ＴＲ－ＳＬＴ－００７９

アンサンブル学習を用いた音韻継続長の予測法

Segmental duration modeling using ensemble learning

山岸　順一　　河井　恒

Junichi Yamagishi　Hisashi Kawai

2004 年 8 月 25 日

概要

テキスト音声合成システムにおいて音韻継続長は合成音声のリズムやテンポの制御を担う重要な特徴量の一つであり、音韻継続長の制御は合成音声の品質を左右する重要な問題である。この音韻継続長の制御は音韻・韻律情報や言語情報を説明変数とした予測問題と考えられ、回帰木やニューラルネットを用いた手法などがいくつか提案されている。本研究では、これらの手法の音韻継続長の予測性能を効率的に向上・改善させるため、Gradient Boosting と呼ばれる逐次型アンサンブル学習を利用し、音韻継続長予測モデルの性能を効率的に向上させることを検討する。この手法は現在の予測モデルにおける残差のみを考慮すればよいため、Bagging などの並列型アンサンブル学習よりも圧倒的に少ないパラメータで性能を向上させることができることが特徴である。ATR 音韻バランス 503 文を用いた実験においては、ベースとなる学習モデルに回帰木を用いた場合、単体の回帰木の場合と比べ、本手法は平均二乗誤差で約 1.30[ms] 程度予測誤差を減少させることができることがわかった。

（株）国際電気通信基礎技術研究所
音声言語コミュニケーション研究所
〒619－0288「けいはんな学研都市」光台二丁目 2 番地 2 TEL：0774－95－1301

Advanced Telecommunications Research Institute International
Spoken Language Translation Research Laboratories
2-2-2 Hikaridai “Keihanna Science City” 619-0288,Japan
Telephone:+81-774-95-1301
Fax　　:+81-774-95-1308

目 次

1 はじめに **2**

2 アンサンブル学習 **2**
2.1 並列型アンサンブル学習：Bagging 2
2.2 逐次型アンサンブル学習：Gradient Boosting 2
2.3 回帰木のアンサンブル学習：Gradient Tree Boosting 3

3 実験 **5**
3.1 実験条件 5
3.2 収束パラメータνおよびパラメータ数の検討 7
3.3 客観評価結果 8

4 まとめ **9**

A 他の学習モデルにおける比較 **11**

B 他の予測モデルとの比較 **12**

1 はじめに

テキスト音声合成システムにおいて音韻継続長は合成音声のリズムやテンポの制御を担う重要な特徴量の一つである。この音韻継続長の予測／制御には音韻・韻律情報や言語情報を説明変数とした音韻継続長関数の関数近似問題と考える手法が広く用いられている。この音韻継続長 (以下継続長と呼ぶことにする。) の予測問題に対してこれまで、重回帰/数量化I類を用いた手法 [1], CART (Classification and Regression Tree) [2] などの回帰木を用いた手法 [3], ニューラルネットを用いた手法 [4][5], sum-of-products を用いた手法 [6] などが提案されている。また、並列型アンサンブル学習の手法の一つである Bagging (Bootstrap and Aggregating) [7] を CART による予測手法に応用し、予測性能を向上させる試みも行われている [8]。

そこで本研究では、音韻継続長の予測性能をさらに向上させるため逐次型アンサンブル学習を導入することを試みる。本研究では、Gradient Boosting[9][10] と呼ばれる加法型回帰モデルを導入することを試みる。Gradient Boosting は損失関数上で逐次最適化を行うことによってベースとなる学習モデルを結合し、予測モデルとなる加法モデルを構成する手法である。本実験では、加法モデルのベースとなる学習モデルには回帰木 (Regression Treee) を用い、予測誤差による客観評価を行った。また Bagging を用いたアンサンブル手法と本手法との比較も行った。

