특징 선택을 위한 문제 공간과 알고리즘 동작 분석

(Analysis of Problem Spaces and Algorithm Behaviors for Feature Selection)

이 진 선 † 오 일 석 ††
(Jin-Seon Lee) (Il-Seok Oh)

요 약 특징 선택 알고리즘들은 좋은 해를 찾기 위해 거대한 문제 공간을 폭넓게 효율적으로 탐색하여야 한다. 이 논문에서는 문제 공간의 적합도 지형을 통찰해보고자 하였으며, 알고리즘들의 탐색 능력을 개선하였다. 지역 최고값과 최저값에 대한 통계에 의해 해 공간을 조사한다. 또한 기존 알고리즘들의 동작을 분석하고 이들의 해를 개선하였다.

키워드 : 특징 선택, 문제 공간, 순차 탐색 알고리즘, 지역 최대점과 최소점, 적합도 지형

Abstract *The feature selection algorithms should broadly and efficiently explore the huge problem spaces to find a good solution. This paper attempts to gain insights on the fitness landscape of the spaces and to improve search capability of the algorithms. We investigate the solution spaces in terms of statistics on local maxima and minima. We also analyze behaviors of the existing algorithms and improve their solutions.*

Key words : Feature Selection, Problem Spaces, Sequential Search Algorithm, Local Maxima and Minima, Fitness Landscape

1. 서 론

특징 선택은 임의의 최적 조건하에서 원래 특징 집합으로부터 부분 집합을 선택하는 문제이다. 특징 선택의 문제 공간은 특징의 개수가 증가함에 따라 지수적으로 커진다. SFS(Sequential Forward Search), PTA(Plus-l-Take-away-r), SFFS(Sequential Floating Forward Search)와 같은 순차 탐색 알고리즘들은 가장 의미없는 특징들은 제거하고 가장 의미있는 특징들은 추가하는 연산을 사용하여 부 최적 해(suboptimal solution)를 찾는다. 알고리즘들은 문제 공간을 결정론적으로 탐색한다[1]. 이러한 알고리즘들과 달리, 유전 알고리즘은 해 집단을 유지하고 부모 해에 교차와 돌연 변이 연산을 적용함으로써 문제공간을 보다 넓고 동적으로 탐색한다[2].

알고리즘의 성공 여부는 거대한 문제 공간을 얼마나 폭넓게 그리고 효율적으로 탐색하느냐에 달려있다. 따라서 문제 공간에 대한 이해는 기존 알고리즘을 개선하거나 새로운 알고리즘을 설계하는데 효과적인 방법이 될 수 있다. 그러나 문제 공간의 거대한 크기로 인해 문제 공간에 대한 완전한 이해는 불가능하다. 한 가지 가능한 방법은 문제 공간의 적합도 지형(fitness landscape)을 통찰해보는 것이다. 적합도는 해의 좋음 또는 성능을 나타내기 위해 유전 알고리즘에서 사용하는 용어이다.

유전 알고리즘의 연구자들은 이미 순회 세일즈맨 문제나 그래프 이등분 문제와 같은 몇몇 조합적 문제들의 적합도 지형을 통찰해보는 연구를 수행해 왔다. Boese 등은 지역 최적해 간의 관계와 지역 최적해와 전체 최적해 간의 관계를 분석하였다[3]. Jones와 Forrest는 적합도 거리 상관관계를 측정하였고 문제의 난이도를 평가하기 위해 이 값들을 사용하였다[4]. Kim과 Moon은 그래프 이등분을 위한 지역 최적해 공간에 대한 경험적인 연구를 수행 하였고 지역 최적해의 중심 지역의 중요성을 지적하였다[5].

이 논문에서는 지역 최고값과 최저값을 사용하여 특징 선택의 해 공간을 통찰한다. 해 공간에 대해 보다 잘 이해함으로써 기존 알고리즘들의 동작을 분석하고 이들의 해를 개선하고자 하였다. 2장에서는 특징 선택의 문제 공간을 정의한다. 3장에서는 지역 최고값과 최저값을

· 이 연구는 한국과학재단 특정기초연구(R01-2003-000-10879-0)의 지원으로 수행되었습니다.
† 정 회 원 : 우석대학교 게임콘텐츠학과 교수
jslee@woosuk.ac.kr
†† 종신회원 : 전북대학교 전자정보공학부 교수
isoh@chonbuk.ac.kr
논문접수 : 2004년 8월 10일
심사완료 : 2006년 4월 26일

분석함으로써 적합도 지형의 구조를 통찰해본다. 몇 가지 통계 자료가 제시된다. 4장에서는 지역 탐색 알고리즘들의 동작을 분석하고 이들의 해를 개선하기 위한 경험적 지식을 제안한다. 마지막으로 5장에서 결론을 내린다.

