공 학 석 사 학 위 논 문

딥러닝 회귀 및 분류 문제에서 변수 선택의 영향에 대한 연구

2021년 2월

부경대학교 대학원

인 공 지 능 융 합 학 과

이 승 재

공학석사 학위논문

딥러닝 회귀 및 분류 문제에서 변수 선택의 영향에 대한 연구

지도교수 장 대 홍

이 논문을 석사 학위논문으로 제출함.

2021년 2월

부경대학교 대학원

인 공 지 능 융 합 학 과

이 승 재

이승재의 공학석사 학위논문을 인준함.

2021년 2월 19일

위 원 장 이학박사 하 일 도 (인)

위 원 이학박사 박 동 준 (인)

위 원 이학박사 장 대 흥 (인)

목 차

표 차례

그림 차례

The Effect of Variable Selection on Deep Learning Regression and Classification Problem

Seung Jae Lee

Department of Artificial Intelligence Convergence, The Graduate School, Pukyong National University

Abstract

In predictive analysis, traditional variable selection methods or penalized regression are mainly used to solve the problem of multicollinearity between explanatory variables in regression analysis. In this study, we evaluate how these variable selection methods or penalized regression using real-world data affect model evaluation measures or the speed of computer computation, and compare how the effect of variable selection depends on the size of the data or input variables.

keywords : Deep learning, regression problem, classification problem, variable selection, penalized regression

제 1장 서론

1.1 연구배경

기계학습 알고리즘 중 하나인 심층 신경망 (deep neural network, DNN)은 입력층 (input layer)과 출력층 (output layer)사이에 여러 개의 은닉층 (hidden layer)들로 이뤄진 인공신경망 (artificial neural network, ANN)이다. 심층 신경망을 이용하면 예측분석에의 회귀 및 분류 문제를 다룰 수 있다. p개의 입력변수 (input variable)가 주어진 특정한 회귀 혹은 분류 문제를 해결하기 위하여 심층 신경망을 사용할 때, 심층 신경망 학습을 위해 p개의 입력변수를 모두 사용하는 것이 일반적이다. 통계학에서는 설명변수 사이의 다중공선성 문제와 최소제곱추정량의 계산 문제를 해결하기 위해 사용되는 변수선택 (variable selection)이나 벌점회귀 (penalized regression) 방법 (Kim, 2018)을 사용한다. 본 연구에서는 변수선택이나 벌점회귀 방법을 통하여 심층 신경망 학습 전에 미리 변수를 선택하는 것이 심층 신경망 모형 성능과 학습 완료 시간에 어떠한 영향을 미치는지 실제 자료들을 이용하여 탐색적으로 평가하였다. 변수선택과 벌점회귀 방법을 통하여 선택된 변수들로 학습한 모형이 자료의 모든 변수를 이용하여 학습한 모형보다 성능 좋고 학습 완료 시간이 짧다면 모든 변수를 수집하여 이용하는 것보다 상대적으로 적은 자원으로 심층 신경망을 학습시킬 수 있을 것이다.

1.2 연구 목적 및 내용

본 연구에서는 실제 자료들을 이용하여 심층 신경망으로 회귀 및 분류 문제를 해결하기 위해 입력변수로 주어진 변수를 모두 사용하였을 때의 심층 신경망 모형의 성능 및 학습시간과 변수선택 또는 벌점회귀 방법을 통하여 선택된 변수들을 입력변수로 사용하여 학습한 심층 신경망 모형의 성능 및 학습시간을 비교하고자 한다.