2 アンサンブル学習

2.1 並列型アンサンブル学習：Bagging

ここでは、並列型アンサンブル学習である Bagging アルゴリズム [7] について簡単に説明する。

ある入力ベクトル $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ の正解値を y とする。学習データは N 個、$\{y_i, \boldsymbol{x}_i\}_1^N$ であるとする。このとき、M 個の異なる学習モデル $h(\boldsymbol{x}, \boldsymbol{a}_m)$ を加算し、単純に平均をとることで新たな予測モデル $F(\boldsymbol{x})$ を作るアルゴリズムが Bagging アルゴリズムである。

$$F(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} h(\boldsymbol{x}, \boldsymbol{a}_m) \tag{1}$$

ここで $\boldsymbol{a}_m$ は m 番目の学習モデルのパラメータセットを表している。Bagging アルゴリズムでは学習データに対してブートストラップ法 (復元抽出) を繰り返し行い、擬似的な学習データセットを M 個作り出し、これらをもとに複数の学習モデル $h(\boldsymbol{x}, \boldsymbol{a}_m)$ を独立に構築する。この並列型のアンサンブル学習は学習データに特異なデータが含まれている場合に汎化性を高める能力を持っているが、質の良い学習データに対しては性能を下げる可能性もある。またパラメータ数も非常に多くなり，効率的な手法とは言い難い.

2.2 逐次型アンサンブル学習：Gradient Boosting

ここでは、本研究において逐次型アンサンブル学習として導入した Gradient Boosting アルゴリズム [9] について簡単に説明する。

Bagging アルゴリズムと同様に、ある入力ベクトル $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ の正解値を y とする。ただし、学習データは N 個、$\{y_i, \boldsymbol{x}_i\}_1^N$ であるとする。このとき、M 個の異なる学習モデル $h(\boldsymbol{x}, \boldsymbol{a}_m)$ を加法的に結合し、

新たな予測モデル $F(\boldsymbol{x})$ を作ることを考える。

$$F(\boldsymbol{x}) = \sum_{m=0}^{M} \beta_m \, h(\boldsymbol{x}, \boldsymbol{a}_m) \tag{2}$$

ここで $\boldsymbol{a}_m$ は m 番目の学習モデルのパラメータセットを表している。一般的に Boosting と呼ばれる逐次型アンサンブル学習では、ベースとなる学習モデルのパラメータセット $\boldsymbol{a}_m$ および各学習モデルの重み係数 β_m を損失関数 $\Psi(y, F(\boldsymbol{x}))$ 上で逐次的に最適化を行うことにより求め、予測モデルとなる加法モデルを構成している。つまり、$m-1$ 番目までの学習モデルを加法したモデルを $F_{m-1}(\boldsymbol{x})$ と表すと m 番目に加法する学習モデルのパラメータセット $\boldsymbol{a}_m$ および現在の学習モデルの重み係数 β_m は以下のように求める。

$$(\beta_m, \boldsymbol{a}_m) = \underset{\beta, \boldsymbol{a}}{\operatorname{argmin}} \sum_{i=1}^{N} \Psi(y_i, F_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i, \boldsymbol{a})) \tag{3}$$

そして m 番目までの学習モデルを加法したモデル $F_m(\boldsymbol{x})$ は β_m および $\boldsymbol{a}_m$ を用いて

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \beta_m h(\boldsymbol{x}, \boldsymbol{a}_m) \tag{4}$$

として与えればいことになる[1]。

しかし、任意の損失関数において式 (3) を解くことは非常に困難な問題である。そこで、Gradient Boosting[9] では上記の問題を $\boldsymbol{a}_m$ を求めるステップおよび β_m を求めるステップの 2 ステップにわけることで任意の損失関数 $\Psi(y, F(\boldsymbol{x}))$ に対して近似的に式 (3) を求めることを可能としている。学習モデルのパラメータ $\boldsymbol{a}_m$ を求めるスッテプでは、現在の加法モデル $F_{m-1}(\boldsymbol{x})$ と正解値 y との損失関数上において

$$\tilde{y}_{im} = -\left[\frac{\partial \Psi(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})} \tag{5}$$

