(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년07월16일
(11) 등록번호 10-2134972
(24) 등록일자 2020년07월10일

(51) 국제특허분류(Int. Cl.)
G06F 7/544 (2017.01) *G06F 7/483* (2006.01)
G06F 7/523 (2006.01)
(52) CPC특허분류
G06F 7/544 (2018.05)
G06F 7/483 (2013.01)
(21) 출원번호 10-2018-0137216
(22) 출원일자 2018년11월09일
심사청구일자 2018년11월09일
(65) 공개번호 10-2020-0053836
(43) 공개일자 2020년05월19일
(56) 선행기술조사문헌
KR1020170105733 A
KR1020110058013 A
JP2000227962 A
KR1020180060968 A

(73) 특허권자
한국교통대학교산학협력단
충청북도 충주시 대소원면 대학로 50
(72) 발명자
임정수
충청북도 충주시 예성로 401, 403동 704호 (연수동, 주공아파트)
김훈석
충청남도 천안시 서북구 봉서산로 85, 110동 101호 (불당동, 호반리젠시빌아파트)
(74) 대리인
김종선, 이형석

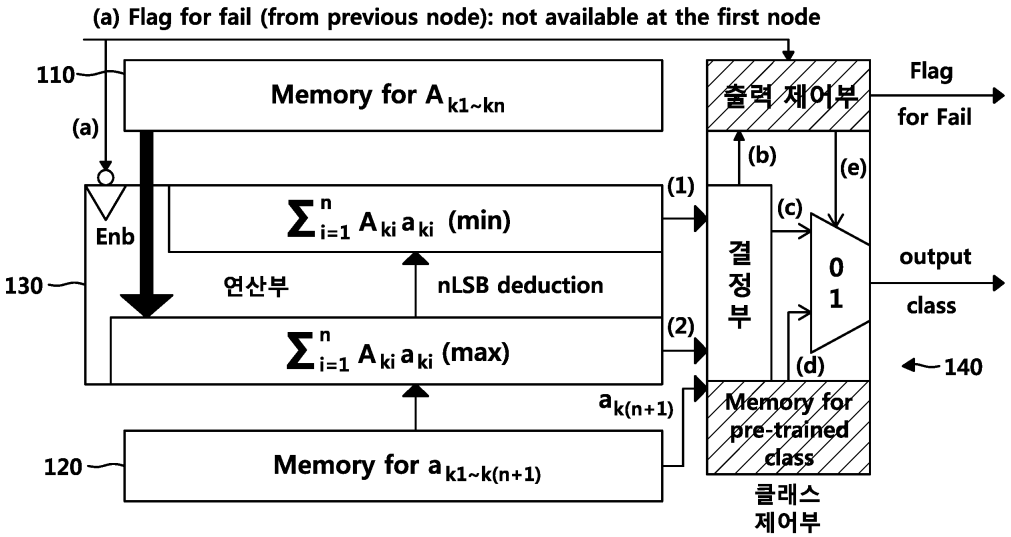
전체 청구항 수 : 총 10 항 심사관 : 지정훈

(54) 발명의 명칭 **속도와 전력효율이 향상된 결정 트리 장치 및 이의 분류 방법**

(57) 요 약

결정 트리 장치가 개시된다. 상기 결정 트리 장치는, 결정 트리로 입력되는 입력 벡터가 고정 소수점으로 근사화되고, 근사화 결과인 상기 입력벡터의 근사 값이 저장된 제1 메모리와, 각 노드에서 데이터의 분류를 위해 학습되어 결정된 속성 벡터가 고정 소수점으로 근사화 되고, 근사화 결과인 상기 속성 벡터의 근사 값이 저장된 제2 메모리와, 상기 입력 벡터의 근사 값과 상기 속성 벡터의 근사값에 대해 곱셈 연산을 수행하고, 곱셈 결과를 모두 더하여 최대 근사값과 최소 근사값을 계산하는 연산부와, 상기 최대 근사값, 상기 최소 근사값, 및 상기 각 노드에서 데이터의 분류를 위해 학습되어 결정된 바이어스의 근사 값을 이용하여 클래스 값을 결정하는 클래스 제어부를 포함한다.

대 표 도 - 도3



(52) CPC특허분류
G06F 7/523 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1345227617
부처명 교육부
연구관리전문기관 한국연구재단
연구사업명 일반연구자지원
연구과제명 모바일 클라우드 컴퓨팅에 적합한 머신러닝 기법의 통합 가속기 개발
기 여 율 1/1
주관기관 한국교통대학교
연구기간 2014.11.01 ~ 2017.04.30

명 세 서

청구범위

청구항 1

결정 트리로 입력되는 입력 벡터가 고정 소수점으로 근사화 되고, 근사화 결과인 상기 입력벡터의 근사 값이 저장된 제1 메모리;

각 노드에서 데이터의 분류를 위해 학습되어 결정된 속성 벡터가 고정 소수점으로 근사화 되고, 근사화 결과인 상기 속성 벡터의 근사 값이 저장된 제2 메모리;

상기 입력 벡터의 근사 값과 상기 속성 벡터의 근사값에 대해 곱셈 연산을 수행하고, 곱셈 결과를 모두 더하여 최대 근사값과 최소 근사값을 계산하는 연산부; 및

상기 최대 근사값, 상기 최소 근사값, 및 상기 각 노드에서 데이터의 분류를 위해 학습되어 결정된 바이어스의 근사 값을 이용하여 클래스 값을 결정하는 클래스 제어부;를 포함하는 결정 트리 장치.

청구항 2

제1항에 있어서, 상기 연산부는,

상기 입력 벡터의 근사값과 상기 속성 벡터의 근사값의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 올림 연산을 수행하고, 올림 연산 결과를 누적하여 합산한 최대 근사값을 계산하고,

상기 입력 벡터의 근사값과 상기 속성 벡터의 근사값의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 내림 연산을 수행하고, 내림 연산 결과를 누적하여 합산한 최소 근사값을 계산하는 것을 특징으로 하는 결정 트리 장치.