2. 문제 공간의 정의

특징 선택 문제는 주어진 최적 조건 하에서 D개의 특징을 갖는 전체 집합으로부터 d개의 특징을 갖는 부분 집합을 선택하는 문제이다. D와 d는 문제 크기를 정의한다. D개의 특징들을 1에서 D까지의 서로 다른 정수를 사용하여 표기하고, D개의 특징을 갖는 전체 특징 집합을 $U=\{1,2,\cdots,D\}$로 표기한다. 특징 선택 과정에서 X는 선택된 특징들의 집합을 나타내고 Y는 이때 선택되지 않은 나머지 특징들의 집합이라 하자. 어떤 순간에서든 $U=X\cup Y$이다. J(X)는 X의 성능을 평가하는 평가함수이다.

정의 1 (공간) : D-공간은 가능한 모든 X의 집합이며, 이를 문제 공간 (problem space)이라 한다. d-공간은 D-공간의 부분집합이며, $|X|=d$ 인 부분집합 X로 정의된다. d-공간은 해 공간(solution space)이라 한다. d-공간은 $S=\{X_i \mid |X_i|=d,\ 1\le i\le {}_DC_d\}$ 로 나타낸다. 예를 들어 $D=3$, $d=2$ 일때 $S = \{\{1,2\}, \{1,3\}, \{2,3\}\}$ 이다. S의 크기는 ${}_DC_d$ 이다.

정의 2 (거리) : d-공간에서 두 개의 해 X_i와 X_j 사이의 거리는 차 집합 (set difference) 연산자를 사용하여 $D(X_i,X_j) = |X_i-X_j|$ 로 정의한다.

정리 1 : d-공간에서 두 개의 해 간의 최대 거리는 $1\le d\le D/2$ 인 경우에는 d, 그리고 $D/2<d\le D$ 인 경우에는 $D-d$ 이다.

정의 3 (이웃) : 만일 d-공간 상의 두 해에 대해 $D(X_i,X_j)=1$이면, X_i와 X_j는 이웃이라 한다. 어떤 해의 이웃의 집합은 $N(X_i) = \{X_j \mid D(X_i,X_j)=1,\ i\neq j\}$로 정의한다.

정리 2 : 어떤 해의 이웃의 개수는 $|N(X_i)| = d(D-d)$ 이다.

정의 4 (최고값과 최저값) : 모든 $X_j\subseteq N(X_i)$에 대해 $J(X_i)\ge J(X_j)$이면 X_i는 지역 최고값이다. 모든 $X_j\subseteq N(X_i)$에 대해 $J(X_i)\le J(X_j)$이면 X_i는 지역 최저값이다. 지역 최대값(최저값) 중에서 가장 좋은(가장 나쁜) 해는 전역 최고값(최저값)이다.

정의 5 (고원) : 고원(plateau)은 지역 최고값의 연결 요소이다. 연결된 지역 최고값의 개수는 고원의 크기이다.

3. 해 공간의 조사

이 절에서는 최고점과 최저점에 관한 통계 값을 사용하여 해 공간의 적합도 지형 구조를 조사한다. 실험을 위해 UCI repository에서 다섯 개의 데이타 집합을 선택하였다[6]. 표 1이 이들을 보여준다. 실험 결과의 객관성을 높이기 위해, 특징 수를 적은 것(Wine의 경우 13개)부터 큰 것(Sonar의 경우 60개)까지 선택하여 문제 공간의 크기를 다양하게 하였다.

