지지 벡터 머신을 이용한 다변수 결정 트리

강선구 이병우° 나용찬 조현성 윤철민 양지훈
서강대학교 컴퓨터학과 데이터마이닝 연구실
sun9sun9@hotmail.com, elva1212@gmail.com°, ycna@sogang.ac.kr,
raptor21c@gmail.com, cmyun@mllag.sogang.ac.kr, yangjh@sogang.ac.kr

A Multivariate Decision Tree using Support Vector Machines

Sungu Kang B.W. Lee° Y.C. Na H.S. Jo C.M. Yoon Jihoon Yang
Computer Science, Sogang Univ. DataMining Lab.

요 약

결정 트리는 큰 가설 공간을 가지고 있어 유연하고 강인한 성능을 지닐 수 있다. 하지만 결정트리가 학습 데이터에 지나치게 적응되는 경향이 있다. 학습데이터에 과도하게 적응되는 경향을 없애기 위해 몇몇 가지치기 알고리즘이 개발되었다. 하지만, 데이터가 속성 축에 평행하지 않아서 오는 공간 낭비의 문제는 이러한 방법으로 해결할 수 없다. 따라서 본 논문에서는 다변수 노드를 사용한 선형 분류기를 이용하여 이러한 문제점을 해결하는 방법을 제시하였으며, 결정트리의 성능을 높이고자 지지 벡터 머신을 도입하였다(SVMDT).

본 논문에서 제시한 알고리즘은 세 가지 부분으로 이루어졌다. 첫째로, 각 노드에서 사용할 속성을 선택하는 부분과 둘째로, ID3를 이 목적에 맞게 바꾼 알고리즘과 마지막으로 기본적인 형태의 가지치기 알고리즘을 개발하였다. UCI 데이터 셋을 이용하여 OC1, C4.5, SVM과 비교한 결과, SVMDT는 개선된 결과를 보였다.

1. 서 론

본 연구에서는 기계 학습 분야에서 오랫동안 연구 되어 왔던 결정 트리 알고리즘과 마찬가지로 오래 전부터 연구되어 온 퍼셉트론이론, 커널 이론 그리고 최적화 이론[1]에 기반을 둔 지지 벡터 머신(Support vector machine)을 이용하여 이 알고리즘을 결합한 지지 벡터 머신 트리(SVMDT, Support vector machine decision tree)에 대하여 설명하겠다. 또한 UCI 데이터 셋을 통한 검증을 하고, 지지 벡터 머신 트리의 앞으로의 가능성을 논하고자 한다.

2. 지지 벡터 머신을 이용한 다변수 결정 트리

본 장에서는 지지 벡터 머신과 결정 트리의 장점을 결합한 지지 벡터 머신 다변수 결정트리의 알고리즘을 설명할 것이다. 지지 벡터 머신을 사용하여 다변수 결정 트리의 이름을 지지 벡터 트리 (SVMDT, Support vector decision tree)라고 이후로 표기하겠다.

2.1 지지 벡터 머신 다변수 결정 트리 알고리즘

다변수 결정 트리는 트리의 크기를 줄여서 일반적인 성능을 향상 시키는데 용이하다[2]. 또한, 지지 벡터 머신의 강점 중의 하나로써 선형으로 분리가 가능하지 않는 데이터에 대해서 커널 함수를 사용하여 분리가 가능하도록 한다.

※ 본 연구는 한국과학재단 특정기초연구 (R01-2004-000-10689-0) 및 교육부 두뇌 한국(BK21) 연구 지원으로 수행되었음.

2.2 지지 벡터 트리(SVMDT) 알고리즘

SVMDT알고리즘은 다변수 노드와 다변수로 쓰일 속성 선택 과정, 노드가 구성되면 결정 트리 형태로 구성하는 과정, 모든 트리를 만들고 나서 과잉 적응을 완화하기 위한 가지치기 과정으로 크게 3가지 과정으로 이루어져 있다.

2.2.1. 속성 선택 알고리즘

정보 획득을 이용할 경우 불필요한 노드의 수가 많아질 수 있으므로 데이터의 수의 균등 분배가 되었는지 측정하는 분할정보(Split information)를 측정한다[3]. 분할정보가 작을수록 데이터의 자식 노드로 균등하게 나누어졌다는 것을 의미한다.