그림 1.2.1 연구의 작업 흐름도

연구의 작업 흐름도를 그림 1.2.1으로 나타냈다. 첫 번째로 회귀 혹은 분류 문제를 다루는 자료를 대상으로 자료의 크기에 따라 소규모 및 중규모, 고차원 자료로 분류하였다. 소규모 및 중규모 자료는 변수의 개수에 비해 인스턴스 (instance) 개수가 충분히 많은 자료이고, 고차원 자료는 변수의 개수가 인스턴스 개수보다 매우 많은 자료이다. 각 자료를 통하여 연구의 마지막 단계에서 변수전체를 이용하여 학습한 모형과 선택된 변수들로만 학습한 모형의 성능 및 학습 완료 시간을 비교하기 위해 서로 다른 8:2

비율의 학습 및 테스트 데이터셋 (dataset) 100개를 비복원 추출을 통하여 생성하였다. Jang 등(2020)은 고차원 자료를 대상으로 딥러닝 (deep learning) 시 변수선택 방법이나 벌점회귀를 하였을 때 모형의 평가 측도 및 학습 수행 완료 시간에 어떠한 영향을 주는지 평가하였다. 본 연구에서는 고차원 자료뿐만 아니라 다양한 특성을 가진 소규모 및 중규모 자료들을 추가하여 연구 결과가 일정한 패턴을 보이는지 혹은 자료에 의존하여 결과가 달라지는지 확인하였다. 두 번째 단계로는 소규모 및 중규모 자료에 대해서 변수선택을 할 때는 통계학의 전통적인 변수선택 방법의 하나인 단계적 방법 (stepwise selection)과 Tibshirani(1996)가 제안한 LASSO (least absolute shrinkage and selection operator) 벌점회귀 방법을 이용하였다. 그리고 고차원 자료에 대해서는 Fan과 Lv(2008)이 고차원 자료를 대상으로 제안한 SIS (sure independence screening)을 이용하여 벌점회귀를 통한 변수선택 이전에 종속변수 (dependent variable)와 독립변수 (independent variable) 사이의 상관관계를 고려하여 종속변수에 영향을 미치는 독립변수들을 선택하는 방법을 이용하였고 변수선택을 위한 벌점회귀 방법으로는 LASSO, SCAD (smoothly clipped absolute deviations) 그리고 MCP (minimax concave penalty)를 활용하여 변수선택을 진행하였다. 입력변수들은 심층 신경망 학습 전 모두 Z-점수 정규화 (z-score normalization)를 통해 입력변수들 사이의 스케일 (scale)을 맞추었다. 다음으로 자료의 모든 변수를 이용하여 심층 신경망을 학습을 진행하여 모형을 만들고, 변수선택 혹은 벌점회귀로 선택된 변수들만을 이용하여 심층 신경망을 학습한 모형을 만들었다. 심층 신경망 모형은 학습 데이터셋만을 사용하여 학습하였고, 학습 데이터와 테스트 데이터에 대한 성능이 더는 개선되지 않을 때 학습을 종료하며 학습시간을 저장했다. 마지막 단계에서는 자료의 모든 변수를 이용하여 학습한 심층 신경망 모형과

변수선택 혹은 벌점회귀로 선택된 변수들로만 학습한 심층 신경망 모형의 성능 및 학습 완료 시간을 비교하였다. 서로 다른 100개의 학습 데이터셋을 이용하여 심층 신경망 모형 100개를 학습하였고 100개의 테스트 데이터셋을 이용하여 회귀문제에서는 RMSE (root mean square error), 분류 문제는 정확도 (accuracy)와 AUC (area under the ROC curve) 값을 구하여 성능 지표의 분포를 구하여 통계적 검정을 통해 모형의 성능을 비교하였다.

심층 신경망 학습을 위해 Apache MXNet R 3.5.3 version을 활용하였고 GPU는 GTX 1060 with MAX-Q Design을 이용하여 모형을 학습하고 학습 완료 시간을 초 단위로 저장하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 네 종류의 소규모 및 중규모 자료의 변수들을 소개하고, 심층 신경망 학습에 변수선택 혹은 벌점회귀 방법을 사용하는 경우 모형의 성능 및 학습 완료 시간에 어떠한 영향을 주는지 탐색적인 방법과 통계적 검정을 통하여 평가하였다. 제 3장에서는 네 종류의 고차원 자료의 변수들을 소개하고, 심층 신경망 학습에 SIS 방법으로 벌점회귀 전 변수를 줄이고 LASSO, SCAD, MCP 방법으로 변수를 선택하는 경우 모형의 성능 및 학습 완료 시간에 어떠한 영향을 주는지 탐색적인 방법과 통계적 검정을 통하여 평가하였다. 마지막으로 4장에서 요약 및 결론을 제시하였다. 부록 A.1에는 고차원 자료 중 하나인 Leukemia 데이터셋으로 작업한 R 코드를 첨부하였고 연구에 사용된 전체 R 코드는 GitHub 저장소 (https://github.com/Lepidemic/The-Effect-of-Variable-Selection-on-Deep-Learning-Regression-and-Classification-Problem.git)에 올려두었다.