작은 문제 공간을 가진 세 개의 데이타 집합에 대해서, 네 가지 d 값에 대하여 d-공간을 전역적으로 (exhaustively) 탐색하였다. 표 2는 지역 최고값과 최저

표 1 실험 데이타집합

	특징 수	부류 수	샘플 수
Wine	13	3	178
Vehicle	18	4	846
Segmentation	19	7	훈련 2100, 검사 210
WDBC	30	2	569
Sonar	60	2	208

표 2 전역 탐색에 의해 찾은 지역 최고점과 최저점에 대한 통계
($^*d$ = D의 1/5, 2/5, 3/5, 그리고 4/5)

데이타집합	d^*	해의 개수 ($={}_DC_d$)	지역 최고점			지역 최저점		
			개수	인식률 범위	평균	개수	인식률 범위	평균
Wine (D=13)	3	286	5	(93.8,93.8)	93.8	2	(48.3,55.6)	52.0
	5	1287	11	(93.8,95.5)	94.5	8	(61.8,71.4)	63.5
	8	1287	2	(93.8,95.5)	94.7	18	(65.2,71.9)	71.5
	10	286	2	(92.1,92.7)	92.4	1	(71.9,71.9)	71.9
Vehicle (D=18)	4	3060	9	(65.0,69.7)	67.5	3	(39.8,42.4)	41.2
	7	31824	14	(69.4,73.5)	71.5	5	(47.4,53.6)	49.8
	11	31824	16	(70.8,72.5)	71.6	1	(50.4,50.4)	50.4
	14	3060	3	(69.4,70.8)	70.3	1	(54.6,54.6)	54.6
Segmentation (D=19)	4	3876	2	(84.0,92.8)	88.4	1	(24.9,24.9)	24.9
	8	75582	10	(92.1,93.0)	92.8	1	(32.5,32.5)	32.5
	11	75582	36	(92.3,93.0)	92.5	2	(52.3,63.6)	58.0
	15	3876	6	(92.3,92.6)	92.4	1	(72.6,72.6)	72.6

값의 개수, 그리고 인식률의 범위와 이들의 평균값을 보여준다. 지역 최고값은 d-공간에서 매우 드물게 나타남을 알 수 있다. 지역 최고값이 가장 많은 경우는 Segmentation에서 d=11일 때 36 개이다. 표 2는 지역 최저값은 더욱 드물게 나타남을 보여준다.

해의 성능은 1-NN 분류기를 사용하여 얻은 인식률로 측정하였다. 해의 성능 그 자체를 적합도로 사용하였으며, 이 적합도 값이 2장에서 사용한 표기법 J(X)와 같다.

지역 최고값의 성능 변화는 크지 않다. 가장 큰 차이는 Vehicle에서 d=4인 경우의 4.73%이다. 지역 최고값과 지역 최저값 사이의 차이는 20% 에서 60% 사이에 놓여있다. 세 개의 데이타 집합에 대해 지역 최저값은 결코 지역 최고값을 능가하지 않는다. 지역 최저값 간의 성능 변화는 지역 최고값 간의 성능 변화보다 더 크다. 예를 들어, Wine에서 d=5인 경우 약 10%의 차이를 보인다.

데이타 집합 WDBC와 Sonar는 매우 큰 해 공간을 가진다. 따라서 전역 탐색은 계산 시간상 불가능하여, 각각 임의적으로 10,000 개와 100,000 개의 해를 생성하였다. 각각의 해에 대해 지역 최고값과 최저값 여부를 검사하였다. 표 3은 결과를 보여준다. 다른 데이타집합과 달리, WDBC는 다른 데이타에 비해 조밀한 지역 최고점과 최저점을 갖는다. d=24에 대해, 지역 최고점은 전체 해 공간의 약 16%를 차지한다. 우리는 고원을 찾고 이들의 크기를 계산함으로써 이러한 지역 최고점의 모양을 살펴보았다. 분석은 생성된 해들만을 사용하여 수행하였다. 표 4는 작은 고원이 큰 고원보다 훨씬 많음을 보여준다. Sonar는 다섯 개의 데이타집합 중 지역 최고점과 최저점이 가장 희소한 분포를 갖는다.

4. 알고리즘 동작과 성능 개선

알고리즘의 동작은 분류의 정확도와 계산 시간 면에서 평가하였다. 정확도는 기각없이 1-NN 분류기의 인식률에 의해 측정하였다. 계산 시간은 원자적 연산(atomic operation)의 개수로 측정하였다. 원자적 연산은 고정된 양의 CPU 사이클을 사용한다[2]. t(s)를 크기가 s인 특징 부분집합 X를 평가하는데 필요한 계산 시간이라 하자. t(s)는 크기 s 뿐만 아니라 훈련 샘플 집합의 크기에 따라 달라진다. 하지만 샘플 집합의 크기는 미리 고정되어있기 때문에 상수로 간주할 수 있다. 따라서 t(s)는 단지 s 값에만 의존한다. 특징 하나에 대한 평가를 원자적 연산이라 하고, 원자적 연산을 위한 t(1)을 원자적 시간 (atomic time)이라 부른다.