연속 변수에 대한 구분 지점을 정하는 데 있어서도 바로 이 획득율을 사용하였다. 언덕오름(Hill-climbing) 알고리즘을 사용하여, 절단 정보가 가장 큰 지점을 구분 지점으로 지정했다.

$$SplitInformation(S,A) \equiv -\sum_{i=1}^{c}\frac{|S_i|}{S}\log_2\frac{|S_i|}{S}$$

$$GainRatio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

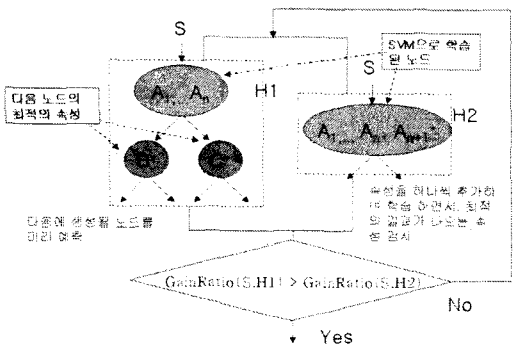
, c는 클래스의 수, S는 속성 집합 A에 의해 나누어진 S의 부분 집합들

단일 변수를 선택한 다음에는 다변수를 사용할 것인지를 정해야 한다. 이 경우에도 획득비율을 사용하여 결정할 수 있다. 이 알고리즘의 기본적인 형태는 순차전방선택법(Sequential foward feature selection)을 띄고 있다. 순차전방선택법은 선택 속성 집합에서 최적이라 예상되는 속성을 하나씩 추가하는 방법이다. [표 1]은 획득율을 사용해서 다변수항의 속성을 결정하는 SVMDT에서 위해 고안한 알고리즘의 Pseudo 코드이다.

[표 1] 다변수항 속성 결정 알고리즘

1. $a^* = \arg\max_{a \in A} GainRatio(S, a)$,
   $g = GainRatio(S, a^*)$, $D \leftarrow a^*$
2. S_+ is the subset of A that is classified positive class using D
   S_- is the subset of A that is classified negative class using D
3. $b = \arg\max_{a \in A} GainRatio(S_+, a)$,
   $g_b = GainRatio(S, b) \times \frac{|S_+|}{S}$
   $c = \arg\max_{a \in A} GainRatio(S_-, a)$,
   $g_c = GainRatio(S, c) \times \frac{|S_-|}{|S|}$
4. $d^* = \arg\max_{d \in A-D} GainRatio(S, SVM(D \cup d))$
   $g_d = GainRatio(S, SVM(D \cup d^*))$
5. if ($g + g_b + g_c < g_d$) THEN $g = g_d$ $D \leftarrow b$ goto 2

그리고, 여기서 단일 노드의 속성 선택 방식은 순위 기반(Rank based) 방식이다. 이 방식은 최적의 속성이라고 보장은 할 수 없다. 하지만 다음 노드를 구성하는 유력한 속성을 순위 기반으로 구분한다는 점에서, 이 방식은 고려할만한 휴리스틱(Heuristic)이라 할 수 있다. [그림 1]은 지지벡터 머신의 속성 선택 과정을 도식화한 것이다.



[그림 1] 노드에서 사용될 SVM속성 선택과정

2.2.2 다변수 결정 트리의 생성 알고리즘

이 알고리즘은 위에서 언급한 속성 선택 과정과 ID3 알고리즘을 결합한 것이다. ID3는 하나의 속성에서 가지(Branch)를 생성하지만 여기서는 양방향(Two-way)의 가지만을 고려한다. 또한 ID3는 명목 속성만을 취급하지만, 여기서는 연속 속성만을 취급한다. [표 2]는 다변수 결정 트리 생성 알고리즘의 Pseudo 코드이다.