청구항 3

제1항에 있어서, 상기 연산부는,

상기 입력 벡터의 근사값과 상기 속성 벡터의 근사값의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 올림 연산을 수행하고, 올림 연산 결과를 누적하여 합산한 최대 근사값을 계산하고,

상기 최대 근사값, 상기 데이터의 개수, 및 소수점 아래 비트수에 기초하여 상기 최소 근사값을 계산하는 것을 특징으로 하는 결정 트리 장치.

청구항 4

제1항에 있어서, 상기 클래스 제어부는,

상기 최대 근사값과 상기 바이어스의 근사값을 비교하여 제1 비교 결과를 계산하고, 상기 최소 근사값과 상기 바이어스의 근사값을 비교하여 제2 비교 결과를 계산하고, 상기 제1 비교 결과와 상기 제2 비교 결과가 불일치하면, 해당 노드에서 연산 오류가 존재할 수 있음을 나타내는 플래그를 설정하는 것을 특징으로 하는 결정 트리 장치.

청구항 5

제4항에 있어서,

상기 플래그가 설정된 데이터는 상기 해당 노드 이후의 노드에서 연산을 수행하지 않고 상기 결정 트리가 데이

터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 클래스 값을 결정하는 것을 특징으로 하는 결정 트리 장치.

청구항 6

제5항에 있어서,

상기 확률적으로 클래스 값을 결정할 때, 학습 기간 동안 상기 해당 노드에서 더 많이 분류된 노드로 상기 데이터를 이동하는 것을 특징으로 하는 결정 트리 장치.

청구항 7

제5항에 있어서,

상기 확률적으로 클래스 값을 결정할 때, 학습 기간 동안 상기 해당 노드에서 가장 많이 결정된 클래스 값으로 상기 데이터의 클래스 값을 바로 결정하는 것을 특징으로 하는 결정 트리 장치.

청구항 8

결정 트리 장치의 분류 방법에 있어서,

결정 트리 장치가,

결정 트리로 입력되는 입력 벡터를 고정 소수점으로 근사화하여 상기 입력벡터의 근사값을 계산하는 단계;

각 노드에서 데이터의 분류를 위해 학습되어 결정된 속성 벡터를 고정 소수점으로 근사화하여 상기 속성벡터의 근사값을 계산하는 단계;

상기 입력 벡터의 근사 값과 상기 속성 벡터의 근사값에 대해 곱셈 연산을 수행하고, 곱셈 결과를 모두 더하여 최대 근사값과 최소 근사값을 계산하는 단계; 및

상기 최대 근사값, 상기 최소 근사값, 및 상기 각 노드에서 데이터의 분류를 위해 학습되어 결정된 바이어스의 근사 값을 이용하여 클래스 값을 결정하는 단계;를 포함하는 결정 트리 장치의 분류 방법.

청구항 9

제8항에 있어서,

상기 클래스 값을 결정하는 단계는,

상기 최대 근사값과 상기 바이어스의 근사값을 비교하여 제1 비교 결과를 계산하는 단계;

상기 최소 근사값과 상기 바이어스의 근사값을 비교하여 제2 비교 결과를 계산하는 단계;

상기 제1 비교 결과와 상기 제2 비교 결과가 일치하면, 비교 결과에 기초하여 상기 클래스 값을 결정하고, 상기 제1 비교 결과와 상기 제2 비교 결과가 불일치 하면, 해당 노드에서 연산 오류가 존재할 수 있음을 나타내는 플래그를 설정하고, 상기 플래그가 설정된 데이터는 상기 해당 노드 이후의 노드에서 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 상기 클래스 값을 결정하는 단계; 및

상기 연산 오류가 존재할 수 있음을 나타내는 플래그를 다음 노드로 전송하는 단계;를 포함하는 결정 트리 장치의 분류 방법.

청구항 10

제9항에 있어서,

현재 노드가 이전 노드로부터 상기 플래그를 수신한 경우,

상기 이전 노드에서 전달된 플래그가 연산 오류가 존재할 수 있음을 나타내는 플래그인 경우, 상기 현재 노드는 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 클래스 값을 결정하여 출력하고,

상기 이전 노드에서 전달된 플래그가 연산 오류가 존재하지 않음을 나타내는 플래그이고, 상기 현재 노드에서의 플래그도 연산 오류가 존재하지 않음을 나타내는 플래그인 경우, 상기 현재 노드에서 수행된 연산으로부터 결정된 클래스 값을 출력하고,

상기 이전 노드에서 전달된 플래그가 연산 오류가 존재하지 않음을 나타내는 플래그이고, 상기 현재 노드에서의 플래그가 연산 오류가 존재함을 나타내는 플래그인 경우, 상기 현재 노드에서 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 클래스 값을 결정하여 출력하는 것을 특징으로 하는 결정 트리 장치의 분류 방법.

발명의 설명

기 술 분 야

[0001] 본 발명의 개념에 따른 실시 예는 결정 트리 장치 및 이의 분류 방법에 관한 것으로, 특히 최적의 전력 효율을 가지고, 연산 속도를 향상시킬 수 있는 결정 트리 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 결정 트리(decision tree(DT))는 의사 결정 규칙과 그 결과들을 트리 구조로 도식화한 의사 결정 지원 도구의 일종이다. 결정 트리는 통계 데이터 분류에서 널리 사용된다. 결정 트리는 각 하위 영역이 특정 클래스와 관련하여 동질성이 될 때까지 반복적으로 특징 공간을 분리된 하위 영역으로 분할한다.

[0003] 도 1의 (a)는 사선(oblique) 결정 트리를 설명하기 위한 개념도이고, 도 1의 (b)는 데이터를 결정 트리 방식으로 분류해내는 실시 예를 나타내는 도면이다. 도 1을 참조하면, 사선 결정 트리는 하기의 [수학식 1]의 함수를 주어진 트리 구조에 따라 내려가면서 순차적으로 노드에서 $A_{k1\cdots kn}$와 $a_{kn\cdots k(n+1)}$를 가지고 연속적으로 곱셈, 덧셈, 비교 연산들을 수행하는 동작으로 클래스를 분류해 낸다.

[0004] [수학식 1]

[0005] $$f_k = \sum_{i=1}^{n} a_{ki} A_{ki} + a_{k(n+1)}$$

[0006] 여기서, f_k는 분류 결과를 나타내고, k는 노드 번호이고, $A_{k1\cdots kn}$는 결정 트리로 입력되는 데이터의 특성을 나타내는 벡터이고, $a_{kn\cdots k(n+1)}$는 각 노드에서의 분류를 위해 학습 기간을 통해 결정된 속성 벡터(attribute vector)를 의미한다.