기존의 특징 선택 알고리즘의 성능을 개선하기 위해 두 가지 형태의 알고리즘을 제안한다. 알고리즘 UP1은 성능을 향상시키는 임의 방향으로 해를 움직인다. 알고리즘 UP2는 성능 개선이 가장 큰 방향으로 해를 움직인다. 따라서 UP1은 UP2보다 더 적은 량의 계산으로 한 단계를 이동할 수 있다. 반면 UP1은 UP2보다 지역 최고값에 다다르기 위해 더 많은 단계가 필요하다. 표 5는 UP1과 UP2에 의해 향상된 해를 보여준다.

```
Algorithm UP1 {
  Obtain a solution S by an existing algorithm;
  while (S is not local maximum) {
    Find a better solution S' among neighbors of S;
    S = S';
  }
  return S;
}
Algorithm UP2 {
  Obtain a solution S by an existing algorithm;
  while (S is not local maximum) {
    Find the best solution S' among neighbors of S;
```

표 3 임의 생성으로 찾은 지역 최고점과 최저점에 대한 통계

데이타 집합	d^*	생성된 해의 개수	지역 최고점			지역 최저점		
			개수	인식률 범위	평균	개수	인식률 범위	평균
WDBC (D=30)	6	10,000	33	(91.4,94.2)	93.0	28	(66.6,86.3)	83.6
	12	10,000	188	(91.4,94.2)	91.9	17	(81.7,90.0)	86.6
	18	10,000	469	(91.4,93.9)	91.7	29	(82.1,90.2)	89.2
	24	10,000	1599	(91.4,93.9)	91.6	243	(90.0,90.2)	90.2
Sonar (D=60)	12	100,000	7	(87.0,89.4)	88.3	1	(76.0,76.0)	76.0
	24	100,000	4	(85.6,89.4)	88.3	5	(74.5,81.3)	78.0
	36	100,000	5	(87.0,89.9)	88.7	2	(78.9,80.3)	79.6
	48	100,000	5	(87.0,88.9)	87.9	14	(79.3,81.3)	80.2

표 4 WDBC의 고원 분포 (d=24)

크기	1	2	3	4	5	6	7	8	9	10	11	12
개수	446	182	73	41	14	11	17	6	3	3	2	2

```
    S = S';
  }
  return S;
}
```

세 가지 순차적 탐색 알고리즘 SFS(Sequential Forward Search), PTA(l=2,r=1), 그리고 SFFS(Sequential Forward Floating Search)의 성능을 측정하였다. PTA는 Plus-l-Take-away-r을 의미한다. 이 세 가지 알고리즘은 아래와 같이 기술할 수 있다. 보다 자세한 내용을 원하는 경우에는, SFS와 PTA에 대해서는 [1]번 논문을, SFFS에 대해서는 [7]번 논문을 참조하기 바란다.

지역 탐색 연산:

rem: Choose the least significant feature x in X such that x = $\text{argmax}_{a\in X}J(X-\{a\})$, and move x to Y.
add: Choose the most significant feature y in Y such that y = $\text{argmax}_{a\in Y}J(X\cup\{a\})$, and move y to X.
REM(k): Repeat operation *rem* k consecutive times.
ADD(k): Repeat operation *add* k consecutive times.

Algorithm SFS:
1. $X=\varnothing$; $Y=\{i|\ 1\le i\le D\}$;
2. ADD(d);

Algorithm PTA(l,r) with $l>r$:
1. $X=\varnothing$; $Y=\{i|\ 1\le i\le D\}$
2. **repeat** {*ADD*(l); *REM*(r);} **until** $|X|=d$;

Algorithm SFFS:
1. $X_0=\varnothing$; $Y=\{i|\ 1\le i\le D\}$; k=0; // initialization
2. **while** ($k<d+\delta$) {
3. y = $\text{argmax}_{a\in Y-X_k}J(X_k\cup\{a\})$; //find the most significant feature
4. X_{k+1} = $X_k\cup\{y\}$; k++;
5. x = $\text{argmax}_{a\in X_k}J(X_k-\{a\})$ // find the least significant feature
6. **while** ($J(X_k-\{x\})>J(X_{k-1})$) {
7. X_{k-1} = $X_k-\{x\}$; k--;
8. x = $\text{argmax}_{a\in X_k}J(X_k-\{a\})$; // find the least significant feature
9. }
10. }

표 5는 실험 결과를 보여준다. UP1과 UP2에 의해 성능 향상이 이루어진 경우는 굵은 글씨체로 강조하였다. Wine의 d=3과 d=8인 경우, 세 알고리즘 모두 전역 최고값을 찾았다. 그리고 d=5와 d=10인 경우에 SFS와 PTA에서 지역 최고값을 생성하였다. 따라서, Wine에 대해서는 아무런 성능 향상도 없었다.