[표 2] 다변수 결정 트리 생성 알고리즘

```
SVMDT(Examples, Attributes)
BEGIN
   Create a root node
   IF (Entropy(Examples) = 0)
      Attach a terminal node with the corresponding label.
   Calculate Split points and GainRatio of attributes.
   A = Select the best attributes of the attributes.
   S = split point of the best attributes.
   Examples- are the negative points of the test(Example_A < S)
   Examples+ are the positive points of the test(Example_A ≥ S)
   SelectedSet ← A
   WHILE(TRUE)
   BEGIN
      Calculate Split points and GainRatio of Examples-
       and Examples+
      NextGainRation is the best GainRatio of Examples-
       and Examples+
      SVMGainRatio is the Maximum GainRatio of SVM
       using FeatureSet ∪ {a|a∈Attribute∩SelectedSet^c},
       and BESTSVM is the corresponding SVM and A_SVM
       is the corresponding feature
      IF (SVMGainRatio < NextGainRatio) BREAK
      Update Examples-, Examples+ using the BESTSVM
      SelectedSet ← A_SVM
      Update classifier of root node
      IF (SelectedSet = Attribute OR (Entropy(Examples-) = 0
       AND Entropy(Examples+) = 0)) BREAK
      IF (Entropy(Examples-)!=ZERO)
         AddLeft(SVMDT(Examples-,Attributes))
      ELSE Attach a leaf node with label = -
      IF (Entropy(Examples+)!=ZERO)
         AddRight(SVMDT(Examples+,Attributes))
      ELSE Attach a leaf node with label = +
      RETURN root node
   END
END
```

2.2.3 SVMDT의 가지치기(Pruning) 알고리즘

SVMDT에서 가지치기 알고리즘은 루트 노드에서 시작하여 단말 노드로 검색하여, 자손(Descendant) 노드들이 사용한 속성을 모두 사용하여 지지 벡터 머신을 사용하여 학습을 한다. 획득율(Gainratio)이 한계 손실 획득률(Gainratio loss threshold) 보다 작을 때, 지지 벡터 머신 노드를 자식 노드로 사용하고, 자손 노드들은 모두 제거한다. 그렇지 않으면 자식 노드들로 이동하여 위의 과정을 반복한다. 여기서 이 알고리즘의 이름을 재귀 자손 가지치기(Recursive descendants pruning) 이라고 하겠다. [표 3]은 SVMDT 가지치기 알고리즘의 Pseudo 코드이다.

[표 3] SVMDT 가지치기 알고리즘

```
RecursiveDNP(Root, Examples)
BEGIN
   IF (Root is Leaf Node) THEN RETURN NULL
   A is a set of attributes that are used in all descendant
    nodes of Root
   RootGainRatio = GainRatio(Examples, Root)
   SVMNew = an SVM trained with Examples using A
   SVMGainRatio = GainRatio(Examples, SVMNew)
```

```
IF (SVMGainRatio-RootGainRatio < GainRatioLossThreshold)
THEN
    Examples+ is a subset of Examples classified to positive.
    Ret=RecursiveDNP(Root.right_child, Examples+)
    IF (Ret != NULL) Root.right_child = Ret
    Examples- is a subset of Examples classified to negative
    Ret = RecursiveDNP(Root.left_child, Examples-)
    IF (Ret != NULL) Root.left_child = Ret
ELSE
    RETURN SVMNEW
ENDIF
END
```

이 알고리즘의 특징은 한계 손실 획득율이 0이면, 학습 데이터에 대한 비손실 가지치기를 할 수 있다. 또한 한계 손실 획득율이 충분히 크다면, 노드는 하나가 된다. 본 연구에서는 한계 손실 획득율을 증가시키면서, 최적의 한계 손실 획득을 구했다. 이러한 불편을 없애는 방법은 본 연구에서는 다루지 않겠다.

3. 실험 및 결과

SVMDT의 SVM은 libSVM 2.81[4]과 SVMTorch2[5]와 합쳐서 만든 것이다. 두 알고리즘은 SMO알고리즘 기반으로, libSVM 2.81에서의 움추림(Shrink)알고리즘[4]과 SVMTorch2[5]의 커널 캐쉬를 조합하여 SVMDT에 탑재하였다.

3.1. 실험 데이터

본 논문에서 실험을 수행한 WDBC, IONOSPHERE, PIMA와 SPAMBASE 데이터는 UCI Machine learning repository의 기계학습 공개 도메인에서 가져온 실제 데이터이다. [표 4]는 UCI 데이터 셋의 정보이다.

