TR-I-0168

統計による音声認識候補の絞り込みに関する考察
— 分散特性と2階差分値を用いた手法 —

Reduction of Speech Recognition Candidates by Statistical Method

坂野俊哉　　森元逞

Toshiya Sakano　Tsuyoshi Morimoto

1990.8

概要

音声認識システムから得られる認識度合付き音声認識候補に対し、その認識度合の分散統計特性を用いて他の入力との単純比較が可能な正規化認識確率を計算し、その値が極端に低い候補を除去することにより認識候補の絞り込みを行なう方式と、認識度合の2階差分値を用いて認識候補の絞り込みを行なう手法に関して述べる。

ＡＴＲ自動翻訳電話研究所

ＡＴＲ Interpreting Telephony Research Laboratories

もくじ

1　音声認識候補の尤度計算について

まず音声認識候補に対する尤度計算に関して、以下に説明する。　音声入力に対してその音韻連鎖に対する確率計算によって得られる値（認識度合）を比較し、その値の優劣により認識候補を決定している音声認識システムの手法では、認識度合いをそのままその認識候補の尤度として用いられているが、一般にこの値は対象とする音声入力に依存するために、異なる入力に対する認識候補間の確からしさを比較する場合には、この値を単純に用いることはできない。このことを具体的に説明する。

認識処理により音韻入力ａの音響的な出力候補として、

$$d_1, \cdots, d_n$$

が得られる。各々の出力に対する確率を P_d（音響的出力確率と呼ぶことにする）で表すと、

$$\sum_{i=1}^{n} P_d(d_i) = 1$$

となる。この音響出力 $d_1, \cdots, d_n$ のうちで、言語的に意味を為すｗｏｒｄに変換できるものを

$$d_1, \cdots, d_m \quad (m \leq n)$$

とし、それらをｗｏｒｄに変換したものを $w_1, \cdots, w_m$ とする。（但し、このｗｏｒｄは各候補に対して一意に決る）。即ち、

$$\begin{aligned} d_1 &\rightarrow w_1 \\ &\vdots \\ d_m &\rightarrow w_m \\ d_{m+1} &\rightarrow \phi \\ &\vdots \\ d_n &\rightarrow \phi \end{aligned}$$

（ϕ は言語的な意味を持たないことを表す、）

という対応をとるものとする。

音声認識システムから出力されるのは，ｗｏｒｄ　$w_1,\cdots,w_m$ であり，その出力確率として $P_d(d_1),\cdots,P_d(d_m)$ が出力されている．ここで欲しいのは，ｗｏｒｄ　$w_1,\cdots,w_m$ の間の確からしさ（尤度）である．この尤度を P_w（言語的出力確率と呼ぶことにする）で表すと，

$$\sum_{i=1}^{m} P_w(w_i) = 1$$

となる．この尤度 P_w は直接音声認識システムからは出力されない．

既述したように，音声認識システムから出力される候補はｗｏｒｄ $w_1,\cdots,w_m$ であるが，これらの出力をこの後の，例えば言語翻訳処理に渡すとしても，候補が複数のためにその計算量は大きくなり，かなりの負担を強いられることになる．
この理由から，音声認識システムから出力される複数候補を何らかの方法により少数に絞り込む必要がある．もし，既述したような候補 $w_1,\cdots,w_m$ に対する尤度 P_w が求まれば，この値が極端に低い候補を切り捨てることにより候補の絞り込みが可能となる．

ここで述べる手法の主たる目的は，この音声認識候補に対する尤度 P_w を求め，精度の高い候補の絞り込みを実現することにある．この手法は，音声認識システムにより出力される認識候補の音響的出力確率値の統計的な特性を用いた２階差分法および分散法から構成される．まず，２階差分法により認識候補の絞り込みを行い，分散法によりその結果の言語的出力確率を求める．この際，２階差分法で候補の絞り込みが行えなかった場合でも，分散法により候補の絞り込みが行われる．

より詳しい説明を次節以下で行う．

2　2階差分法

2.1　原理

これは認識候補間の音響的出力確率値の変化特性を用いた手法である.
データ列 $\{D_i\}_i$ $(i \geq 1)$ に対して,

$$\begin{aligned} D_1'' &= D_2 - D_1 \\ D_i'' &= D_{i-1} - 2D_i + D_{i+1} \quad (i \geq 2) \end{aligned}$$

をデータ D_i の2階差分という. データ D_{i-1} からデータ D_i への増分を Δx_i, データ D_i からデータ D_{i+1} への増分を Δx_{i+1} とすると, 2階差分 D_i'' は Δx_i から Δx_{i+1} への増分を表す. 即ち, データが

$$\begin{array}{ccccc} D_{i-1} & \rightarrow & D_i & \rightarrow & D_{i+1} \\ & \Delta x_i & & \Delta x_{i+1} & \end{array}$$

と推移する時のその変化率を表している.