[0007] 도 1의 (b)를 참조하면, 네모 상자 안의 숫자 1, 2, 3, 및 4 각각은 클래스의 종류를 나타내고, 원 안의 숫자 1, 2, 3, 4, 및 5 각각은 노드 번호를 나타낸다. 결정 트리는 각 노드에서 도 1의 (a)와 같은 연산을 행할 수 있는 전용 하드웨어 또는 범용 하드웨어를 연산에 적합하도록 하는 소프트웨어를 필요로 한다.

[0008] 그러나, 사선 결정 트리가 데이터의 클래스를 분류할 때 각 노드가 필요한 곱셈을 실행 함에 있어서, 연산에 사용되는 정수의 비트 수가 너무 많거나 정수가 아닌 실수 연산이 필요할 때 많은 하드웨어 자원(hardware resource)이 필요하게 된다. 실수 연산의 경우 고정 소수점 또는 부동 소수점 연산을 하게 되기 때문이다. 이 경우, 이진수의 소수점 아래 자리 수를 제한하거나, 이진수 전체 비트 수를 제한하여 연산하게 된다.

[0009] 그러나, 이러한 방식은 필연적으로 근사값 계산을 필요로 하고, 각 노드에서 근사값으로 연산한 곱셈 결과값을 누적하는 과정에서 오차값 누적에 의한 연산 오류가 발생하게 된다. 오차값 누적에 의한 연산 오류는 중간에 확인할 방법이 없으므로 비록 각 노드에서 결정된 방향이 잘못 결정되었다고 하더라도 결정된 방향으로 모든 데이터를 결정 트리의 맨 아래까지 내려보내는 의미 없는 연산을 계속하게 된다.

[0010] 특히, 하드웨어 자원을 많이 사용하는 곱셈 연산이 잘못 분류된 채로 계속 되는 것은 결정 트리의 속도나 전력 면에서 큰 손실이다.

선행기술문헌

특허문헌

[0011] (특허문헌 0001) 등록특허 제10-1806011호

발명의 내용

해결하려는 과제

[0012] 본 발명은 상기와 같은 문제점을 해결하기 위하여 안출된 것으로서, 본 발명의 결정 트리 장치 및 이의 분류 방법은 근사값 누적에 의한 오류가 잠재되어 있는 데이터에 플래그를 이용하여 표시하고, 이 후에 이루어지는 연산을 수행하지 않고 결정 트리가 학습 기간 중에 축적한 정보를 참조하여 상기 데이터를 확률적으로 분류하는 것을 목적으로 한다.

과제의 해결 수단

[0013] 상기와 같은 목적을 달성하기 위한 본 발명의 결정 트리 장치는, 결정 트리로 입력되는 입력 벡터가 고정 소수점으로 근사화 되고, 근사화 결과인 상기 입력벡터의 근사 값이 저장된 제1 메모리와, 각 노드에서 데이터의 분류를 위해 학습되어 결정된 속성 벡터가 고정 소수점으로 근사화 되고, 근사화 결과인 상기 속성 벡터의 근사값이 저장된 제2 메모리와, 상기 입력 벡터의 근사 값과 상기 속성 벡터의 근사값에 대해 곱셈 연산을 수행하고, 곱셈 결과를 모두 더하여 최대 근사값과 최소 근사값을 계산하는 연산부와, 상기 최대 근사값, 상기 최소 근사값, 및 상기 각 노드에서 데이터의 분류를 위해 학습되어 결정된 바이어스의 근사 값을 이용하여 클래스 값을 결정하는 클래스 제어부를 포함한다.

[0014] 상기 연산부는, 상기 입력 벡터의 근사값과 상기 속성 벡터의 근사값의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 올림 연산을 수행하고, 올림 연산 결과를 누적하여 합산한 최대 근사값을 계산하고, 상기 입력 벡터의 근사값과 상기 속성 벡터의 근사값의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 내림 연산을 수행하고, 내림 연산 결과를 누적하여 합산한 최소 근사값을 계산한다.

[0015] 상기 연산부는, 상기 입력 벡터의 근사값과 상기 속성 벡터의 근사값의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 올림 연산을 수행하고, 올림 연산 결과를 누적하여 합산한 최대 근사값을 계산하고, 상기 최대 근사값, 상기 데이터의 개수, 및 소수점 아래 비트수에 기초하여 상기 최소 근사값을 계산하거나, 그 역으로 계산할 수 있다.

[0016] 상기 클래스 제어부는, 상기 최대 근사값과 상기 바이어스의 근사값을 비교하여 제1 비교 결과를 계산하고, 상기 최소 근사값과 상기 바이어스의 근사값을 비교하여 제2 비교 결과를 계산하고, 상기 제1 비교 결과와 상기 제2 비교 결과가 불일치 하면, 해당 노드에서 연산 오류가 존재할 수 있음을 나타내는 플래그를 설정한다.

[0017] 상기 플래그가 설정된 데이터는 상기 해당 노드 이후의 노드에서 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 클래스 값을 결정한다.

[0018] 상기 확률적으로 클래스 값을 결정할 때, 학습 기간 동안 상기 해당 노드에서 더 많이 분류된 노드로 상기 데이터를 이동하거나, 학습 기간 동안 상기 해당 노드에서 가장 많이 결정된 클래스 값으로 상기 데이터의 클래스 값을 바로 결정하는 등의 방법을 사용할 수 있다.

[0019] 상기와 같은 목적을 달성하기 위한 본 발명의 결정 트리 장치의 분류 방법은, 결정 트리로 입력되는 입력 벡터를 고정 소수점으로 근사화하여 상기 입력벡터의 근사값을 계산하는 단계와, 각 노드에서 데이터의 분류를 위해 학습되어 결정된 속성 벡터를 고정 소수점으로 근사화하여 상기 속성벡터의 근사값을 계산하는 단계와, 상기 입력 벡터의 근사 값과 상기 속성 벡터의 근사값에 대해 곱셈 연산을 수행하고, 곱셈 결과를 모두 더하여 최대 근사값과 최소 근사값을 계산하는 단계와, 상기 최대 근사값, 상기 최소 근사값, 및 상기 각 노드에서 데이터의 분류를 위해 학습되어 결정된 바이어스의 근사 값을 이용하여 클래스 값을 결정하는 단계를 포함한다.