표 5 다섯 개 데이타집합의 인식률 향상 (단위: %)

데이타집합	d^*	전역 최고값	SFS			PTA			SFFS		
			SFS	UP1	UP2	PTA	UP1	UP2	SFFS	UP1	UP2
Wine (D=13)	3	93.8	93.8	93.8	93.8	93.8	93.8	93.8	93.8	93.8	93.8
	5	95.5	94.4	94.4	94.4	94.4	94.4	94.4	94.9	94.9	94.9
	8	95.5	95.5	95.5	95.5	95.5	95.5	95.5	95.5	95.5	95.5
	10	92.7	92.1	92.1	92.1	92.1	92.1	92.1	92.7	92.7	92.7
Vehicle (D=18)	4	69.7	62.8	**69.7**	**69.7**	64.8	**67.1**	**67.1**	69.2	**69.7**	**69.7**
	7	73.5	69.2	**71.5**	**71.5**	70.1	**71.5**	**71.5**	73.5	73.5	73.5
	11	72.5	69.5	**71.8**	**72.1**	71.8	71.8	71.8	71.9	71.9	71.9
	14	70.8	68.2	**70.8**	**70.8**	70.8	70.8	70.8	70.8	70.8	70.8
Segmentation (D=19)	4	92.8	92.8	92.8	92.8	92.8	92.8	92.8	92.8	92.8	92.8
	8	93.0	93.0	93.0	93.0	93.0	93.0	93.0	93.0	93.0	93.0
	11	93.0	93.0	93.0	93.0	93.0	93.0	93.0	93.0	93.0	93.0
	15	92.6	92.6	92.6	92.6	92.6	92.6	92.6	92.6	92.6	92.6
WDBC (D=30)	6	94.9	93.2	93.2	93.2	93.2	93.2	93.2	94.2	94.2	94.2
	12	NA	92.6	**93.2**	**93.0**	93.0	93.0	93.0	94.2	**94.4**	**94.4**
	18	NA	94.0	**94.2**	**94.2**	94.2	94.2	94.2	94.2	94.2	94.2
	24	NA	92.4	**93.9**	**93.9**	93.5	**93.9**	**93.9**	93.9	93.9	93.9
Sonar (D=60)	12	NA	87.0	**90.9**	**89.4**	89.4	89.4	89.4	92.3	92.3	92.3
	24	NA	89.9	**90.4**	**90.4**	90.9	90.9	90.9	93.8	93.8	93.8
	36	NA	88.5	**90.9**	**91.4**	91.8	**94.7**	**95.2**	93.3	93.3	93.3
	48	NA	91.8	91.8	91.8	92.3	92.3	92.3	91.45	91.4	91.4

표 6 계산 시간 (단위: 원자적 연산의 수)

데이타 집합	d^*	SFS			PTA			SFFS		
		SFS	UP1	UP2	PTA	UP1	UP2	SFFS	UP1	UP2
Wine (D=13)		70	93	93	187	93	93	2184	93	93
	5	155	205	205	415	205	205	2184	205	205
	8	300	328	328	872	328	328	2184	328	328
	10	385	310	310	1230	310	310	2184	310	310
Vehicle (D=18)	4	160	604	684	412	456	684	6000	372	456
	7	392	707	1638	1022	1120	1638	6000	546	546
	11	748	2772	2574	2134	858	858	6000	858	858
	14	980	1806	2394	3122	798	798	6000	798	798
Segmentation (D=19)	4	170	244	244	436	244	244	4266	244	244
	8	516	712	712	1352	712	712	4266	712	712
	11	814	979	979	2277	979	979	4266	979	979
	15	1160	915	915	3725	915	915	4266	915	915
WDBC (D=30)	6	560	870	870	1370	870	870	25224	870	870
	12	1768	5196	5208	4468	2604	2604	25224	3576	5208
	18	3192	4032	7812	8862	3906	3906	25224	3906	3906
	24	4400	4248	13920	14120	3816	10440	25224	3480	3480
Sonar (D=60)	12	4108	14160	20772	9508	6924	6924	315353	3900	3900
	24	13400	33816	41520	32840	20760	20760	315353	12696	12696
	36	24420	39240	62280	66540	89532	124560	315353	24660	24660
	48	33712	27696	27696	107152	27696	27696	315353	25392	25392