$\{d_i\}_i$ を音声入力 a に対する認識候補列であり, 各々の音響的出力確率の良いものから順序付けされているとする. 2階差分法は認識候補の音響的出力確率 $\{P_a(d_i)\}_i$ の各々に対して2階差分を求め, その最大値（最大2階差分値）を $P_a''(d_j)$ とするならば, この j を音声入力 a の正解ランクとし, j より大きいランクの候補を切り捨てることにより認識候補の絞り込みを行う手法である.

2.2 動作例

音声入力 < *kizituto* > に対して音声認識処理を施し，その結果次のような音声認識候補ラティス（音声認識候補の集合）が得られたとする．（この場合，2番目の候補が正解である．）

(1)	kizitsu-sou-o	(1.642287)
(2)	**kizitsu-to**	**(1.652949)**
(3)	kizitsu-sou	(1.675942)
(4)	chizu-chizu-to	(1.767259)
(5)	ichi-chizu-to	(1.813519)

各候補の右側の数値はその候補に対する音響的出力確率を表すスコアであり，値が小さいほどその候補の認識の確からしさが大きい．これらの候補の2階差分値を計算すると次のようになる．

(1)	1.652949 - 1.642287	=	0.010662
(2)	1.642287 - 2×1.652949 + 1.675942	=	0.012331
(3)	**1.652949 - 2×1.675942 + 1.767259**	**=**	**0.068324**
(4)	1.675942 - 2×1.767259 + 1.813519	=	-0.045057

（候補数が5つだから4番目の候補までの2階差分値が求まる．）

この時，3番目の候補の2階差分値が最も大きいので，4番目と5番目の候補は切り捨てられ，他の3つの候補は残される．

2.3　2階差分法の信頼度

2階差分法の信頼度を示したのが図1である．この図によれば，最大2階差分値が大きいほど2階差分法の信頼度が高いといえる．2階差分法の信頼度が高い音声入力に対しては2階差分法を用い，その他の場合には適用しない．具体的には次のように行なう．図1より2階差分法を適用される最低信頼度を表す閾値を決める．（図1の点Mによりその閾値とその時の最大2階差分値を表している．）

音声入力を認識処理し，その候補ラティスに対する最大2階差分値を計算し，その値が先ほど決めた閾値（図1では0．０６８）以上ならば2階差分法を，そうでなければ分散法をこの音声認識候補ラティスに適用する．

