隆寛

Beyond Borders

講義スケジュール

(第1～4回、第14回) (第5～13回、第15回)

□ 担当教員：村上 陽平先生・福森 隆寛

1	機械学習とは、機械学習の分類
2	機械学習の基本的な手順
3	識別（１）
4	識別（２）
5	識別（３）
6	回帰
7	サポートベクトルマシン
8	ニューラルネットワーク

9	深層学習
10	アンサンブル学習
11	モデル推定
12	パターンマイニング
13	系列データの識別
14	強化学習
15	半教師あり学習

□ 担当教員：叶 昕辰先生（第16回の講義を担当）

今回の講義内容

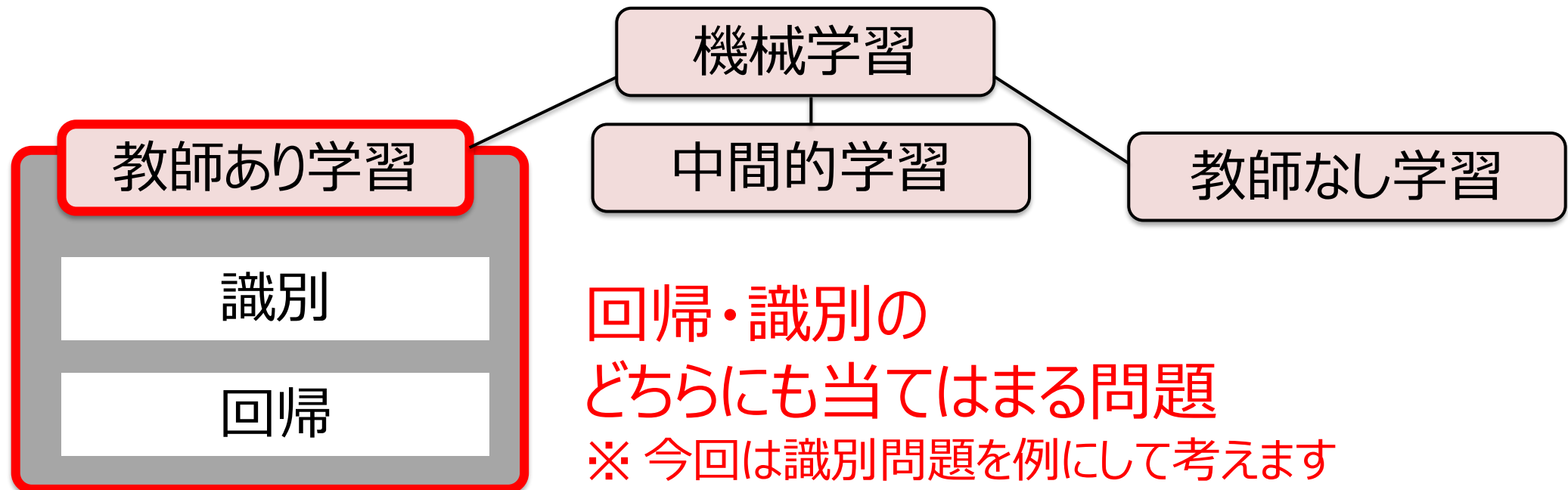
- 取り扱う問題の定義
- アンサンブル学習
- バギング
- ランダムフォレスト
- ブースティング
 - AdaBoost
 - 勾配ブースティング
- 演習問題