제 2장 딥러닝 회귀 및 분류 문제에서 변수 선택의 영향 - 소규모 및 중규모 자료의 경우

본 장에서는 회귀 및 분류 문제를 다루는 소규모 및 중규모 자료를 이용하여 심층 신경망 모형을 학습할 때 변수선택 혹은 벌점회귀 방법이 모형의 성능과 학습 완료 시간에 어떠한 영향을 미치는지 다룬다.

본 연구에서 사용된 소규모 및 중규모 자료는 네 종류로 표 2.1에 제시하였다.

표 2.1 소규모 및 중규모 자료 구성 및 출처

자료 이름	목적	입력변수 개수	인스턴스 개수	출처
Boston	회귀	13	506	MASS package in R
Turbine	회귀	10	7,411	Kaya 등(2019)
Pima	분류	6	724	mlbench package in R
Adult	분류	14	30,162	Dua과 Graff(2017)

2.1 Boston dataset

Boston 데이터셋은 14개의 변수와 506개의 인스턴스로 구성되어 있으며, 13개의 입력변수를 통하여 본인 소유의 주택가격을 예측하는 회귀문제를 다룬다. Boston 데이터셋의 변수들에 대한 설명은 그림 2.1.1에 제시하였다.

	변수	설명
입력변수	crim	자치시(town) 별 1인당 범죄발생률
	zn	25,000 ft^2 를 초과하는 거주지역의 비율
	Indus	비소매상업지역이 점유하고 있는 토지의 비율
	chas	찰스강과 접한 지역은 1, 아니면 0인 더미변수
	nox	10ppm 당 농축 일산화질소
	rm	주택 1가구당 평균 방의 개수
	age	소유자 주거지 중 1940년 이전에 지어진 집들의 비율
	dis	보스턴의 5대 고용중심으로부터의 가중 평균 거리
	rad	도시 순환 고속도로까지의 접근성 지수
	tax	10,000 달러 당 주택 재산세율
	ptratio	자치시(town)별 학생-교사 비율
	black	자치시(town)별 흑인 인구 비율
	lstat	저소득 주민들의 비율
출력변수	medv	본인 소유의 주택가격(중앙값) 단위 : $1,000

그림 2.1.1 (Boston 데이터셋) 변수설명

단계적 변수선택 방법을 이용하여 변수를 선택하면 총 11개의 변수 crim, zn, nox, rm, dis, rad, tax, ptratio, black, lstat가 선택되고 10-겹 교차검증 (cross validation)을 이용한 LASSO 벌점회귀를 통하여 변수를 선택하면 총 9개의 변수 crim, zn, chas, nox, rm, dis, ptratio, black, lstat가 선택된다.

심층 신경망 학습을 위한 초매개변수 (hyperparameter) 설정은 표 2.1.1과 같다.

표 2.1.1 (Boston 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Linear Regression
은닉노드 (Hidden node) 수	5
배치 크기 (Batch Size)	20
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	100
활성화 함수 (Activation Function)	sigmoid

입력변수와 인스턴스의 개수가 적기 때문에 은닉노드의 수를 5개로 설정하였고, 또한 2개의 은닉층을 사용했을 때보다 은닉층을 1개만 사용했을 때 성능이 더 좋음을 확인하여 은닉층은 1개로 설정하였다. Boston 데이터셋 뿐만 아니라 나머지 소규모 및 중규모 데이터셋에서도 같은 결과를 보여 분석의 일관성과 결과의 건실성 (robustness) 등을 유지하기 위하여 은닉층의 개수는 모두 1개로 고정하였다. 가중치 초기값은 -0.01과 0.01 사이의 균일분포에서 추출하였으며 의사난수 시드 값을 동일하게 주었다. 최적화기는 확률적 경사 하강법 (stochastic gradient descent)을 주었고, 회귀문제이므로 출력노드는 값을 받은 그대로 내보낼 수 있도록 설정하였다. 은닉노드의 수는 5개, 배치 크기는 20개, 학습률은 작은 값으로 0.001, 모멘텀은 큰 값으로 0.9를 설정 하였으며 학습 반복 수는 100으로 활성화 함수는 시그모이드 (sigmoid)로 설정하였다.