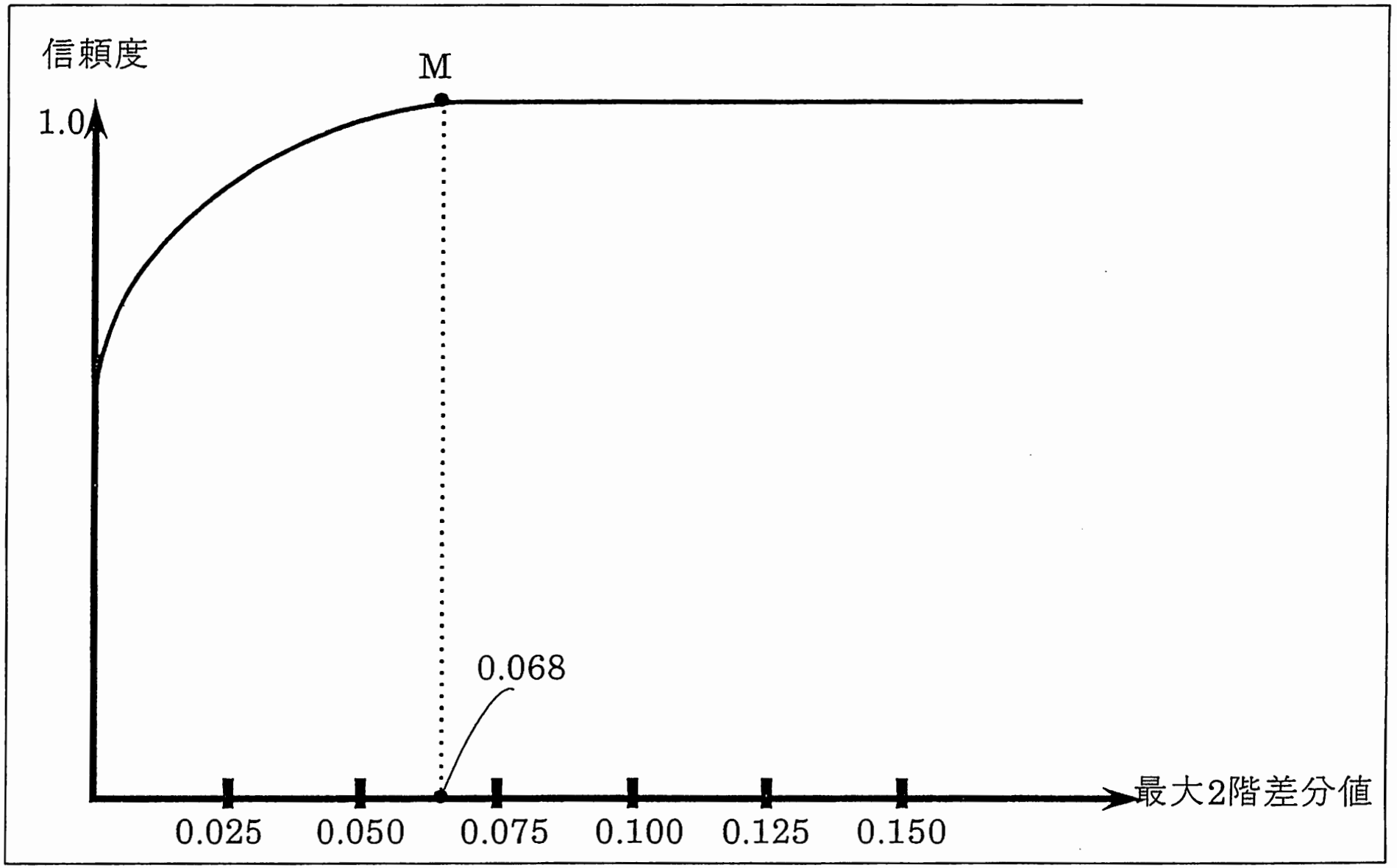

図1　2階差分法の信頼度

3 分散法

3.1 原理

これは音響的出力確率値の分散特性を用いて，各候補に対する正規化確率の計算と候補の絞り込みを行うための手法である．音声認識システムでは音声入力 a を認識処理することにより音響的出力確率の高い方から r 個の認識候補 $d_1, \cdots, d_r$ を出力する．音響的出力確率を P_d で表すと，この候補 $d_1, \cdots, d_r$ の音響的出力確率に対する分散 v は次のようになる．

$$v = \frac{1}{r}\sum_{i=1}^{r}(P_d(d_i) - mean)^2 \quad (1)$$

$$mean = \frac{1}{r}\sum_{i=1}^{r}P_d(d_i) \quad (2)$$

分散は対象となるデータのばらつき具合を表す．
候補 $d_1, \cdots, d_r$ のうちで $d_k(1 \leq k \leq r)$ が音声入力 a の正解とすると，この k を a の正解ランクと呼ぶことにする．

音声入力サンプル $\{a_1, \cdots, a_n\}$ に認識処理を施し，音声入力毎に得られる候補に対する分散と，その正解ランクとの関係をグラフに表したのが図２である．このグラフの横軸が分散，縦軸が正解ランクを表す．ある音韻入力においてその音声認識結果が極端に良い，即ち，ある候補に対する音響的出力確率が極端に良ければ，その候補以外の候補に対する音響的出力確率が低くなり，分散は大きくなる．この時，正解ランクは低くなる傾向がある．
逆に，認識候補に対して一様な音響的出力確率を出力すれば分散は小さくなり，その正解ランクも大きくなる．この理由から図２のグラフは単調減少の傾向を示している．
音声認識システムからの音声認識結果は一様に r 個の候補を出力している．図２で言えば，点線で表されている直線より下の候補を総て出力していることになる．しかし，実際には図２のグラフにしか正解が存在しないために，最適な場合，このグラフで表されている部分だけを候補として出力すれば良い．ただ，このグラフを正確に表現できないので，図２の実線で示される直線によりグラフを近似し，その直線よりも下の候補を出力することにする．これが分散法による候補の絞り込みの原理である．