取り扱う問題の定義：教師あり問題

- 特徴ベクトルを入力して、それをクラス分けする識別器、または、それに対応する数値を出力する関数を作る

※ 教師あり学習の問題での学習データは、以下のペアで構成される

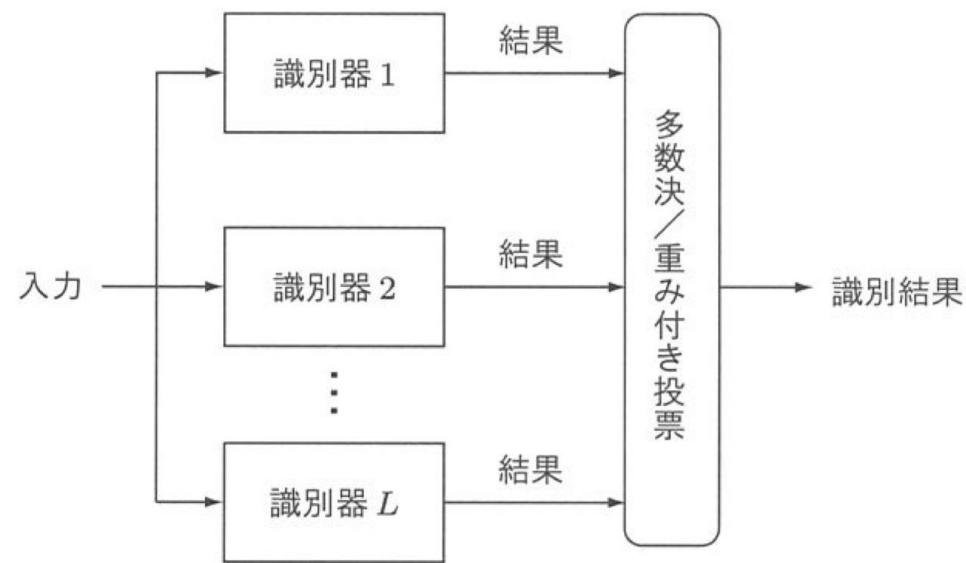
入力データの特徴ベクトル $\leftarrow \{\underline{x_i}, \underline{y_i}\}, \quad i = 1, 2, \dots, \underline{N} \longrightarrow$ 学習データの総数
正解情報



アンサンブル学習

□ アンサンブル学習

- 識別器を複数組み合わせ、それらの結果を統合することで個々の識別器よりも性能を向上させる方法
- なぜ性能が向上するのか？



アンサンブル学習の考え方

アンサンブル学習

□ 以下の識別器を考える

- 同じ学習データを用いて、異なる識別器を L 個作成
- 識別器の誤り率 ϵ は、全て等しく、その誤りは独立と仮定
 - 評価用データに対して、それぞれの識別器が誤る確率が独立（多くの識別器が一緒に誤るようなデータはない）ということ