Vehicle에 대해서는 d=4와 d=14인 경우에 SFS에 의한 해들이 UP1과 UP2에 의해 전역 최고값으로 향상되었다. d=4인 경우, SFFS에 의한 해도 전역 최고값으로 개선되었다.

Segmentation의 경우 세 가지 알고리즘 모두 전역 최고값을 생성하였다. WDBC의 경우, d=12인 경우 SFFS에 의해 해가 개선되었다. Sonar의 경우, SFS와 PTA 알고리즘의 몇가지 해들이 개선되었다.

SFFS는 Vehicle에서 d=4인 경우와 WDBC에서 d=12인 경우의 두 가지 경우를 제외하고 전역 또는 지역 최고값을 생성하였다. PTA와 SFS는 각각 지역 최고값이 아닌 4개와 10개의 해를 생성하였다. 이것은 SFFS가 SFS와 PTA 보다 전역 또는 지역 최고값을 찾는데 있어 우수하다는 점을 보여준다.

표 6은 계산 시간을 보여준다. 알고리즘 UP2는 UP1 보다 더 많은 시간을 사용하였다. 그러나 인식 성능 면에서 UP2가 UP1 보다 우월함을 보여주는 경우는 없었다. 따라서 UP1을 사용할 것을 권장한다.

5. 결 론

특징 선택을 위해 문제 공간을 정의하였다. 지역 최고값과 최저값에 대한 통계를 사용하여 실제 데이타 집합에 대한 문제 공간을 분석하였다. 또한 기존 알고리즘의 성능을 인식률과 계산시간 면에서 분석하였고 이들의 해를 개선하는 알고리즘을 제안하였다.

앞으로의 연구에서는 Simulated annealing이나 Tabu search, 그리고 유전 알고리즘 등 공간을 보다 동적으로 탐색하는 알고리즘들을 분석하고자 한다. 이러한 분석은 전역 최고값에 더욱 가까운 해를 찾도록 기존 알고리즘을 개선하는데 통찰력을 제공할 것이다.

참 고 문 헌

[1] J. Kittler, "Feature selection and extraction," in Handbook of Pattern Recognition and Image Processing, Academic Press (Edited by T.Y. Young and K.S. Fu), pp.59-83, 1986.

[2] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Local search-embedded genetic algorithms for feature selection," Proc. of 16th International Conf. on Pattern Recognition, Vol.II, pp.148-151, 2002.

[3] K.D. Boese, A.B. Kahng, and S. Muddu, "A new adaptive multi-start technique for combinatorial global optimizations," Operations Research Letters, Vol.16, pp.101-113, 1994.

[4] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," 6th International Conference on Genetic Algorithms, 1995.

[5] Y-H. Kim and B-R. Moon, "Investigation of the fitness landscapes and multi-parent crossover for graph bipartitioning," GECCO2003, pp.1123-1135, 2003.

[6] P.M. Murphy and D.W. Aha, "UCI repository for

machine learning databases (http://www.ics.uci.edu/~mlearn/MLRepository.html)," Irvine, CA: University of California, Department of Information and Computer Science, 1994.
[7] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," Pattern Recognition Letters, Vol.15, pp.1119-1125, 1994.

이 진 선

1985년 전북대학교 전산통계학과 졸업(학사). 1988년 전북대학교 대학원 전산통계학과(이학석사). 1995년 전북대학교 대학원 컴퓨터공학과(공학박사). 1988년~1992년 한국전자통신연구원 연구원. 1995년~2006년 2월 우석대학교 컴퓨터공학과 교수. 2006년 3월~현재 우석대학교 게임콘텐츠학과 교수. 관심분야는 멀티미디어, 패턴인식, 영상처리

오 일 석

1984년 서울대학교 컴퓨터공학과 졸업(학사). 1992년 KAIST 전산학과(공학박사). 1992년~현재 전북대학교 전자정보공학부 교수. 관심분야는 문서영상처리, 패턴인식, 컴퓨터비전, 유전알고리즘의 패턴인식 응용