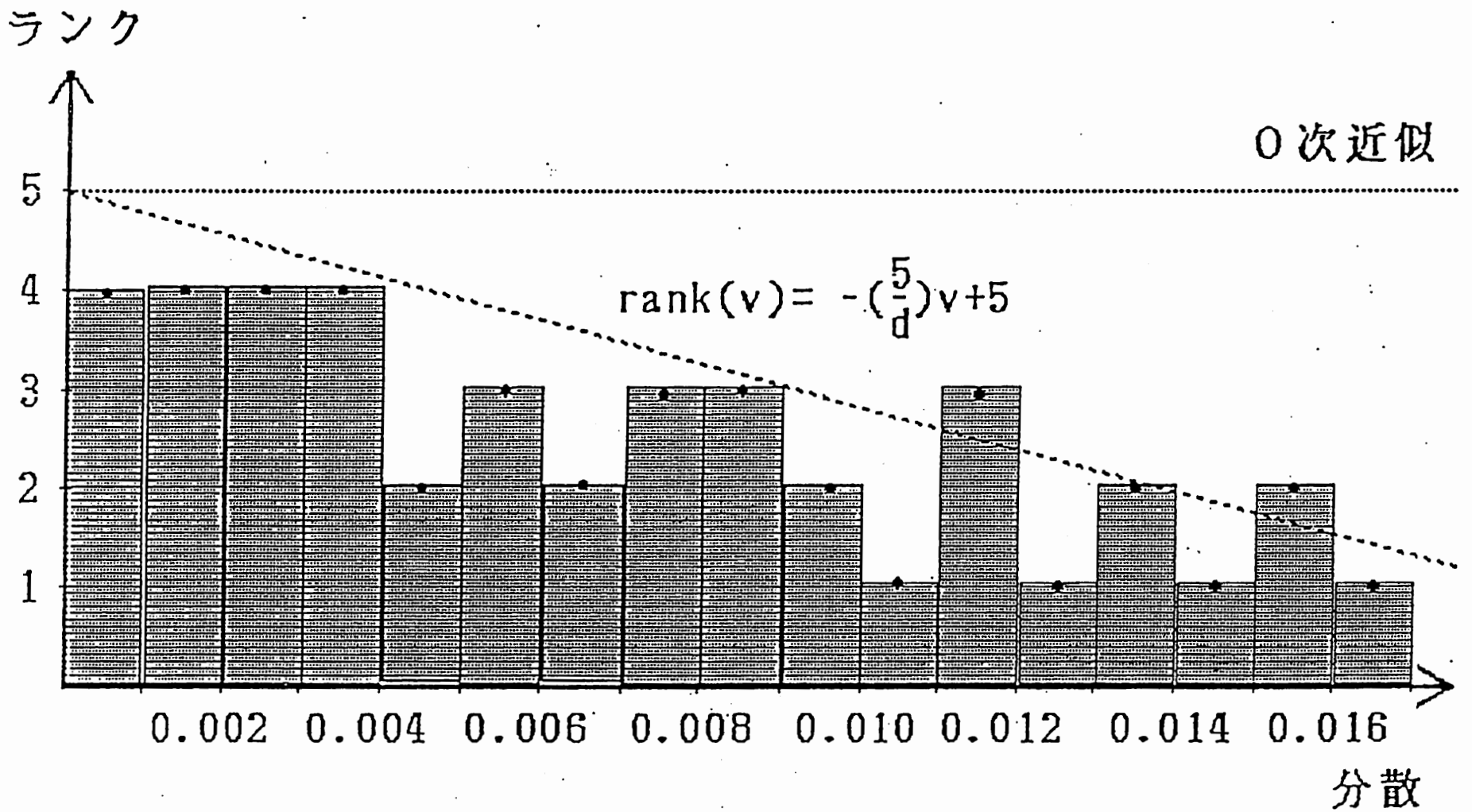
ランク
0次近似
5
4
3
2
1
rank(v)= -(5/d)v+5
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
分散

図2　分散と正解ランク

次に言語的出力確率Ｐｗの求め方を述べる．
音声入力サンプル $\{a_1, \cdots, a_n\}$ のうち，その音響的出力確率の分散が v_1 であるものを

$$\{b_1, \cdots, b_m\}\ (m \leq n)$$

とする．このサンプルを各々の正解ランクを階級として分類し，各階級における正解確率（認識確率）を求めグラフにしたものが図３である．図３で表される正解確率分布は音響的出力確率の分散により異なる．この正解確率分布を P で表すことにすれば，音声入力 a の第 r 位の候補 d_r に対する正解確率は， a の音響的出力確率の分散を v とするならば， r と v により $P(v,r)$ と決定される．正解確率分布 P は言語的に意味のある候補に対する分布であるから，これを近似的に言語的出力確率 P_w とみなす．即ち，

$$P_w(d_r) \quad \approx \quad P(v,r)$$

と書く．これが分散法による言語的出力確率を求めるための原理である．
以上の原理では図２，図３で表されている分布を求めなければならないが，一般にはこの分布を簡単な形式で表現することは困難である．

以下では分散法を実際に運用するための簡便法を述べる．
まず，音声入力サンプル $\{a_1, \cdots, a_n\}$ に対して各正解ランク毎に音響的出力確率の分散と正解確率との分布を１次近似式で求める（分布を直線で近似する）と，図４のようなグラフで表される．分散階級毎に正解確率分布を求めるには，各正解ランクの近似直線からその分散階級に相当する値を求め，それを標準化（即ち，合計が１になるように比例配分する）することにより求める．その際，近似直線の値が０の正解ランクを削除することにより，候補の絞り込みが同時に行われる．但し，２階差分法が適用され，候補が絞り込まれた候補の正規化確率の求め方は次の様にして行なう．絞り込み後の候補数を j とすると，図５においてランク j を表す直線が分散を表す軸と交わる点の分散値 v をこの候補の分散値とみなし，この値を基に分散法により正規化確率を求める．