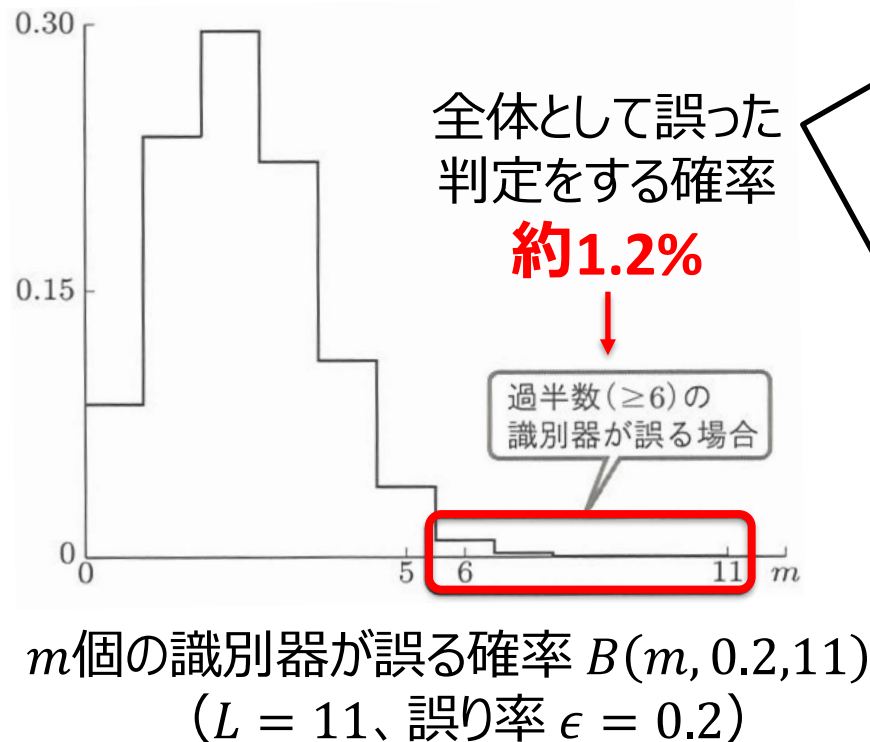


- この識別器の集合に、ある評価用データを与えたとき、 m 個の識別器が誤る確率は二項分布 $B(m, \epsilon, L)$ となる

$$B(m, \epsilon, L) = {}_L C_m \epsilon^m (1 - \epsilon)^{L-m}$$

アンサンブル学習

□ m 個の識別器が誤る確率



理論上は、**多数決による識別結果の選択で大幅に誤り率が減少**

- 個々の識別器の誤り率：**20%**
- 多数決方式の場合の誤り率：**1.2%**



現実^{うま}は、こんなには上手い^{うま}かない

識別器の出す誤りが「独立ではない」
(識別しやすい/識別しにくいデータがあるということ)

識別しにくいデータは、多くの識別器が誤った結果を出力するので、多数決による誤りの減少は期待できない

アンサンブル学習は「いかにして独立な識別器を作るか？」がポイント

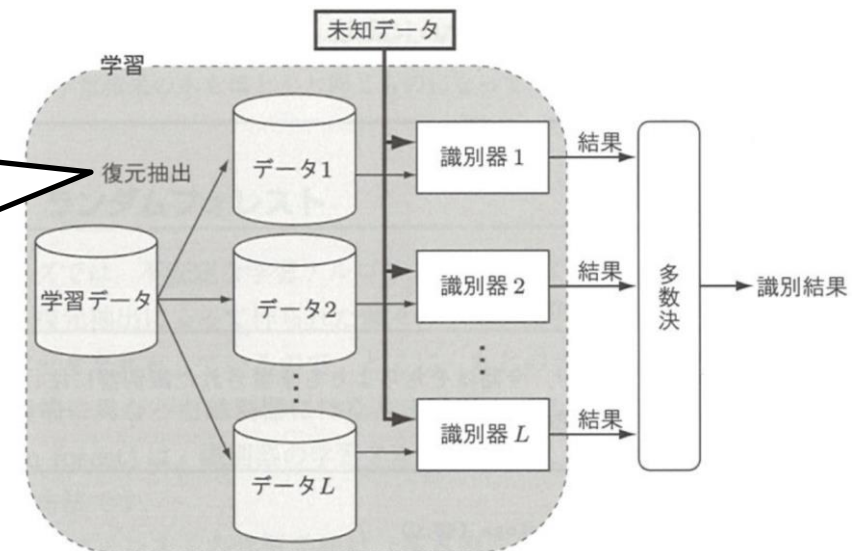
アンサンブル学習：バギング

□ バギング (bagging)

- 学習データから復元抽出^{ふくげん}することで、もとのデータと同じサイズの独立なデータ集合を複数作成し、各々のデータ集合に対して同じアルゴリズムで識別器を作成する方法
- 識別結果は、多数決によって決定する

復元抽出

取り出したデータを記録して、また元に戻す抽出法
同じデータが何回も取り出されることがある一方、
1回も取り出されないデータもある




バギング

アンサンブル学習：バギング

- 復元抽出によって作成されたデータ集合が、もとのデータ集合とどれくらい異なるのか？
- もとのデータ集合に含まれるあるデータが、復元抽出後のあるデータ集合に含まれない確率

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

N : データ集合の要素数



$N = 10$: 0.349
$N = 100$: 0.366
$N \rightarrow \infty$: 0.368 ($1/e$)

- おおよそ、どのような N に対しても、復元抽出後のデータ集合には、もとのデータ集合の約1/3のデータが含まれないことになる

アンサンブル学習：バギング

- 識別器を作成するアルゴリズムは不安定（学習データの違いに敏感）な方が良い
 - 少しデータが異なるだけで、異なった識別器になる
 - それぞれの識別器が、同じデータ量で学習しているので全ての識別器は、同程度に信用できると考えて、結果の統合は単純な多数決とする