[표 4] UCI Machine Learning Repository 에서 발췌한 데이터

데이터 이름	속성수	클래스 수	데이터 개수
WDBC	30	2	511
PIMA	8	2	768
IONOSPHERE	35	2	327
SPAMBASE	57	2	4601
Echocardiogram	8	2	108

UCI 데이터 셋 중 손실값(Missing value)이 있는 데이터는 제외하고 실험하였다.

3.2 실험 결과

각각 10등분을 해서 그 중 하나의 집합은 검증 데이터로, 나머지는 학습데이터로 분류기를 구성하여 10번의 학습결과의 평균치를 내었다. 그리고 성능 평가를 위한 수치로 정확도(Accuracy)를 사용하였다.

$$accuracy = \frac{the\ number\ of\ hit\ instance}{the\ number\ of\ total\ instances}$$

하지만, 이 분류기는 음의 클래스를 분류하는 능력이 없기 때문에 정확도만으로는 분류의 성능을 판단할 수 없다. 분류기의 분류 성능을 측정하기 위해 F1-measure를 택하였다. 이후엔 편의상 F-measure라고 하겠다.

$$F-measure = \frac{2rp}{r+p}$$

r과 p는 각각 Recall, Precision로 다음과 같다.

$$Precision = \frac{TP}{TP+FP},\ Recall = \frac{TP}{TP+FN}$$

즉, F-measure는 양의 클래스에서 양인 것을 지적해 주는 것과, 양이라고 판단한 것 중에 실제 데이터가 어느 하나로 치우쳤을 때 그 치우침이 미친 영향을 고려하여 성능을 평가하기 위해 사용된다.

3.2.1 SVMDT vs OC1

[표 5]는 UCI 데이터를 이용하여 SVMDT와 OC1 성능을 비교하기 위한 실험 결과이다.

[표 5] UCI 데이터의 OC1(가지치기 사용)과 SVMDT(선형 커널, 실험상 최적의 한계 손실 획득율 사용) 비교

데이터 이름	OCI		
	정확도	F-measure	단말노드 수
WDBC	95.07%	93.33%	1.3
PIMA	73.56%	61.91%	5.7
IONOSPHERE	81.48%	86.36%	3.2
SPAMBASE	82.09%	73.35%	4.3
Echocardiogram	63.8%	13.62%	5

데이터 이름	SVMDT			
	한계 손실 획득율	정확도	F-measure	단말 노드 수
WDBC	0.05	97.89%	95.65%	3.12
PIMA	0.1	76.95%	61.77%	8.7
IONOSPHERE	0.0	88.03%	90.94%	10.6
SPAMBASE	0.05	91.11%	89.44%	4.9
Echocardiogram	0.2	66.67%	36.36%	6.6

전반적으로 OC1의 성능을 능가하고 있다. SVMDT는 OC1과 달리 가지치기 알고리즘이 체계적이지 못하다. OC1에 비해 SVMDT가 단말노드의 수가 많다는 점에서 체계적인 가지치기 알고리즘이 있으면 더욱 높은 성능을 보여줄 수 있다는 점에서 SVMDT의 우월성을 확인할 수 있다.

3.2.2 SVMDT vs C4.5

[표 6]은 UCI 데이터를 이용하여 SVMDT와 C4.5 성능을 비교하기 위한 실험 결과이다.

[표 6] UCI 데이터의 C4.5(가지치기 사용)과 SVMDT(선형 커널, 실험상 최적의 한계 손실 획득율 사용) 비교

데이터 이름	C4.5		
	정확도	F-measure	단말노드 수
WDBC	94.02%	93.97%	6.4
PIMA	72.26%	60.98%	3.4
IONOSPHERE	82.67%	86.36%	6.6
SPAMBASE	86.41%	80.65%	25.2
Echocardiogram	71.32%	11.75%	5

데이터 이름	SVMDT			
	한계 손실 획득율	정확도	F-measure	단말 노드 수
WDBC	0.05	97.89%	95.65%	4.2
PIMA	0.1	76.95%	61.77%	4.8
IONOSPHERE	0.0	88.03%	90.94%	10.6
SPAMBASE	0.05	91.11%	89.44%	11.9
Echocardiogram	0.2	66.67%	36.36%	6.6