により与えれる $\tilde{y}_{im}$ を学習データ i の現時点における仮残差とし、この仮残差 $\tilde{y}_{im}$ に対する二乗誤差を最小にするモデルパラメータを $\boldsymbol{a}_m$ として求める。

$$\boldsymbol{a}_m = \underset{\boldsymbol{a}, \rho}{\operatorname{argmin}} \sum_{i=1}^{N} [\tilde{y}_{im} - \rho h(\boldsymbol{x}, \boldsymbol{a})]^2 \tag{6}$$

そして、上記の $\boldsymbol{a}_m$ により与えられる学習モデル $h(\boldsymbol{x}, \boldsymbol{a}_m)$ をもとに、与えられた損失関数上での誤差が最小になるように学習モデルの重み係数 β_m を決定する。

$$\beta_m = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \Psi(y_i, F_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i, \boldsymbol{a}_m)) \tag{7}$$

2.3 回帰木のアンサンブル学習：Gradient Tree Boosting

次に Gradient Boosting アルゴリズムにおいてベースの学習モデルとして L 個のリーフノードにより構成される回帰木 (Regression tree) を用いた場合について説明する。この手法は Gradient Tree Boosting もしくは Multiple Additive Regression Trees(MART) と呼ばれている．L 個のリーフノードを持つ回帰木は入力ベクト

[1]損失関数を $\Psi(y, F) = e^{-yF}$ とすれば Adaboost[11] アルゴリズムである。

ル空間 $\boldsymbol{x}$ を互いに素な L 個の空間 $\{R_{lm}\}_{l=1}^{L}$ に分割し、各空間である定数を返す予測モデルであるので、m 番目の繰り返しにおける学習モデルは以下のように表すことが出来る。

$$h(\boldsymbol{x}, \{R_{lm}\}_{l=1}^{L}) = \sum_{l=1}^{L} \bar{y}_{lm} 1(\boldsymbol{x} \in R_{lm}) \tag{8}$$

ここで

$$\bar{y}_{lm} = mean_{\boldsymbol{x}_i \in R_{lm}}(\tilde{y}_{im}) \tag{9}$$

であり、$1(\cdot)$ は引数が真のとき 1 を返し、真でないときは 0 を返す関数である。

回帰木は各リーフノードである定数を返すモデルであるので式 (7) は以下のように簡単化して書くことができる。

$$\gamma_{lm} = \underset{\gamma}{\text{argmin}} \sum_{\boldsymbol{x}_i \in R_{lm}} \Psi(y_i, F_{m-1}(\boldsymbol{x}_i) + \gamma) \tag{10}$$

ただし、ここで $\gamma_{lm} = \beta_m \bar{y}_{lm}$ である。また同様に式 (4) は以下のように書くことができる。

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \gamma_{lm} 1(\boldsymbol{x} \in R_{lm}) \tag{11}$$

ここで新たに収束パラメータ ν を上式に導入する。この収束パラメータは $\nu \leq 1$ のとき汎化誤差を少なくすることが経験的に知られている。この収束パラメータ ν を用いると式 (11) は以下のようになる。

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \nu \cdot \gamma_{lm} 1(\boldsymbol{x} \in R_{lm}) \tag{12}$$

これらのことをまとめるとベースの学習モデルに回帰木を用いた場合の Gradient Boosting アルゴリズムは以下のようになる。

Algorithm : Gradient Tree Boosting

1 $F_0(\boldsymbol{x}) = \text{argmin}_\gamma \sum_{i=1}^{N} \Psi(y_i, \gamma)$
2 For $m = 1$ to M do:
3 $\quad \tilde{y}_{im} = -\left[\frac{\partial \Psi(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})}, i = 1, \cdots, N$
4 $\quad \{\tilde{y}_{im}, \boldsymbol{x}_i\}_1^N$ をもとに回帰木を構築：リーフノード $\{R_{lm}\}_1^L$
5 $\quad \gamma_{lm} = \text{argmin}_\gamma \sum_{\boldsymbol{x}_i \in R_{lm}} \Psi(y_i, F_{m-1}(\boldsymbol{x}_i) + \gamma)$
6 $\quad F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \nu \cdot \gamma_{lm} 1(\boldsymbol{x} \in R_{lm})$
7 endFor