演習問題10-1（10分間）

- バギングで作成する識別器は不安定な方が良い。
このアルゴリズムで用いる不安定な識別器は、
どのようなものがあるか考えよ。

アンサンブル学習：ランダムフォレスト

□ ランダムフォレスト (random forest)

- 複数の決定木を統合したアンサンブル学習手法
- 学習データの復元抽出で生成した複数のデータ集合に対して決定木を作成
 - この点は、バギングアルゴリズムと同様
- バギングアルゴリズム以上に異なる決定木が生成されるような様々な仕組みを導入
 - ランダムに選択した特徴の中から分割に用いる特徴を決定する
 - 意図的に過学習させる（木を大きく成長させる）

アンサンブル学習：ランダムフォレスト

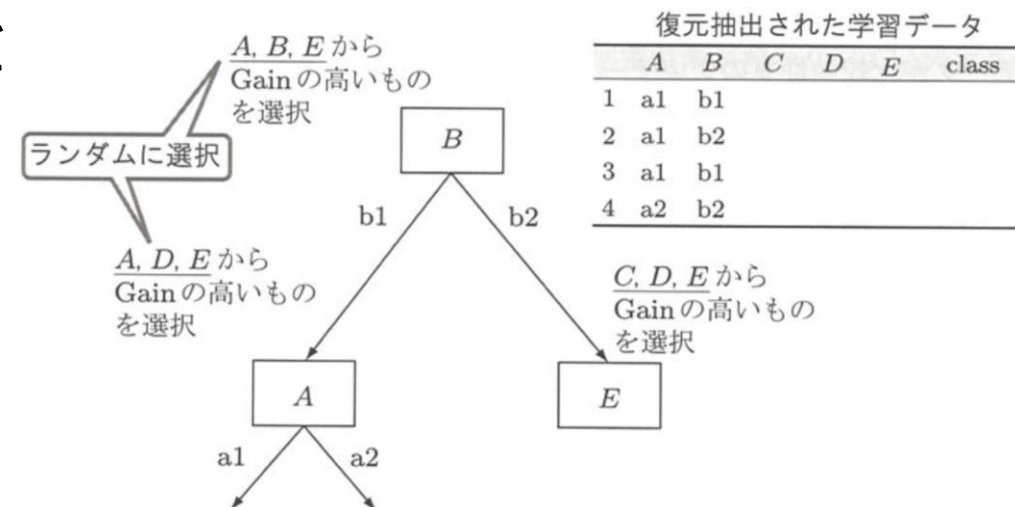
□ ランダムフォレストの学習方法

1. 学習データから復元抽出して、同サイズのデータ集合を複数作る
2. 各データ集合に対して、識別器として決定木を作成
 - 分岐特徴を選ぶとき、全特徴から予め決められた数だけ特徴をランダムに選出し、その中から最も分類能力の高い特徴（例：情報獲得量が高い特徴）を選ぶ
 - リーフが単一クラスの集合になるまで、これを再帰的に行う（過学習）
3. 全ての決定木の出力結果の多数決に基づいて最終的な識別結果を決定

アンサンブル学習：ランダムフォレスト

□ ランダムフォレストによる個々の木の作成の様子^{ようす}

- 学習データの特徴： $\{A, B, C, D, E\}$ の5種類
- ルートノードの分岐に用いる特徴
 - 予め決められた数（例えば、3種類）をランダムに選択
 - $\{A, B, E\}$ が選択されたとすると、それぞれの分類能力を計算し、最も分類能力の高い特徴を選んでデータを分割
- この特徴選択アルゴリズムを分割されたデータ集合にも適用させて木を成長させる