전반적으로 SVMDT가 C4.5보다 정확도뿐만 아니라 F-Measure의 경우에도 높은 수치를 보인다. Echocardiogram의 경우, 정확도는 C4.5가 좋지만 F-measure가 현저하게 떨어진다. 즉, C4.5가 분리하는 것이 양의 클래스 또는 음의 클래스에 현저하게 치우쳐 있음을 나타낸다. 이 점에서 SVMDT는 단일 변수만을 사용하는 C4.5보다 이 데이터에 대해서 좋은 성능을 보이고 있음을 알 수 있다.

3.2.3 SVM vs SVMDT

[표 7]은 선형 커널에서 UCI 데이터를 이용하여 SVM과 SVMDT의 성능을 비교하기 위한 실험 결과이다.

[표 7] UCI 데이터의 SVM과 SVMDT(선형 커널, 실험상 최적의 한계 손실 획득율 사용) 비교

데이터 이름	SVM	
	정확도	F-measure
WDBC	96.84%	95.71%
PIMA	76.83%	62.76%
IONOSPHERE	87.18%	90.52%
SPAMBASE	90.92%	88.36%
Echocardiogram	65.71%	25.81%

데이터 이름	SVMDT		
	한계 손실 획득율	정확도	F-measure
WDBC	0.05	97.89%	97.11%
PIMA	0.1	76.95%	61.77%
IONOSPHERE	0.1	89.74%	91.87%
SPAMBASE	0.05	91.11%	88.61%
Echocardiogram	0.15	66.67%	36.37%

선형 커널을 사용한 결과 SVMDT가 조금 좋은 성능을 보였는데 이는 SVMDT가 가설 공간을 조금 더 유연하게 사용하며, 가지치기를 하여 보다 분류 성능에 방해가 되는 요소를 줄이고 있기 때문이라고 판단된다.

[표 8]은 2차 다항 커널에서 UCI 데이터를 이용하여 SVM과 SVMDT의 성능을 비교하기 위한 실험 결과이다.

[표 8] UCI 데이터의 SVM과 SVMDT(2차 다항 커널, 실험상 최적의 한계 손실 획득율 사용) 비교

데이터 이름	SVM	
	정확도	F-measure
WDBC	95.43%	93.92%
PIMA	76.69%	62.00%
IONOSPHERE	87.18%	90.36%
SPAMBASE	92.96%	92.40%
Echocardiogram	65.71%	37.12%

데이터 이름	SVMDT		
	한계 손실 획득율	정확도	F-measure
WDBC	0.05	97.01%	95.96%
PIMA	0.1	76.69%	61.34%
IONOSPHERE	0.1	90.03%	92.60%
SPAMBASE	0.05	87.05%	81.82%
Echocardiogram	0.1	69.52%	40.66%

이 결과에서는 SVMDT와 SVM이 비슷한 성능을 보였는데 이는 SVMDT가 과잉적응을 해결하지 못해서이다. 이러한 문제점을 해결하려면 보다 정밀한 과잉 적응을 해결하는 것이 고안되어야 한다.

3.2.4 SVMDT 자체 비교

[표 9]는 선형 커널과 2차 다항 커널에서 UCI 데이터를 이용하여 SVMDT의 성능을 비교하기 위한 실험 결과이다.

[표 9] UCI 데이터의 SVMDT(선형 커널)과 SVMDT(2차 다항 커널) 비교

데이터 이름	SVMDT(선형 커널)		
	정확도	F-measure	단말노드 수
WDBC	97.89%	97.11%	4.2
PIMA	76.95%	61.77%	10
IONOSPHERE	89.74%	91.87%	29
SPAMBASE	91.11%	88.61%	11.9
Echocardiogram	66.67%	41.67%	6.6

데이터 이름	SVMDT(2차 다항 커널)		
	정확도	F-measure	단말 노드 수
WDBC	97.19%	95.96%	2.9
PIMA	76.69%	61.34%	5.9
IONOSPHERE	90.03%	92.60%	8.6
SPAMBASE	87.05%	81.82%	10.6
Echocardiogram	69.52%	35.48%	10

전반적으로 2차 다항 커널이 노드 수는 선형 커널 보다 적지만 좋지 않은 성능을 보였다. 다항 커널의 표현 범위가 선형 커널 보다 크기 때문에 노드 수가 줄어든 것이지만, 전반적으로 데이터가 선형적인 성질을 따르기 때문이다 판단된다.