[0020] 상기 클래스 값을 결정하는 단계는, 상기 최대 근사값과 상기 바이어스의 근사값을 비교하여 제1 비교 결과를 계산하는 단계와, 상기 최소 근사값과 상기 바이어스의 근사값을 비교하여 제2 비교 결과를 계산하는 단계와, 상기 제1 비교 결과와 상기 제2 비교 결과가 일치하면, 비교 결과에 기초하여 상기 클래스 값을 결정하고, 상기 제1 비교 결과와 상기 제2 비교 결과가 불일치 하면, 해당 노드에서 연산 오류가 존재할 수 있음을 나타내는 플래그를 설정하고, 상기 플래그가 설정된 데이터는 상기 해당 노드 이후의 노드에서 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 상기 클래스 값을 결정하는 단계와, 상기 결정된 클래스 값과 상기 연산 오류가 존재할 수 있음을 나타내는 플래그를 다음 노드로 전송하는 단계를 포함한다.

[0021] 현재 노드가 이전 노드로부터 상기 플래그를 수신한 경우, 상기 이전 노드에서 전달된 플래그가 연산 오류가 존재할 수 있음을 나타내는 플래그인 경우, 상기 현재 노드는 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 클래스 값을 결정하여 출력하고, 상기 이전 노드에서 전달된 플래그가 연산 오류가 존재하지 않음을 나타내는 플래그이고, 상기 현재 노드에서의 플래그도 연산 오류가 존재하지 않음을 나타내는 플래그인 경우, 상기 현재 노드에서 수행된 연산으로부터 결정된 클래스 값을 출력하고, 상기 이전 노드에서 전달된 플래그가 연산 오류가 존재하지 않음을 나타내는 플래그이고, 상기 현재 노드에서의 플래그가 연산 오류가 존재함을 나타내는 플래그인 경우, 상기 현재 노드에서 연산을 수행하지 않고 상기 결정 트리가 데이터 분류를 위해 학습하는 과정에서 축적된 정보를 이용하여 확률적으로 클래스 값을 결정하여 출력한다.

발명의 효과

[0022] 상기한 바와 같은 본 발명의 결정 트리 장치 및 방법은 연산 시 오류가 잠재되어 있는 데이터에 대해서 이 후에 이루어지는 연산을 건너 뛰므로 전력 효율을 높일 수 있고, 속도를 향상시킬 수 있는 효과가 있다.

[0023] 또한, 플래그 값을 최종 노드에 전달함으로써 사용자에게 오류가 잠재되어 있는 데이터라는 것을 알릴 수 있는 효과가 있다.

도면의 간단한 설명

[0024] 도 1의 (a)는 사선(oblique) 결정 트리를 설명하기 위한 개념도이고, 도 1의 (b)는 데이터를 결정 트리 방식으로 분류해내는 실시 예를 나타내는 도면이다.

도 2는 본 발명의 실시 예에 따른 분류 연산을 구현하기 위한 결정 트리 내의 노드 내의 구조를 나타내는 블록도이다.

도 3은 본 발명의 다른 실시 예에 따른 결정 트리 내의 노드에서의 분류 연산을 구현하기 위한 블록도이다.

도 4는 본 발명의 실시 예에 따른 확률을 이용한 데이터 분류 방법을 설명하기 위한 개념도이다.

발명을 실시하기 위한 구체적인 내용

[0025] 이하에서는 본 발명에 따른 실시 예 및 도면을 참조하여, 본 발명을 더욱 상술한다.

[0026] 도 2는 본 발명의 실시 예에 따른 분류 연산을 구현하기 위한 결정 트리 내의 노드 내의 구조를 나타내는 블록도이다. 도 2를 참조하면, 본 발명의 실시 예에 따른 결정 트리(100)는 제1 메모리(110), 제2 메모리(120), 연산부(130), 및 클래스 제어부(140)를 포함한다.

[0027] 제1 메모리(110)에는 결정 트리로 입력되는 입력 벡터가 저장된다. 예컨대, 제1 메모리(110)는 i가 1 내지 n(n은 자연수)의 범위를 가지고, 현재 노드가 k-번째일 때, 입력 벡터의 요소인 $A_{k1\sim kn}$을 저장할 수 있다. 이 때, 제1 메모리(110)에 저장되는 입력 벡터는 이미 고정 소수점으로 근사화 되어있는 값이다. 예컨대, 입력 벡터는 원래 값을 반올림하여 구해질 수 있으나 이에 한정되는 것은 아니다.

[0028] 제2 메모리(120)에는 각 노드에서 데이터의 분류를 위해 학습되어 저장된 속성 벡터(attribute vector)가 저장된다. 예컨대, 제2 메모리(120)는 i가 1 내지 n+1의 범위를 가지고, 현재 노드가 k-번째일 때, 속성 벡터의 요소인 $a_{k1\sim k(n+1)}$를 저장할 수 있다. 이 때, 제2 메모리(120)에 저장되는 속성 벡터는 이미 고정 소수점으로 근사화 되어있는 값이다. 예컨대, 속성 벡터는 원래 값을 반올림하여 구해질 수 있으나 이에 한정되는 것은 아니

다.

[0029] 연산부(130)는 하기의 [수학식 2]와 [수학식 3]과 같은 연산을 수행한다.

[0030] [수학식 2]

[0031] $$\sum_{i=1}^{n} A_{ki} a_{ki}(min)$$

[0032] [수학식 3]

[0033] $$\sum_{i=1}^{n} A_{ki} a_{ki}(max)$$

[0034] 연산부(130)는 [수학식 2]를 이용하여 최소 근사값을 구하고, [수학식 3]을 이용하여 최대 근사값을 구한다. 연산부(130)는 최대 근사값을 구하기 위해 이미 근사화된 각 입력 벡터와 각 속성 벡터의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 올림 연산을 수행하고, 올림 연산 결과들을 모두 더하여 최대 근사값을 구한다.