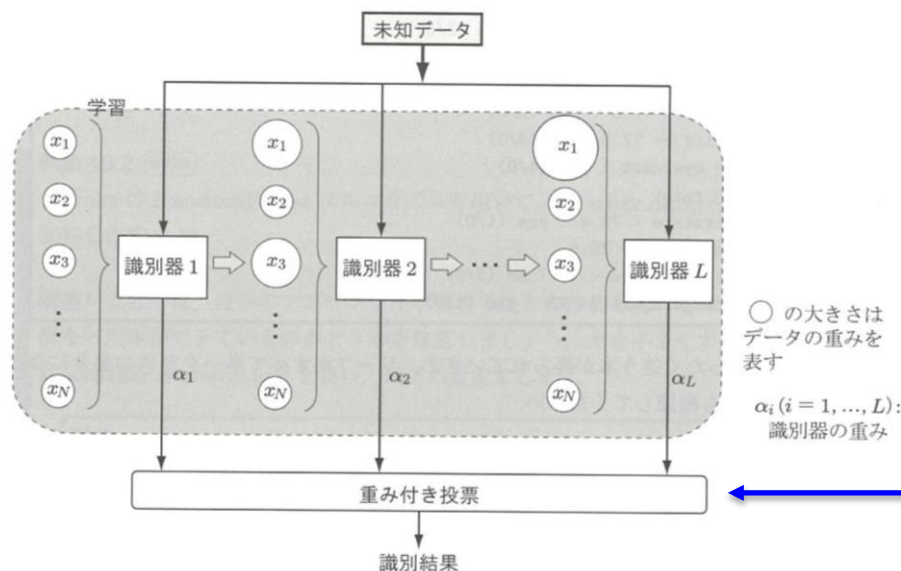


アンサンブル学習：ブースティング

□ ブースティング (boosting)

- 誤りを減らすことに特化した識別器を次々^{つぎつぎ}に加える方法で異なる振る舞いをする識別器の集合を作成
- 誤りを減らすことに特化した重みを与えたデータのもとでそれぞれの識別器を作成
- 各識別器の性能の重み付き投票^{とうひょう}から識別結果を求める

ブースティングの
考え方



ブースティングにおける識別器

もとの学習データをゆがめて作成
(識別器ごとに信頼性が異なる)

↓
単純な多数決ではなく、
重み付き投票で識別結果を算出

アンサンブル学習：AdaBoost

□ AdaBoost

- 重み付きデータと重み付き識別器を用いる手法
 - ・ ブースティングアルゴリズムの代表例

□ データの重み

- 誤識別されたデータの重みの和と
正しく識別されたデータの重みの和を等しくする
 - ・ t 段階目の識別器の誤りを ϵ_t とすると
 - 誤識別されたデータの重み： $\frac{1}{2\epsilon_t}$
 - 正しく識別されたデータの重み： $\frac{1}{2(1-\epsilon_t)}$
- 更新後の
それぞれの重みの総和が1/2
- 誤識別データの重みが大きくなるように更新する
 - ・ よって、更新中に $\epsilon_t \geq 0.5$ となると繰り返しを中断させる

アンサンブル学習 : AdaBoost

□ 識別器の重み α_t

- 誤り率 ϵ_t が増えるにしたがって、 α_t が小さくなり、 $\epsilon_t = 0.5$ で $\alpha_t = 0$ となるような算出式を用いる

$$\alpha_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$$

- 上記の重みを用いることで、以下の指数損失を最小化する

$$\sum_{i=1}^N \exp\{-\hat{c}(\mathbf{x}_i)y_i\}$$

- $\hat{c}(\mathbf{x}_i)$: 統合後の識別器の出力
- $y_i = \{1, -1\}$: 正解情報

AdaBoost : アルゴリズム

入力 : 正解付学習データ D

出力 : 重み付き識別器集合

データの重み $w_1^{(i)}$ を全て $1/N$ とする。ただし、 $i = 1, \dots, N$ 、 N はデータ数

$t = 1$

while $0 < \epsilon_t < 1/2$ かつ繰り返し回数 t が制限 T 以下 **do**

識別器 $h_t(x)$ を学習

/* 誤り率を計算 */

$$\epsilon_t = \sum_{i=1}^N w_t^{(i)} I(y_i \neq h_t(x_i))$$

$I(\quad)$

引数の条件が成立すれば1、
そうでなければ0を返す関数

/* 識別器の重みを計算 */

$$\alpha_t = \ln \sqrt{(1 - \epsilon_t) / \epsilon_t}$$

/* データの重みを更新 */

$$w_{t+1}^{(i)} \leftarrow w_1^{(i)} / 2\epsilon_t, \text{ 誤識別されたデータ } x_i \in D$$