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를

모두 이용하는 경우와 변수선택 및 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. 그림 2.1.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, 단계적 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 RMSE 분포를 바이올린 그림으로 나타냈다. 변수선택에 대한 모의실험의 경우 반복회수를 100번으로 하는 것이 통상적 (Fan과 Li, 2001)임을 참고하여 본 연구에서도 반복회수를 100번으로 설정하였다. 바이올린 그림의 삼각형은 RMSE 값들의 평균, 원은 RMSE 값들의 중앙값을 나타낸다.

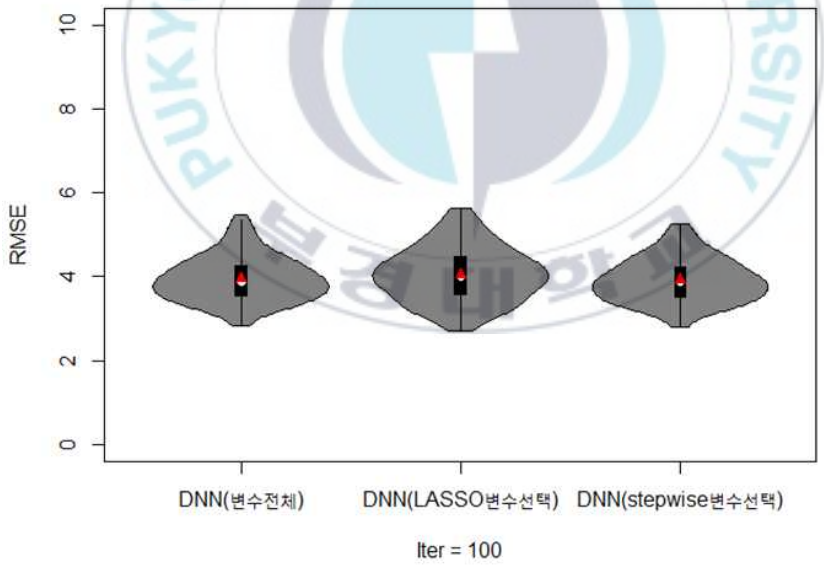

그림 2.1.2 (Boston 데이터셋) 모형별 RMSE 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 구한 모형들 사이의 RMSE 값들에 통계적으로

유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 (significance level) 5%에서 수행한 결과 p-값이 각각 0.212, 0.280이 나왔다. 따라서 3가지 모형들 사이의 RMSE 값에는 차이가 없음을 알 수 있다.

다음으로 그림 2.1.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, 단계적 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

그림 2.1.3 (Boston 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

모형들 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 각각 0.004, 0.003×10^{-1}이 나왔다. 따라서 3가지 모형들 사이의 학습 완료 시간 (단위:초)은 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 2.1.2와 같았다.

표 2.1.2 (Boston 데이터셋) 학습 완료 시간 (단위:초)에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 학습 완료 시간 (단위:초)	집단
DNN (LASSO 변수선택)	3.402	a
DNN (Stepwise 변수선택)	3.339	b
DNN (변수전체)	3.332	b

Boston 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 단계적 변수선택 방법, LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 RMSE 값 즉, 성능에는 통계적으로 유의한 차이가 없으나 학습 완료 시간에는 차이가 있었다. LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 다른 방법들에 비하여 평균적으로 0.07초 느리게 학습되었다.