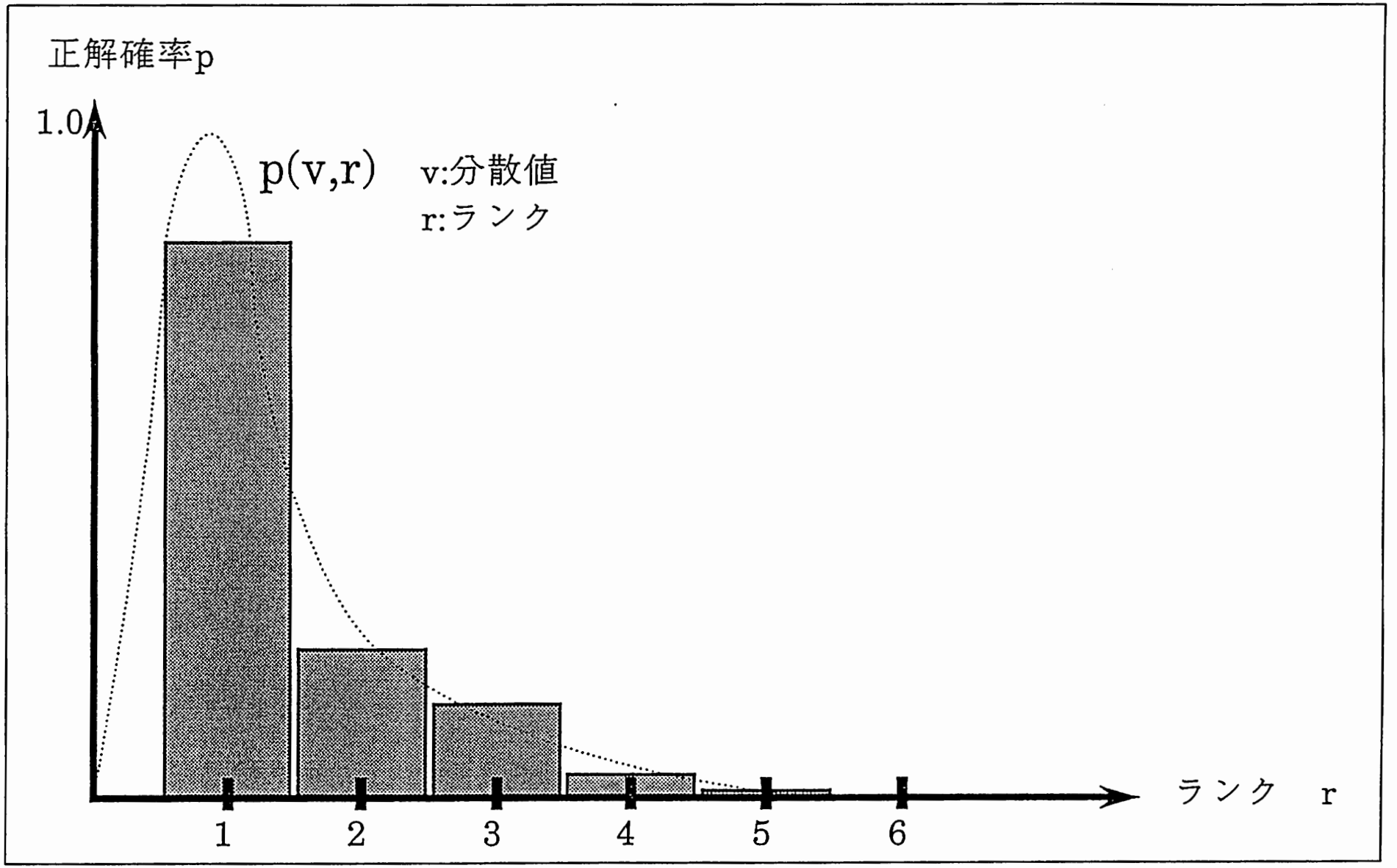

図3　ランクと正解確率のモデル

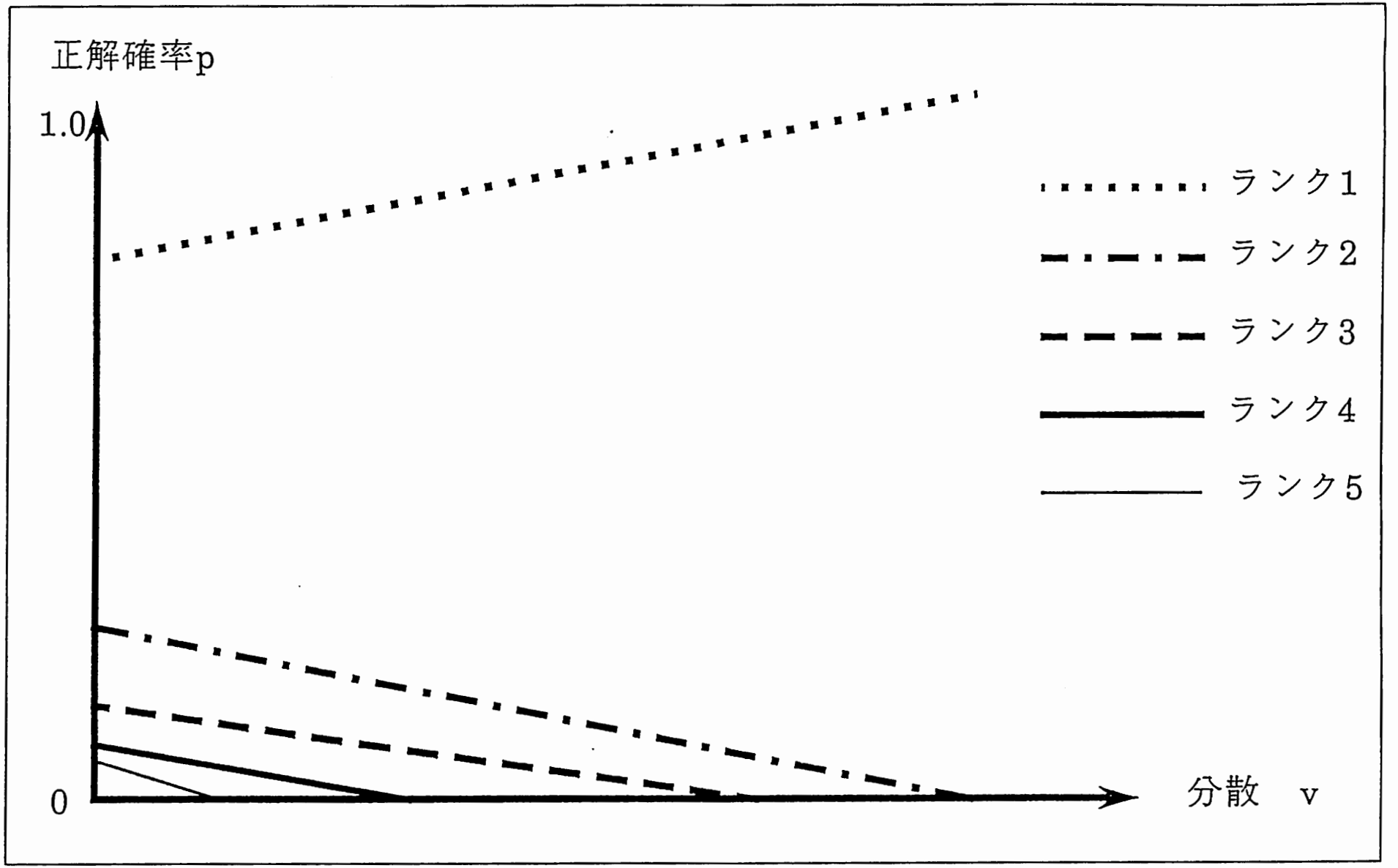

図4　分布モデルp(v,r)の近似

3.2 動作例

音声入力 a の認識候補を $d_1, \cdots, d_5$ とする．この時，音響的出力確率の分散 v_0 は次のように計算できる．

$$v_0 = \frac{1}{5} \cdot \{(P_w(d_1) - m_0)^2 + \cdots + (P_w(d_1) - m_0)^2\} \quad (3)$$

$$m_0 = \frac{1}{5} \cdot \{P_w(d_1) + \cdots + P_w(d_5)\} \quad (4)$$

図5より分散階級 v_0 に相当する各正解ランク毎の正解確率を求め，それを $p'_1, p'_2, p'_3, p'_4, p'_5$ とする．p'_4, p'_5 は，図5より0である．
最後に $p'_1, p'_2, p'_3, p'_4, p'_5$ を次のように標準化する．

$$p_1 = \frac{p'_1}{p'_1 + p'_2 + p'_3 + p'_4 + p'_5} \quad (5)$$

$$p_2 = \frac{p'_2}{p'_1 + p'_2 + p'_3 + p'_4 + p'_5} \quad (6)$$

$$p_3 = \frac{p'_3}{p'_1 + p'_2 + p'_3 + p'_4 + p'_5} \quad (7)$$

$$p_4 = 0 \quad (8)$$

$$p_5 = 0 \quad (9)$$

$$(10)$$

これらの値 p_1, p_2, p_3, p_4, p_5 が音声入力 a の認識候補 d_1, d_2, d_3, d_4, d_5 に対する言語的出力確率であるが，d_4, d_5 に対応する値 p_4, p_5 が0であるために，候補 d_4, d_5 は削除され，よって音声入力 a の認識候補は d_1, d_2, d_3 に絞り込まれたことになる．