[0035] 연산부(130)는 최소 근사값을 구하기 위해 이미 근사화된 각 입력 벡터와 각 속성 벡터의 곱셈 연산을 수행하고, 곱셈 연산 결과에 대해 내림 연산을 수행하고, 내림 연산 결과들을 모두 더하여 최소 근사값을 구한다.

[0036] 예컨대, 제1 메모리(110)에는 A_{11} = 000110.101, A_{12} = 000011.100 가 저장되고, 제2 메모리(120)에는 a_{11}= 000010.110, a_{12}= 000100.011 가 저장되어 있다고 가정하자. 즉, 이미 고정 소수점으로 근사화 되어 있는 값으로 원래 값을 반올림하여 구해진 것으로 가정한다.

[0037] $A_{11}*a_{11}$ = 010010.001110 이다. $A_{11}*a_{11}$를 소수점 셋째자리로 올림하면 010010.010 이고, $A_{11}*a_{11}$를 소수점 넷째자리로 내림하면 010010.001 이다.

[0038] 또한, $A_{12}*a_{12}$ = 001111.0101 이다. $A_{12}*a_{12}$를 소수점 셋째자리로 올림하면 001111.011 이고, $A_{12}*a_{12}$를 소수점 셋째자리로 내림하면 001111.010 이다.

[0039] 상기 계산 결과를 기초로 올림한 결과들을 모두 더하면 다음과 같다.

[0040] $\sum_{i=1}^{2} A_{1i} a_{1i}(max)$ = 010010.010 + 001111.011 = 100001.101

[0041] 상기 계산 결과를 기초로 내림한 결과들을 모두 더하면 다음과 같다.

[0042] $\sum_{i=1}^{2} A_{1i} a_{1i}(min)$ = 010010.001 + 001111.010 = 100001.011

[0044] 클래스 제어부(140)는 연산부(130)의 연산 결과에 따라 클래스 출력값을 분류 결과로써 출력한다.

[0045] $\sum_{i=1}^{n} A_{ki} a_{ki}$ 연산을 수행할 때 하드웨어 자원의 제약에 따라 제한된 비트 수에 근거하여 LSB(least significant bit)가 결정되고, 그 아래의 숫자는 버리거나, 올림하거나, 반올림할 수 있다.

[0046] 이 때, 원래 연산 결과가 양수의 경우 올림한 값, 원래 연산 결과가 음수의 경우 내림한 값 즉, 원래 연산 결과보다 가장 근사하게 큰 근사값을 $A_{ki} a_{ki}(max)$이라 하고, 원래 연산 결과가 양수의 경우 내림한 값, 원래 연산 결과가 음수의 경우 올림한 값 즉, 원래 연산 결과보다 가장 근사하게 작은 근사값을 $A_{ki} a_{ki}(min)$이라 하면, LSB 이하의 숫자가 전혀 없는 경우에만 두 값이 일치하고, 그렇지 않은 경우 LSB 1비트 만큼의 차이가 발생하게 된다.

[0047] 이 값들을 클래스 분류하기 위해 모두 더하게 되면, 최종적으로 $\sum_{i=1}^{n} A_{ki} a_{ki}(min)$ 와 $\sum_{i=1}^{n} A_{ki} a_{ki}(max)$ 사이에는 근사값을 전혀 사용하지 않은 경우에서 근사값을 매번 사용하게 된 경우까지 0 내지 n*LSB 만큼 차이가 나게 된다. 상기의 예에서, n=2(= 이진수 10)이고, 소수점 아래 자리수가 3 이므로 상기 LSB는 0.001이다. 검증을 위하여, 내림한 결과들을 모두 더한 값 "100001.011"에 "10 * 0.001"을 더하면 100001.011 + 0.010 = 100001.101 가 된다. 즉, 내림한 결과들을 모두 더한 값에 n*LSB를 더하면 올림한 결과들을 모두 더한 값 "100001.101"과 일치하게 된다.

[0048] 따라서, 클래스 제어부(140)는 $\sum_{i=1}^{n} A_{ki}a_{ki}(min)$ 와 $\sum_{i=1}^{n} A_{ki}a_{ki}(max)$ 각각의 결과를 $a_{k(n+1)}$과 비교하고, 비교의 결과에 따라 클래스 출력값 값을 결정한다. $a_{k(n+1)}$ 는 바이어스를 나타내는 매개변수를 의미하고, 결정 트리가 학습하는 과정에서 속성 벡터와 함께 수집되는 정보이다. $a_{k(n+1)}$ 는 제2 메모리에 저장된 n+1번째 속성 벡터의 값이다.

[0049] 예컨대, 두 결과가 모두 $-a_{k(n+1)}$ 보다 작거나 같으면 각 노드에서 좌측 또는 우측으로 보내고, 두 결과가 모두 $-a_{k(n+1)}$ 보다 크면 각 노드에서 두 결과가 모두 $a_{k(n+1)}$ 보다 작거나 같을 때의 방향과 반대 방향으로 보낸다.

[0050] 만약, 두 결과에 차이가 나면, 근사 방식에 따라 방향이 달라질 수 있는 경우가 된다. 따라서, 클래스 제어부(140)는 해당 데이터에 근사값 누적에 의한 오류가 잠재되어 있는 경우로 인지하여 상기 데이터의 flag for fail 값을 1로 설정하고, 결정 트리의 다음 노드로 flag for fail 값을 전달한다. 이 후에 이루어지는 $\sum_{i=1}^{n} A_{ki}a_{ki} + a_{k(n+1)}$ 의 연산은 앞 단에서 잘못 결정된 데이터에 대해서 필요 없는 연산을 지속하게 될 가능성을 갖게 된다.

[0051] 따라서, 이 후의 노드에서의 클래스 결정 방향은 $\sum_{i=1}^{n} A_{ki}a_{ki} + a_{k(n+1)}$ 연산을 건너 뛰고, 결정 트리가 학습 기간 중에 축적한 정보를 참조하여 확률적으로 결정한다.