損失関数には二乗誤差 $\Psi(y, F) = (y - F)^2$ や絶対誤差 $\Psi(y, F) = |y - F|$ や Huber M $\Psi(y, F) = (y - F)^2 1(|y - F| \leq \delta) + 2\delta(|y - F| - \delta/2) 1(|y - F| > \delta)$ などが用いられることが多い。なお本研究では損失関数 $\Psi(y, F)$ として二乗誤差 $\Psi(y, F) = (y - F)^2/2$ を用いた．この損失関数 $\Psi(y, F) = (y - F)^2/2$ においては，$\rho = \beta$ となり，さらに仮残差は $\tilde{y}_{im} = y_i - F_{m-1}(\boldsymbol{x}_i)$ となる．これは単純に現在の予測モデルにおける残差をもとに逐次的に次の予測モデルを構築するアルゴリズムとなり，一般化加法モデル (Generalized Additive Model) に似たアルゴリズムになる[2]。

[2]一般的に boosting では繰り返しごとに学習データの分布の重みを変える。本アルゴリズムではそのような行程はない。従って本実験に用いた Gradient Boosting アルゴリズムは一般的な boosting とは異なり、加法型回帰モデル (Additive Regression Model) に近いアルゴリズムであると言える。

3 実験

3.1 実験条件

提案法の有効性を示すため、音素継続時間長の予測誤差による客観評価を行った。実験に用いたデータは話者 M007 および話者 F009 の ATR 音韻バランス文セット B の 503 文章である。本実験では 503 文中の 400 文章を学習データとして用い、残りの 103 文章をテストデータとして用いた。なお、実験に用いた音素継続時間のラベルはハンドラベリングにより与えている。アンサンブル学習のベースとなる学習モデルの構築には 2 分木の回帰木 (Regression Tree) を用いた。ただし、回帰木の構築後、枝狩りおよびスムージングをおこなっている [12]。学習データの説明変数としては以下の 47 種類の音韻・韻律情報や言語情報を用いた。

1. 先先行音素
2. 先行音素
3. 当該音素
4. 後続音素
5. 後後続音素
6. アクセント核とモーラ位置との差 (単位：モーラ)
7. 前からカウントした当該韻律語内での当該モーラ位置 (単位：モーラ)
8. 後ろからカウントした当該韻律語内での当該モーラ位置 (単位：モーラ)
9. 先行形態素の品詞
10. 先行形態素の品詞の活用形
11. 先行形態素の品詞の活用型
12. 先行形態素の境界の種別
13. 当該形態素の品詞
14. 当該形態素の品詞の活用形
15. 当該形態素の品詞の活用型
16. 当該形態素の境界の種別
17. 後続形態素の品詞
18. 後続形態素の品詞の活用形
19. 後続形態素の品詞の活用型
20. 後続形態素の境界の種別

21. 先行韻律語の長さ (単位：モーラ)
22. 先行韻律語のアクセント型
23. 先行韻律語と当該韻律語間のポーズの有無
24. 当該韻律語の長さ (単位：モーラ)
25. 当該韻律語のアクセント型
26. 前からカウントした当該呼気段落での韻律語の位置 (単位：韻律語)
27. 後ろからカウントした当該呼気段落での韻律語の位置 (単位：韻律語)
28. 前からカウントした当該呼気段落での韻律語の位置 (単位：モーラ)
29. 後ろからカウントした当該呼気段落での韻律語の位置 (単位：モーラ)
30. 後続韻律語の長さ (単位：モーラ)
31. 後続韻律語のアクセント型
32. 後続韻律語と当該韻律語間のポーズの有無
33. 先行呼気段落の長さ (単位：韻律語)
34. 先行呼気段落の長さ (単位：モーラ)
35. 当該呼気段落の長さ (単位：韻律語)
36. 当該呼気段落の長さ (単位：モーラ)
37. 前からカウントした文中での当該呼気段落の位置 (単位：呼気段落)
38. 後ろからカウントした文中での当該呼気段落の位置 (単位：呼気段落)
39. 前からカウントした文中での当該呼気段落の位置 (単位：韻律語)
40. 後ろからカウントした文中での当該呼気段落の位置 (単位：韻律語)
41. 前からカウントした文中での当該呼気段落の位置 (単位：モーラ)
42. 後ろからカウントした文中での当該呼気段落の位置 (単位：モーラ)
43. 後続呼気段落の長さ (単位：韻律語)
44. 後続呼気段落の長さ (単位：モーラ)
45. 文の長さ (単位：呼気段落)
46. 文の長さ (単位：韻律語)
47. 文の長さ (単位：モーラ)