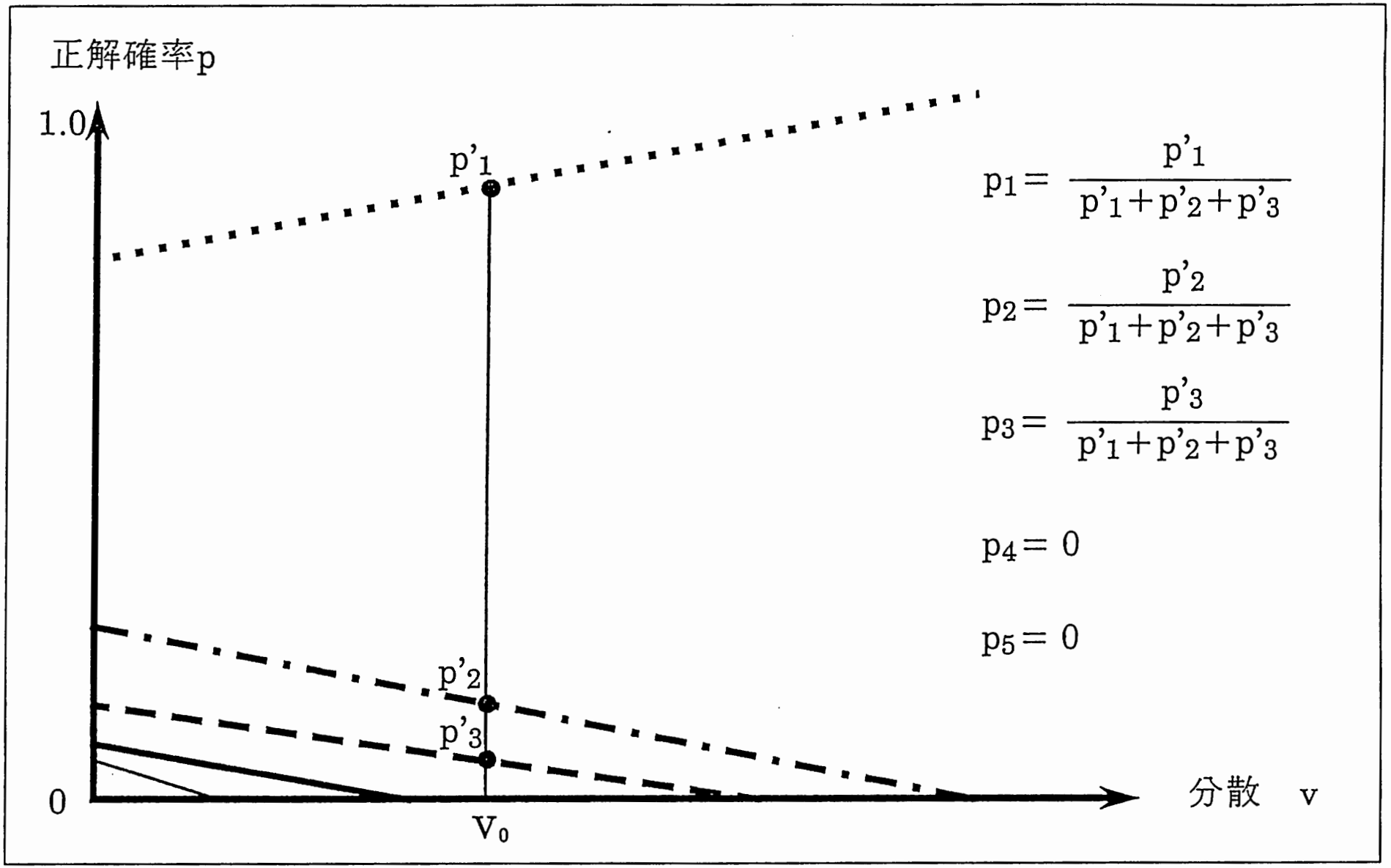

図5　正規化確率の求め方

4　実施例

図６は２階差分法と分散法による音声認識候補絞り込みのためのシステム構成を表す．簡単に説明する．

音声入力を音声認識システムで処理し，その入力に対する候補をその認識スコアと共に出力する（（１）のデータ）．
（１）のデータの２階差分値を計算し，その値が閾値以上ならば２階差分法ルーチンへ（（２））移り，候補を絞り込んだ後に分散法ルーチンにより正規化確率を求める．２階差分値が閾値に満たない場合は，そのまま分散法ルーチンで候補の絞り込みと正規化確率の計算を行なう．