다항 커널을 사용하여 성능이 좋아지지 않은 경우는 그 노드 수가 크게 감소하지 않았다. 이는 2차 다항 커널은 데이터의 속성 공간을 높인 반면, 그를 통해 얻어진 정보 획득이 적기 때문이다.

[표 10]은 가지치기를 하지 않을 경우 선형 커널과 2차 다항 커널에서 SVMDT의 다변수 노드 수를 비교한 것이다.

[표 10] UCI 데이터의 SVMDT(선형 커널)과 SVMDT(2차 다항 커널) 다변수 노드수의 비교 (가지치기 하지 않을 경우)

데이터 이름	SVMDT(선형 커널)		
	단일변수	다변수	비율
WDBC	18.2	1.0	0.05
PIMA	178.9	9.1	0.05
IONOSPHERE	16.2	4	0.20
SPAMBASE	464.9	25.2	0.05
Echocardiogram	27.3	1.3	0.05

데이터 이름	SVMDT(2차 다항 커널)		
	단일변수	다변수	비율
WDBC	9.9	1.1	0.1
PIMA	178.4	10.9	0.06
IONOSPHERE	10.7	3.7	0.26
SPAMBASE	559	44.5	0.074
Echocardiogram	25.2	2.2	0.08

이 결과에서는 가지치기를 하지 않고 100에 가까운 학습이 끝난 후 나온 노드의 수이다. 다항 커널을 사용했을 경우 SPAMBASE에서만 노드의 수가 많아졌고, PIMA에서는 거의 비슷한 수의 노드의 수가 나왔다. 이러한 점이 다변수 노드의 비율이 2차 다항 커널을 썼을 경우에 [표 9]에서 SPAMBASE와 PIMA와 같이 선형 커널보다 떨어지게 나오게 했다.

[표 11]은 실험상 최적의 가지치기를 했을 경우 선형 커널과 2차 다항 커널에서 SVMDT의 다변수 노드 수를 비교한 것이다.

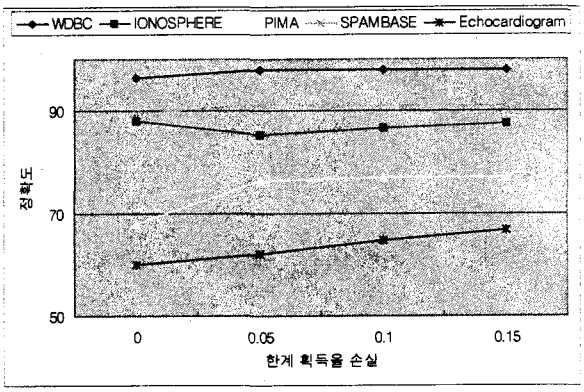
[표 11] UCI 데이터의 SVMDT(선형 커널)과 SVMDT(2차 다항 커널) 다변수 노드수의 비교 (실험상 최적의 가지치기 했을 경우)

데이터 이름	SVMDT(선형 커널)		
	단일변수	다변수	비율
WDBC	2.2	1.0	0.31
PIMA	5	4	0.44
IONOSPHERE	1	2	0.66
SPAMBASE	3	8	0.73
Echocardiogram	3	3	0.5

데이터 이름	SVMDT(2차 다항 커널)		
	단일변수	다변수	비율
WDBC	0.6	1.1	0.65
PIMA	5	4	0.44
IONOSPHERE	0	1	0.1
SPAMBASE	5	5	0.5
Echocardiogram	3	6	0.66

[표 10]에서 가지치기를 하지 않았을 경우에는 대다수가 단일 변수 노드였지만 재귀 자손 가지치기를 한 경우에는 다변수 노드의 비율이 상당이 커짐을 [표 11]에서 확인할 수 있다. 그리고 이 경우에도 PIMA와 SPAMBASE 데이터는 노드의 수가 차이가 얼마 나지 않는다. 이것은 2차 다항 커널이 선형 커널 보다 떨어지는 성능을 보이는 원인이 됨을 확인할 수 있다.