表 1: Comparison of number of leaf nodes. Basic learning algorithm is regression tree, and 10 classifiers are made in ensemble method. Regression tree is constructed at each target class.

			Number of Ensumble										
Speaker	Target	Method	1	2	3	4	5	6	7	8	9	10	RMSE[ms]
M007	Vowel	Regression Tree	146	-	-	-	-	-	-	-	-	-	17.49
		Bagging	208	244	216	255	256	230	153	194	216	207	17.05
		Boosting $\nu = 1.0$	146	58	18	17	1	-	-	-	-	-	16.71
		Boosting $\nu = 0.7$	146	70	57	31	22	23	11	1	1	1	16.56
		Boosting $\nu = 0.5$	146	122	61	47	27	30	15	1	1	1	16.19
		Boosting $\nu = 0.3$	146	111	81	67	55	45	55	48	25	26	16.17
		Boosting $\nu = 0.1$	146	152	131	129	105	96	87	95	73	92	18.97
		Boosting $\nu = 0.05$	146	150	156	131	139	119	119	130	122	105	22.47
	Consonant	Regression Tree	223	-	-	-	-	-	-	-	-	-	14.21
		Bagging	341	303	299	318	259	333	256	280	257	323	13.57
		Boosting $\nu = 1.0$	223	87	37	7	22	1	-	-	-	-	13.31
		Boosting $\nu = 0.7$	223	137	73	37	30	19	5	12	10	1	12.91
		Boosting $\nu = 0.5$	223	167	117	72	73	45	17	23	1	1	12.63
		Boosting $\nu = 0.3$	223	175	144	116	95	92	58	60	15	51	12.75
		Boosting $\nu = 0.1$	223	190	196	189	182	167	163	154	127	143	15.60
		Boosting $\nu = 0.05$	223	217	200	201	192	188	189	170	175	173	19.47
F009	Vowel	Regression Tree	166	-	-	-	-	-	-	-	-	-	15.21
		Bagging	187	278	242	268	282	270	211	246	269	217	14.81
		Boosting $\nu = 1.0$	166	67	27	29	4	12	14	1	-	-	14.71
		Boosting $\nu = 0.7$	166	86	58	40	9	18	1	1	1	1	14.37
		Boosting $\nu = 0.5$	166	93	59	58	32	32	21	17	18	1	14.03
		Boosting $\nu = 0.3$	166	99	134	80	63	68	43	31	35	20	14.19
		Boosting $\nu = 0.1$	166	179	114	111	87	81	113	103	90	111	16.10
		Boosting $\nu = 0.05$	166	166	171	134	112	128	113	120	96	108	18.06
	Consonant	Regression Tree	240	-	-	-	-	-	-	-	-	-	13.29
		Bagging	320	301	270	317	331	374	367	404	337	356	13.17
		Boosting $\nu = 1.0$	240	77	30	1	-	-	-	-	-	-	12.18
		Boosting $\nu = 0.7$	240	135	73	35	38	1	1	1	1	1	11.98
		Boosting $\nu = 0.5$	240	162	96	97	46	42	1	1	1	1	11.84
		Boosting $\nu = 0.3$	240	192	133	111	85	108	72	58	26	35	11.99
		Boosting $\nu = 0.1$	240	206	203	170	175	184	163	153	140	147	17.24
		Boosting $\nu = 0.05$	240	204	198	228	180	189	193	162	186	162	22.64