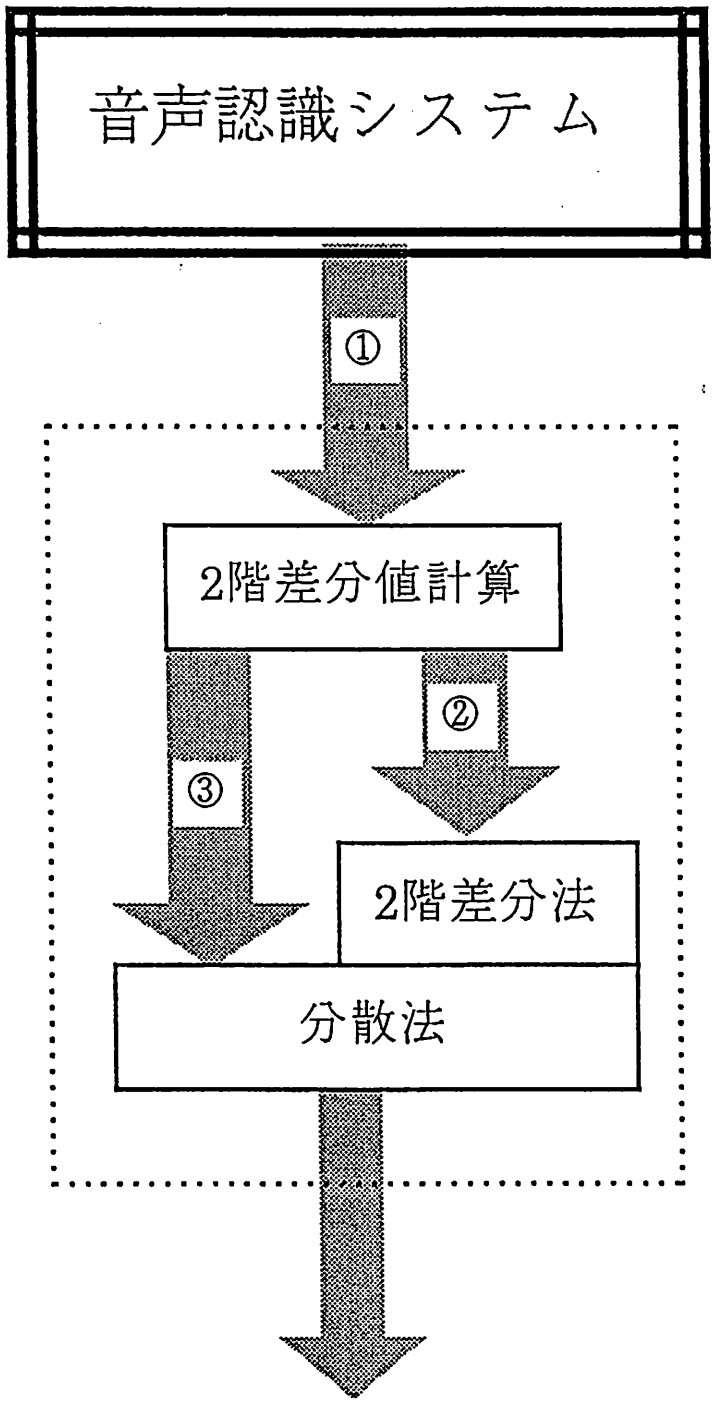

図6　システム構成

5　付録

音声言語統合化実験システムＳＬ－ＴＲＡＮＳにおいて、音声認識結果と共に出力される認識スコアは、以上述べた手法に基づいて計算されている。そのソースコードは、以下の通りである。この中で、関数 normalize がメインに呼ばれるが、引数 *func* によって絞り込みモードと単なる正規化モードを切替える。

normal.c

```
#define BORDER 0.07
#define SAFETY  0.05
#define REDUCT 8

mkrate(rate,v,func)
 double rate[], v;
 int func;
 {
 int rank;
 double min;

   rank = 0;
   min = 0.0;
   rate[0] = 19.412*v + 0.67;
     rank++;
     min = rate[0]/REDUCT;
   rate[1] = -12.941*v + 0.22;
   if((rate[1] <= 0.0) || (rate[1] == rate[0]))
       rate[1]=func*min;
   else
 rank++;
   min = rate[1]/REDUCT;

   rate[2] = -6.667*v + 0.08;
   if((rate[2] <= 0.0) || (rate[2] == rate[1]))
       rate[2]=func*min;
   else
       rank++;
```

```
  min = rate[2]/REDUCT;

  rate[3] = -5.714*v + 0.04;
  if((rate[3] <= 0.0) || (rate[3] == rate[2]))
      rate[3]=func*min;
  else
rank++;
  min = rate[3]/REDUCT;

  rate[4] = -10.0*v + 0.02;
  if((rate[4] <= 0.0) || (rate[4] == rate[3]))
      rate[4]=func*min;
  else
      rank++;

 return rank;
}

normalize(score, func)
 double score[];
 int func; /* func=0--> reduce mode  func=1--> non-reduce mode */
 {
 int  rank, c, i, inflection, safety;
 double r, v, mean, mininf, rate[5], sum, tmp;

  for(c=0; score[c]!=0 && c<5; c++);
  safety = 0;

 /* variance */
  if(c == 5)
    {
```

```
      mean = (score[0]+score[1]+score[2]+score[3]+score[4])/5;
      v = 0;
      for(i=0; i<5; i++)
        v += (score[i]-mean)*(score[i]-mean);
      v = v/5;
    }
  else
    v = -0.004875*c+0.021938;

/* rate of distribution */
  rank = mkrate(rate, v, func);
if(c == 5 )
 {
/* inflection */
  mininf = score[1]-score[0];
  inflection = 1;
  for(i=1; i<4; i++)
    if( (score[i-1]+score[i+1]-2*score[i]) >= mininf )
      {
        inflection = i+1;
mininf = score[i-1]+score[i+1]-2*score[i];
      }

  if( mininf < SAFETY )
    {
      rank = mkrate(rate, v/10000, func);
      safety = 1;
    }

/* calculation of rank */
  if( mininf >= BORDER )
```

```
    {
    v = -0.004875*inflection+0.021938;
    rank = mkrate(rate, v, func);
    }
 }
else
 rank = mkrate(rate, -0.004875*c+0.021938, 0);

/* nomarizing of rate */
  sum = rate[0]+rate[1]+rate[2]+rate[3]+rate[4];
  tmp = score[0];
  score[0] = rate[0]/sum;
  for(i=1; i<5; i++)
    if( score[i] == tmp )
      score[i] = score[i-1];
    else
     {
      tmp = score[i];
      score[i] = rate[i]/sum;
     }

  return safety;
}
```