$$w_{t+1}^{(j)} \leftarrow w_1^{(j)} / 2(1 - \epsilon_t), \text{ 正しく識別されたデータ } x_j \in D$$

end while

$$\text{return } \hat{c}(x) = \sum_{i=1}^t \alpha_i h_i(x)$$

勾配ブースティング

□ 勾配ブースティング

- 損失が最も減るような識別器を直接加える処理を逐次的に行う手法
- 全体の識別器の出力： M 個の識別器 h_m の重み付き和

$$F(\mathbf{x}) = \sum_{m=1}^M \alpha_m h_m(\mathbf{x}; \gamma_m) \quad \gamma_m : \text{関数 } h_m \text{ のパラメータ}$$

- この形式でブースティングにおける逐次的な学習プロセス

$$\underbrace{F_m(\mathbf{x})}_{m\text{個分の識別器の出力}} = \underbrace{F_{m-1}(\mathbf{x})}_{m-1\text{個分の識別器の出力}} + \underbrace{\alpha_m h_m(\mathbf{x}; \gamma_m)}_{m\text{番目の識別器の出力を } \alpha_m \text{ で重み付け}$$

勾配ブースティング

□ 勾配ブースティング（つづき）

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \alpha_m h_m(\mathbf{x}; \gamma_m)$$

- 新たに導入した損失関数 L が最小となるものを新たに加える識別器として選択

$$(\alpha_m, \gamma_m) = \operatorname{argmin}_{\alpha, \gamma} \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \alpha h(\mathbf{x}_i; \gamma))$$

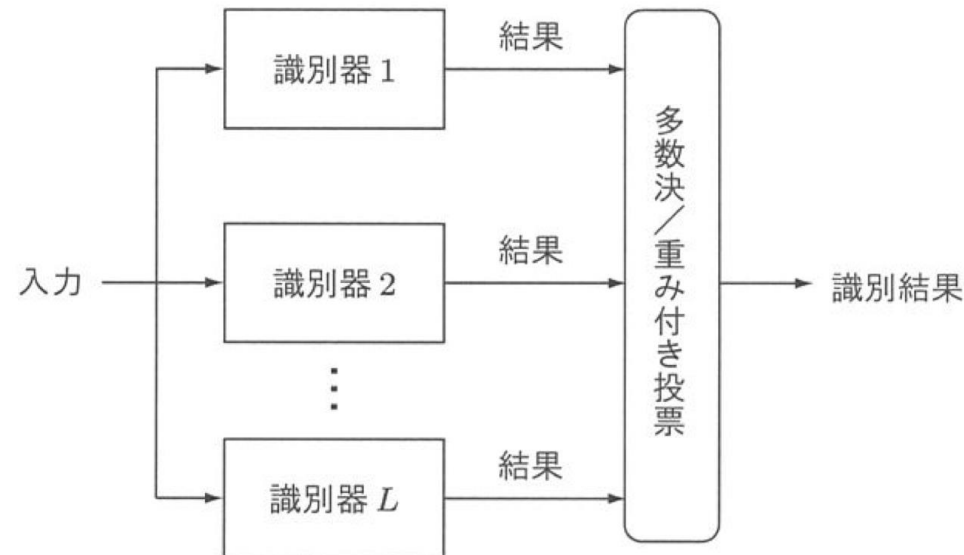
- 損失関数の最小化の部分を、損失関数の勾配として、新しい識別器を構成するのが勾配ブースティング

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \alpha_m \sum_{i=1}^N \nabla_F L(y_i, F_{m-1}(\mathbf{x}_i))$$

※ 損失関数として、二乗誤差、誤差の絶対値、フーバー損失などが用いられる

演習問題10-2（10分間）

- この講義ではアンサンブル学習によって識別問題を解く方法を紹介した。これを回帰問題に適用するにはどうすれば良いか考えなさい。



アンサンブル学習による識別問題