[0052] 결정 트리는 학습 과정에서 샘플 데이터로 결정 트리를 시뮬레이션하여 결정 트리의 속성 벡터 $a_{kn\cdots k(n+1)}$ 를 정하고, 정확도와 구조를 확인하게 된다. 따라서, 각 노드에서 어떤 방향으로 몇 개의 데이터가 흘러가는지 또는 데이터가 어떤 클래스로 많이 분류되었는지에 대한 정보 수집이 가능하다.

[0053] 따라서, flag for fail 값을 이 후의 단계로 계속 보내어 flag for fail 값이 1로 표시된 데이터에 대해서는 학습 기간 중에 축적된 정보를 이용하여 분류할 수 있다. 또한, flag for fail 값은 최종 노드에서 최종 클래스 출력값과 함께 전달되어, 해당 데이터가 오류 가능성이 있고, 확률적으로 분류된 데이터임을 사용자가 구분할 수 있도록 한다.

[0055] 도 3은 본 발명의 다른 실시 예에 따른 결정 트리 내의 노드에서의 분류 연산을 구현하기 위한 블록도이다. 도 2의 실시 예와 다른 점은 연산부(130)가 최소 근사값을 구할 때, 내림 연산을 수행하지 않고 하기의 [수학식 4]와 같이 올림 연산 결과들을 이용하여 최소 근사값을 구할 수 있다.

[0056] [수학식 4]

[0057] $\sum_{i=1}^{n} A_{ki}a_{ki}(min) = \sum_{i=1}^{n} A_{ki}a_{ki}(max)$ - n*LSB

[0058] 즉, 연산부(130)는 최대 근사값과 최소 근사값 양쪽을 계산하는 것이 아니라 한 쪽만 계산하고, 다른 쪽 값은 누적 연산 없이 차이 값인 0 내지 n*LSB 를 더하거나 빼는 보정을 통해서 결정할 수 있다. 따라서, 연산부(130)는 하드웨어 자원을 절약할 수 있다.

[0059] 예컨대, 도 2의 실시 예에서, $\sum_{i=1}^{2} A_{1i}a_{1i}(max)$ = 100001.101 이고, n = 10 (이진수)이고, LSB = 0.001 이므로, 이를 [수학식 4]에 대입하면 하기와 같이 최소 근사값이 동일하게 얻어진다.

[0060] $\sum_{i=1}^{2} A_{1i}a_{1i}(min)$ = 100001.101 - (10*0.001)

[0061] = 100001.101 - 0.010 = 100001.011 .

[0063] 도 3을 구체적으로 설명하면 다음과 같다. 신호 (a)는 이전 노드에서 전달된 flag for fail 값이고, 만약 flag for fail 값이 '1'이면 연산부(130)는 비활성화(disable)되어 연산이 생략되고, flag for fail 값이 '0'이면 연산부(130)는 활성화(able)되어 연산을 수행한다.

[0064] 신호 (b)는 신호 (1)과 신호 (2)를 비교하여, 두 신호의 부호가 다르면 '1'이 되고, 두 신호의 부호가 같으면

'0'이 된다. 즉, 신호 (b)는 현재 노드에서의 flag for fail 값이 된다.

[0065] 신호 (c)는 현재 노드에서 수행된 연산으로부터 구해진 클래스 값이고, 신호 (d)는 현재 노드에서 미리 학습되어 저장된 클래스 값이다.

[0066] 클래스 제어부(140)는 선택 신호 (e)에 응답하여, 이전 노드에서 전달된 flag for fail 값(a)이 '1'이면, 현재 노드에 미리 학습되어 저장되어 있는 클래스 값(d)을 출력하고, 이전 노드에서 전달된 flag for fail 값(a)이 '0'이고, 현재 노드에서의 flag for fail 값이 '0'이면, 현재 노드에서 수행된 연산으로부터 구해진 클래스 값(C)을 출력하고, 이전 노드에서 전달된 flag for fail 값(a)이 '0'이고, 현재 노드에서의 flag for fail 값이 '1'이면, 현재 노드에서 미리 학습되어 저장된 클래스 값(d)을 출력한다.

[0067] 다시 말하면, 현재 노드에서의 flag for fail 값이 '1'이면 입력 벡터의 클래스 값은 미리 학습된 클래스 값으로 출력된다. 미리 학습된 클래스 값은 연산에 의한 예측 값이 아니라 미리 오프라인에서 학습되어 저장되어 있는 예측 값이다.

[0069] 도 2와 도 3에서 설명한 바와 같이, 결정 트리는 곱셈, 덧셈, 및 비교기로 이루어지는데, 가장 많은 하드웨어 자원을 요구하면서 가장 많은 지연을 차지하는 부분이 곱셈 연산이다. 따라서, 곱셈 연산을 할 때 하드웨어 자원의 허용치에 따라 이진수의 비트수를 제한하여 연산한다.

[0070] 본 발명은 결정 트리의 각 노드에서 근사값의 곱셈의 결과를 모두 올림하여 더한 최대 근사값과, 근사값의 곱셈의 결과를 모두 내림하여 더한 최소 근사값을 이용하여 연산을 수행하되, 연산 시 근사값 누적에 의한 오류가 잠재되어 있는 데이터를 인식하고, 이 후 노드에서는 원래의 연산을 수행하지 않고 결정 트리가 학습 기간 중에 축적한 정보에 따라 확률적으로 결과를 도출한다. 즉, 원래 노드에서 수행되는 연산을 건너뜀으로써 건너뛴 연산에 해당하는 만큼 결정 트리의 분류 속도가 향상되는 효과를 볼 수 있다.

[0071] 또한, 근사화로 인한 오류의 존재 가능성을 검출하기 위해 추가된 덧셈기와 비교기에서 소비되는 전력보다 건너뛴 곱셈 연산에 해당하는 만큼 절약된 전력이 현저히 많을 것으로 예상되며, 절약되는 전력은 연산이 생략된 노드의 개수가 많을수록 증가한다.

[0073] 도 4는 본 발명의 실시 예에 따른 확률을 이용한 데이터 분류 방법을 설명하기 위한 개념도이다. 도 4를 참조하면, 결정 트리가 학습 기간 중에 축적한 정보를 이용하여 확률적으로 데이터를 분류하는 방법으로 이용할 수 있다.