3.2 収束パラメータ ν およびパラメータ数の検討

Gradient Boosting アルゴリズムにおける収束パラメータ ν は $\nu \leq 1$ のとき汎化誤差を少なくできることが経験的に知られている。そこで、収束パラメータを $\nu = 0.05,\ 0.1,\ 0.3,\ 0.5,\ 0.7,\ 1.0$ とし、テストデータに対

する平均二乗誤差を各値ごとに求めた。また、10回の繰り返しにおける回帰木のリーフノード数の変化およびBaggingアルゴリズムにおける回帰木のリーフノード数の変化も比較のため調べた。表1にその結果を示す。

この表において“M007”と“F009”はそれぞれ話者M007およびF009による結果を示し、“Vowel”と“Consonant”はそれぞれ母音を学習データとしたモデルにおける母音によるテストデータの評価結果、子音を学習データとしたモデルにおける子音によるテストデータの評価結果を示す。また“Regression Tree”、“Bagging”、“Boosting”はそれぞれ単体の回帰木による結果、Baggingによるアンサンブル学習を行った回帰木を用いた結果、Gradient Boostingによるアンサンブル学習を行った回帰木を用いた結果を示す。“ν”は収束パラメータ値を示し、表中の数値は各繰り返し(アンサンブル)におけるリーフノード数を表す。また“RMSE[ms]”はテストデータに対する平均二乗誤差を示す。

この表からBaggingアルゴリズムもしくはGradient Boostingアルゴリズムによるアンサンブル学習を行うことで、どちらの話者においても単体の回帰木の場合よりも二乗誤差を少なくすることができることが確認できる。また収束パラメータνを小さくすると、各繰り返しにおける回帰木のリーフノード数が増えていくこともわかり、10回の繰り返しにおいては収束パラメータがおおむね$\nu = 0.5$のとき二乗誤差が最も少なくなっていることがわかる。さらに収束パラメータが$\nu = 0.5$のときのGradient Boostingアルゴリズムは、Baggingアルゴリズムよりも少ないリーフノード(パラメータ)数でより誤差の少ない予測モデルを構築できていることがわかる。本実験の場合、Gradient Boostingアルゴリズムは単体の回帰木の場合と比べ、平均二乗誤差で約1.30[ms]程度、Baggingアルゴリズムと比べ約0.9[ms]程度誤差を減少させることができることがわかる。

3.3 客観評価結果

表2に単体の回帰木、Baggingによる回帰木、Gradient Boosting($\nu = 0.5$)による回帰木におけるテストデータに対する相関係数、平均絶対誤差、平均二乗誤差、ノーマライズド平均二乗誤差を客観評価結果として示す。これらの表において“r”、“AAE”、“RMSE”、“NMSE”はそれぞれピアソンの積率相関係数、平均絶対誤差、平均二乗誤差、ノーマライズド平均二乗誤差を示す。これらの客観評価結果からもGradient Boostingアルゴリズム($\nu = 0.5$)は、Baggingアルゴリズムよりも誤差の少ない予測モデルを構築できることがわかる。

表2: Comparison of duration modeling for speakers M007 and F009. Basic learning algorithm is regression tree, and 10 classifiers were made in ensemble method. r is correlation coefficient, AAE is average absolute error in [ms], RMSE is root mean squared error in [ms], and NMSE is normalized mean squared error. Regression tree is constructed at each target class.