[그림 2]는 한계 획득율 손실에 따른 정확도를 측정한 결과이다.



[그림 2] 한계 획득율 손실에 따른 10겹 교차 검증의 정확도의 경향 (선형 커널)

위의 그림에서 보는 것과 같이 한계 획득율 손실에 따 검증 데이터의 정확도가 최고점에 이르다가 떨어지는 것을 볼 수 있다. 최고점을 측정하는 정도를 통계적으로 정할 수 있는 방법은 재귀 자손 가지치기에서 도입하지 않았다.

재귀 자손 가지치기는 통계적인 접근 방법을 사용할 수 있도록 한 점은 본 연구에서 포함하지 않았다. 하지만 [그림 2]의 결과를 통해서 적절한 한계 획득율 손실을 실험을 통해 정해야 한다는 불편함을 보여 주지만, 동시에 SVMDT를 위해 설계한 가지치기 기법을 수정하면 보다 좋은 결과를 얻을 수 있는 가능성을 보여 주고 있다.

4. 결론 및 성과

본 논문은 지지 벡터 머신을 이용한 다변수 결정 트리를 구성하는 알고리즘을 제안하였다.

속성 선택 알고리즘은 얻고자 하는 획득율을 단변수로 구성할 때보다 적은 노드를 사용하여 얻을 수 있도록 했다.

그리고 기존의 ID3의 알고리즘을 속성 선택 알고리즘에 맞게 수정하였고, 연속인 수치 속성과 다변수 결정트리 구조에 맞게 변형하였다.

재귀 자손 가지치기는 지지 벡터 머신의 학습 데이터가 적은 상황에서도 강인한 성능을 발휘할 수 있다는 장점을 살린 알고리즘이다.

또한 다변수 결정 트리 측면에서 지지 벡터의 사용은 선형으로 분리가 가능한 경우에는 보다 작은 수의 노드

로 구성할 수 있게 했다.

뿐만 아니라 선형으로 분리가 가능하지 않는 경우의 데이터는 커널 함수를 사용하여 보다 적은 노드를 가지고 분리가 가능하도록 했다. 하지만, 선형의 성질을 가진 데이터에 대해서는 크게 노드의 수를 줄이지 못하고, 오히려 성능이 떨어진 경우도 있었다. 즉, 비선형 커널을 사용할 때는 데이터의 특징을 파악한 다음 사용해야 한다.

5. 향후 과제

이 연구 논문에서 다루지 못한 부분은 크게 3가지라고 할 수 있다. 앞으로 이러한 점을 보완해 나가야 할 것이다.

첫째로 분류하고자 하는 클래스가 세 가지 이상일 때, 즉 다중 클래스 분류(Multi-class classification)일 경우에 처리 방안을 제시하지 못했다.

둘째로 명목 속성, 즉 속성이 수치가 아닐 때의 처리 방안을 다루지 않았다.

셋째로 가지치기 알고리즘에서의 한계 손실 획득율의 지정을 해야 하고, 한계 손실 획득율에 따른 가지치기 정도를 본 연구만으로 예상할 수 없다.

참고문헌

[1] N. Cristianini and J. Shawe-Taylor, An Introduction Support Vector Machine, Cambridge University Press, Cambridge, 2000.

[2] R.O. Duda, P.E. Hart and D.G. Stork, Pattern Classification, John Wiley & Sons, Inc., New York, 2000.

[3] M. Nunez, "The use of Background Knowledge in Decision Tree Induction," Machine Learning, vol.6, no.3, pp.77-100, 1991.

[4] C-C, Chang and C-J, Lin, LIBSVM: A Library for Support Vector Machine, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[5] R. Collobert and S. Bengio, "SVMTorch: a Support Vector Machine for Large-Scale Regression and Classification Problems," Journal of Machine Learning Research, vol.1, pp.143-160, 2001.