[0074] 상자 안의 숫자 1, 2, 3, 및 4 각각은 클래스의 종류를 나타내고, 괄호 안의 숫자 5, 40, 55, 60, 100, 140, 160, 200, 300, 및 500 각각은 데이터의 개수를 나타내고, 동그라미 안의 숫자 1, 2, 3, 4, 및 5 각각은 노드 번호를 나타낸다. 이러한 정보를 통해서 결정 트리가 확률적으로 데이터를 분류하는 방법은 다음과 같다.

[0075] 1. 데이터가 학습 기간 동안 더 많이 분류된 쪽으로 결정

[0076] 만약, flag for fail 값이 1이면, 노드 ①에서 500개의 데이터 중에서 300개의 데이터가 좌측으로 가고 200개의 데이터가 우측으로 갔으므로 좌측으로 간 데이터가 더 많기 때문에 해당 데이터에 대해서는 좌측으로 보내고, 노드 ②에서는 좌측으로 간 데이터가 더 많이 때문에 계산 없이 클래스 출력 값을 1로 결정한다.

[0077] 또는, flag for fail 값이 1이면, 노드 ①에서는 첫번째 연산이므로 연산을 피할 수 없기 때문에 $\sum_{i=1}^{n} A_{ki}a_{ki}(\max) + a_{k(n+1)}$ 와 $\sum_{i=1}^{n} A_{ki}a_{ki}(\min) + a_{k(n+1)}$ 의 결과가 0에서 더 먼쪽으로 결정할 수 있다. 만약 노드 ①에서 우측으로 결정했다면 노드 ③에서는 좌우로 결정된 경우의 수가 같다. 이 때 같은 경우의 수에 대해 무조건 좌측으로 보내도록 하거나 가장 짧은 경로를 택하도록 한다면 클래스 출력 값은 4로 결정된다. 반대로 같은 경우의 수에 대해 무조건 우측으로 보내도록 하면 노드 ④에서는 우측으로 가고 노드 ⑤에서는 더 많이 분류된 쪽인 좌측으로 가게 되어 클래스 출력값을 2로 결정하게 된다.

[0079] 2. 데이터가 학습 기간 동안 해당 노드에서 가장 많이 분류된 클래스로 바로 결정

[0080] 노드 ①에서 flag for fail 값이 1이면, 노드 ①에서는 2로 간 경우가 많기 때문에 클래스 출력값이 2로 결정되고, 노드 ③에서는 4로 간 경우가 많기 때문에 클래스 출력값이 4로 결정된다.

[0082] 결정 트리는 확률적으로 계산한 데이터의 클래스 값과 함께 flag for fail 값을 함께 표시함으로써 완전하게 계산이 이루어진 데이터와 그렇지 않은 데이터를 사용자가 알 수 있게 한다. 따라서, 사용자는 클래스 분류의 신

뢰도를 의심하는데 참조할 수 있다.

[0085] 본 발명은 도면에 도시된 일 실시 예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시 예가 가능하다는 점을 이해할 것이다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