		Vowel				Consonant			
Speaker	Method	r	AAE	RMSE	NMSE	r	AAE	RMSE	NMSE
M007	Regression Tree	0.81	13.19	17.49	0.34	0.85	10.67	14.21	0.28
	Bagging	0.82	12.90	17.05	0.33	0.86	10.27	13.57	0.25
	Boosting $\nu = 0.5$	0.84	12.17	16.19	0.29	0.88	9.47	12.63	0.22
F009	Regression Tree	0.72	11.50	15.21	0.48	0.91	9.89	13.29	0.17
	Bagging	0.75	11.10	14.81	0.45	0.92	9.83	13.17	0.17
	Boosting $\nu = 0.5$	0.77	10.46	14.03	0.40	0.93	8.84	11.84	0.13

4 まとめ

本研究では、音韻継続長の予測性能をさらに向上させるため Gradient Boosting と呼ばれる逐次型アンサンブル学習を導入した。Gradient Boosting では損失関数上で逐次最適化を行うことによってベースとなる学習モデルを結合し、予測モデルとなる加法モデルを構成する。本実験では加法モデルのベースとなる学習モデルに回帰木を用い、オープンデータに対する予測誤差による客観評価を行った。客観評価の結果、Gradient Boosting アルゴリズムは、Bagging アルゴリズムよりも少ないパラメータ数でより誤差の少ない予測モデルを構築できることがわかった。

今後の課題は本実験で用いた損失関数とは異なる損失関数の利用や学習データの分布の重みを変えるアプローチの導入などが挙げられる。

参考文献

[1] N. Kaiki, T. Takeda and Y. Sagisaka, "Vowel duration control using linguistic information," Trans. IEICE, vol.J75-A, no.3, pp.467–473, March 1992.

[2] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, "Classification and Regression Trees," Wadsworth Statistics/Probability Series, Belmont, 1984

[3] M.D. Riley, "Tree-based modelling of segmental duration," Talking Machines : Theories, Models, Designs, pp.265–273, 1992.

[4] W. Campbell, "Analog I/O nets for syllable timing," Speech Communication, vol.9, pp.57–61, 1990.

[5] M.Riedi, "A neural-network-based model of segmental duration for speech synthesis," Proc. EUROSPEECH-95, pp.599–602 Sept. 1995.

[6] J.P.H. van Santen, "Assignment of segmental duration in text-to-speech synthesis," Computer Speech and Language, vol.8, pp.95–128, 1994.

[7] L. Breiman, "Bagging predictors," Machine Learning, no.24, pp.123–140, 1996.

[8] S. Lee and Y. Oh, "CART-based modelling of Korean segmental duration," Proc. Oriental COCOSDA '99, pp.109–112, 1999.

[9] J.H. Friedman, "Greedy function approximation : A gradient boosting machine," Annals of Statistics, vol.29, no.5, pp.1189–1232, 2001.

[10] J.H. Friedman, "Stochastic gradient boosting," Computational Statistics & Data Analysis, vol.38, no.4, pp.367–378, 2002.

[11] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol.55, no.1, pp.119–139, 1997.

[12] J.R. Quinlan, "Learning with continuous classes," Proc. AI'92, pp.343–348, 1992.

[13] Y. Wang and I.H. Witten, "Inducing model trees for continuous classes," Proc. European Conference on Machine Learning, pp.128–137, 1997.

A 他の学習モデルにおける比較

付録データとして、加法モデルのベースとなる学習モデルに Decision Stump と Model Tree (M5) [12][13] を用いた場合の予測誤差による客観評価結果を示す。Model Tree は入力ベクトル空間を互いに素な空間に分割し、分割された各空間において線形回帰による近似関数を行う予測モデルである。つまり、回帰木と線形回帰を組み合わせたアルゴリズムである。

表 3: Comparison of duration modeling for speakers M007 and F009. Basic learning algorithm is decision stump, and 10 classifiers are made in ensemble method. r is correlation coefficient, AAE is average absolute error in [ms], RMSE is root mean squared error in [ms], and NMSE is normalized mean squared error. Decision stump is constructed at each target class.