부호의 설명

[0086] 100; 결정 트리

110; 제1 메모리

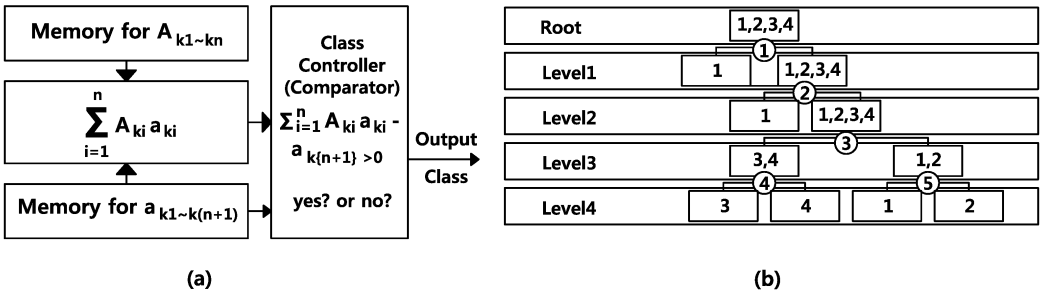
120; 제2 메모리

130; 연산부

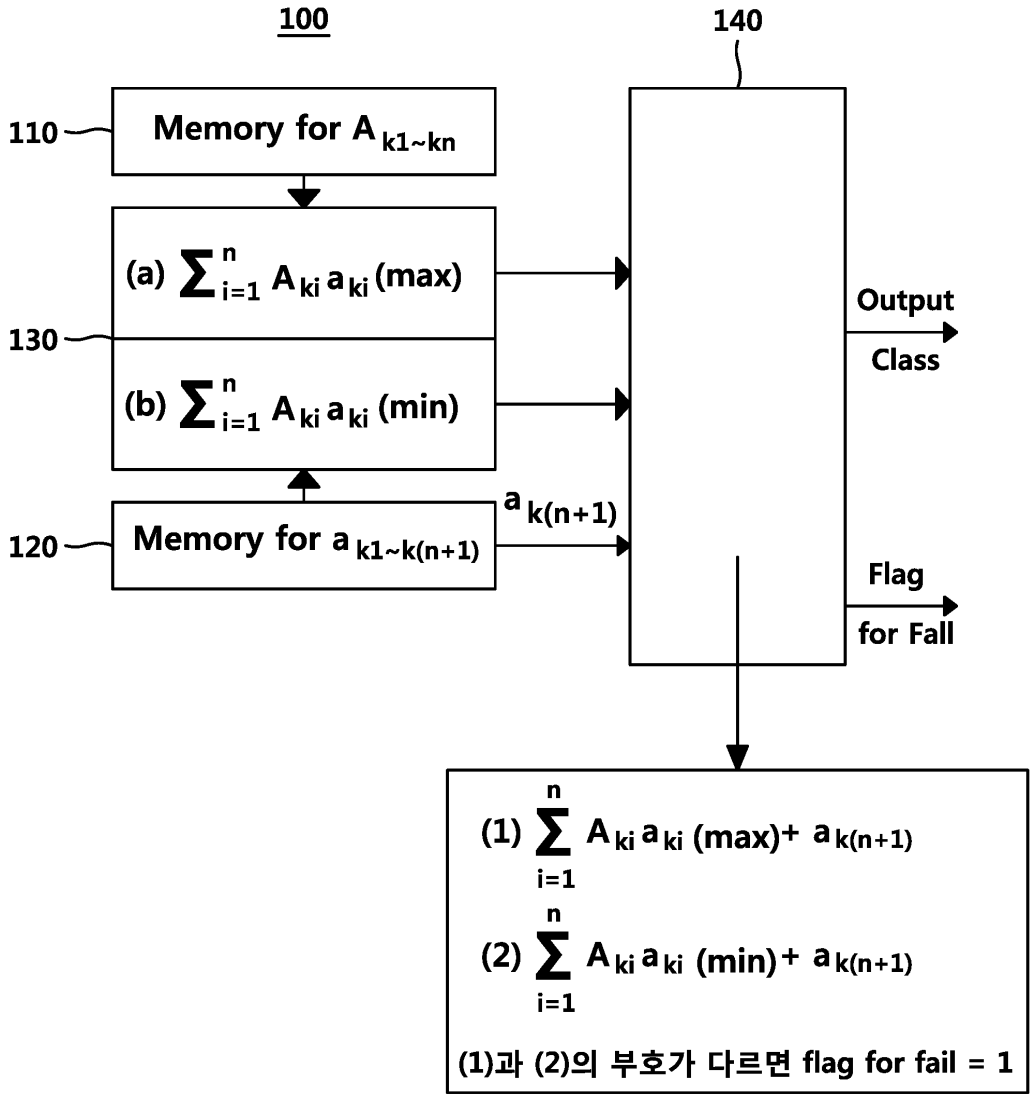
140; 클래스 제어부

도면

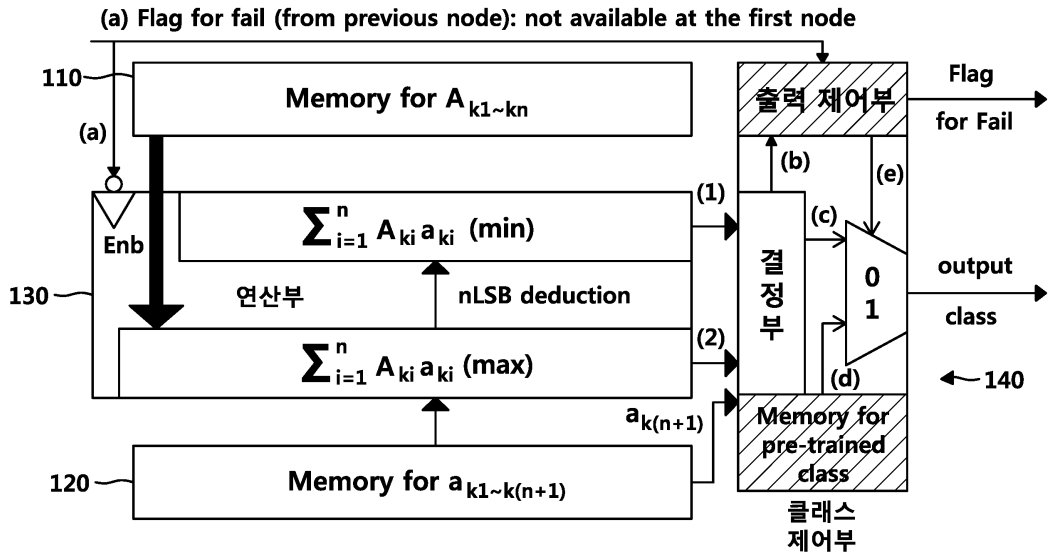
도면1



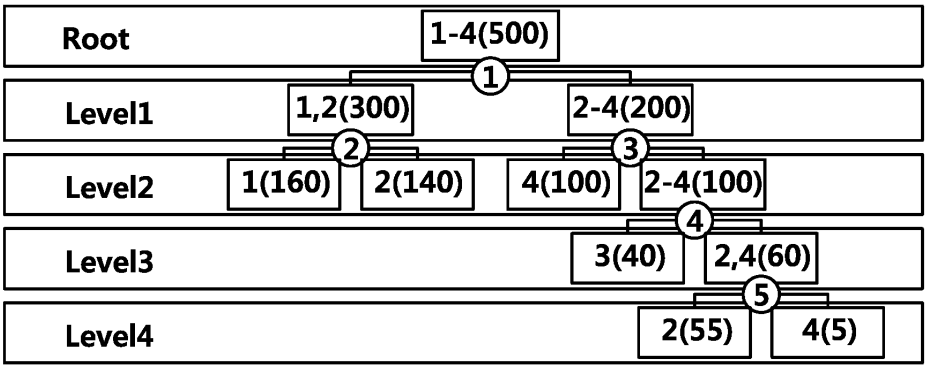
도면2


100
140
110
Memory for $A_{k1\sim kn}$
(a) $\sum_{i=1}^{n} A_{ki} a_{ki}$ (max)
130
(b) $\sum_{i=1}^{n} A_{ki} a_{ki}$ (min)
120
Memory for $a_{k1\sim k(n+1)}$
$a_{k(n+1)}$
Output
Class
Flag
for Fall
(1) $\sum_{i=1}^{n} A_{ki} a_{ki}$ (max)+ $a_{k(n+1)}$
(2) $\sum_{i=1}^{n} A_{ki} a_{ki}$ (min)+ $a_{k(n+1)}$
(1)과 (2)의 부호가 다르면 flag for fail = 1

도면3


(a) Flag for fail (from previous node): not available at the first node
110
Memory for $A_{k1\sim kn}$
(a)
Enb
130
$\sum_{i=1}^{n} A_{ki} a_{ki}$ (min)
연산부
nLSB deduction
$\sum_{i=1}^{n} A_{ki} a_{ki}$ (max)
120
Memory for $a_{k1\sim k(n+1)}$
$a_{k(n+1)}$
출력 제어부
Flag
for Fail
(b)
(e)
(1)
(c)
결정부
0
1
output
class
(2)
(d)
140
Memory for pre-trained class
클래스 제어부

도면4


Root
1-4(500)
1
Level1
1,2(300)
2-4(200)
2
3
Level2
1(160)
2(140)
4(100)
2-4(100)
4
Level3
3(40)
2,4(60)
5
Level4
2(55)
4(5)