		Vowel				Consonant			
Speaker	Method	r	AAE	RMSE	NMSE	r	AAE	RMSE	NMSE
M007	Decision stump	0.67	17.01	22.18	0.55	0.42	19.43	24.39	0.82
	Bagging	0.67	17.01	22.19	0.55	0.42	19.41	24.38	0.82
	Boosting	0.74	15.54	20.16	0.46	0.68	16.28	20.73	0.59
F009	Decision stump	0.32	16.21	20.76	0.89	0.46	23.94	28.79	0.79
	Bagging	0.33	16.21	20.76	0.89	0.46	23.94	28.78	0.79
	Boosting	0.58	14.34	18.53	0.40	0.73	18.53	23.46	0.53

表 4: Comparison of duration modeling for speakers M007 and F009. Basic learning algorithm is model tree, and 10 classifiers are made in ensemble method. r is correlation coefficient, AAE is average absolute error in [ms], RMSE is root mean squared error in [ms], and NMSE is relative mean squared error (normalized mean squared error). Model tree is constructed at each target class.

		Vowel				Consonant			
Speaker	Method	r	AAE	RMSE	NMSE	r	AAE	RMSE	NMSE
M007	Model Tree	0.83	12.53	16.68	0.31	0.85	10.57	14.19	0.28
	Bagging	0.85	12.00	16.00	0.29	0.88	9.64	12.80	0.22
	Boosting	0.84	12.23	16.25	0.30	0.88	9.62	12.85	0.23
F009	Model Tree	0.75	10.94	14.56	0.44	0.92	9.10	12.29	0.14
	Bagging	0.76	10.71	14.32	0.42	0.93	8.73	11.72	0.13
	Boosting	0.76	10.69	14.25	0.42	0.93	8.77	11.68	0.13

表 3 および表 4 にそれぞれ Decision Stump と Model Tree (M5) を用いた結果を示す。これらの表において "M007" と "F009" はそれぞれ話者 M007 および F009 による結果を示し、"Vowel" と "Consonant" はそれぞれ母音を学習データとしたモデルにおける母音によるテストデータの評価結果、子音を学習データとしたモデルにおける子音によるテストデータの評価結果を示す。また "Decision Stump"、"Model Tree"、"Bagging"、

"Boosting" はそれぞれ単体の Decision Stump による結果、単体の Model Tree (M5) による結果、Bagging によるアンサンブル学習を行った結果、Gradient Boosting によるアンサンブル学習を行った結果を示す。これらの表において "r"、"AAE"、"RMSE"、"NMSE" はそれぞれピアソンの積率相関係数、平均絶対誤差、平均二乗誤差、ノーマライズド平均二乗誤差を示す。

この表から Decision Stump においては単体の場合よりも予測精度を非常に改善できることがわかるが、Model Tree (M5) においては単体の場合と比べ予測精度をそれほど改善できないことがわかる。このことから本実験に用いた逐次型アンサンブル学習は、ベースとなる学習モデルの予測精度が低い場合には予測精度を効果的に改善する働きをするが、ベースとなる学習モデルの精度が高い場合には予測精度を効果的には改善できないことがわかる。

B 他の予測モデルとの比較

付録データとして、重回帰/数量化 I 類を用いた手法 [1] との予測誤差による比較結果を表 5 に示す。

表 5: Comparison of duration modeling using linear regression method. r is correlation coefficient, AAE is average absolute error in [ms], RMSE is root mean squared error in [ms], and NMSE is normalized mean squared error.

		Vowel				Consonant			
Speaker	Method	r	AAE	RMSE	NMSE	r	AAE	RMSE	NMSE
M007	Linear Regression	0.83	12.62	16.79	0.32	0.83	11.39	15.03	0.31
	Regression Tree	0.81	13.19	17.49	0.34	0.85	10.67	14.21	0.28
	Boosting $\nu = 0.5$	0.84	12.17	16.19	0.29	0.88	9.47	12.63	0.22
F009	Linear Regression	0.75	10.98	14.63	0.44	0.88	11.44	15.18	0.22
	Regression Tree	0.72	11.50	15.21	0.48	0.91	9.89	13.29	0.17
	Boosting $\nu = 0.5$	0.77	10.46	14.03	0.40	0.93	8.84	11.84	0.13