2.2 Turbine dataset

Turbine 데이터셋은 11개의 변수와 7,411개의 인스턴스로 구성되어 있으며, 10개의 입력변수를 통하여 터빈 에너지 산출량 (단위:MWH)을 예측하는 회귀문제를 다룬다. Turbine 데이터셋의 변수들에 대한 설명은 그림 2.2.1에 제시하였다.

	변수	설명
입력변수	AT	Ambient temperature (주변 온도)
	AP	Ambient pressure (주변 압력) (mbar)
	AH	Ambient humidity (주변 습도) (%)
	AFDP	Air filter difference pressure (공기 필터 차압) (mbar)
	GTEP	Gas turbine exhaust pressure (가스터빈 배기압) (mbar)
	TIT	Turbine inlet temperature (터빈 입구 온도) (℃)
	TAT	Turbine after temperature (터빈 후 온도) (℃)
	CDP	Compressor discharge pressure (컴프레서 배출 압력) (mbar)
	CO	Carbon monoxide (일산화탄소) (mg/m3)
	NOx	Nitrogen oxides (질소산화물) (mg/m3)
출력변수	TEY	Turbine energy yield (터빈 에너지 산출량) (MWH)

그림 2.2.1 (Turbine 데이터셋) 변수설명

단계적 변수선택 방법을 이용하여 변수를 선택하면 모든 변수가 선택되고 10-겹 교차검증 (cross validation)을 이용한 LASSO 벌점회귀를 통하여 변수를 선택하면 CO (carbon monoxide) 변수를 제외한 총 9개의 변수가 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 2.2.1과 같다.

표 2.2.1 (Turbine 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Linear Regression
은닉노드 (Hidden node) 수	5
배치 크기 (Batch Size)	20
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	100
활성화 함수 (Activation Function)	sigmoid

Turbine 데이테셋으로 심층 신경망 학습을 위한 초매개변수 설정은 2.1절의 Boston 데이터셋에서 설정한 값과 동일하게 하였다.

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를 모두 이용하는 경우와 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. Turbine 데이터셋에서 단계적 변수선택 방법을 이용하여 변수를 선택하였을 때 모든 변수가 선택되었으므로 제외하였다. 그림 2.2.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 RMSE 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 RMSE 값들의 평균, 원은 RMSE 값들의 중앙값을 나타낸다.

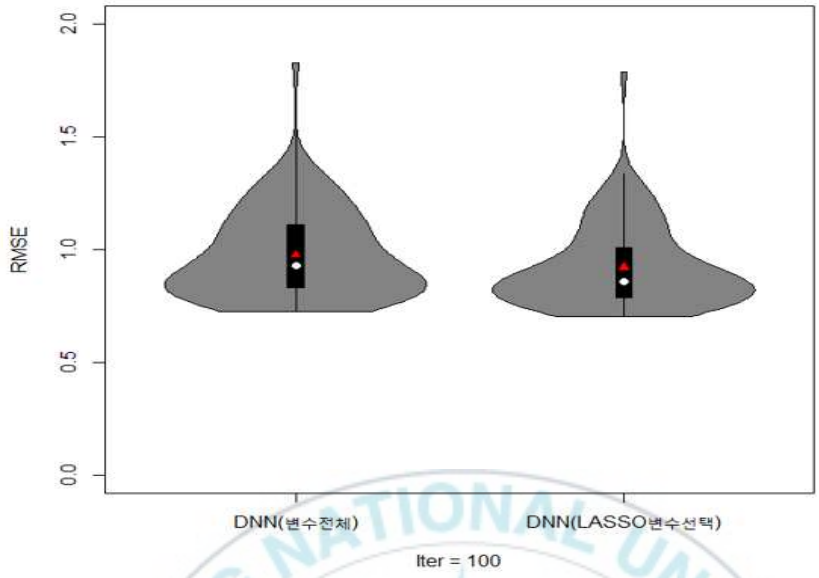

그림 2.2.2 (Turbine 데이터셋) 모형별 RMSE 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 두 모형 사이의 RMSE 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 t-test를 유의 수준 5%에서 수행한 결과 p-값이 0.037으로 LASSO 벌점회귀를 통하여 변수선택 후 학습한 심층 신경망 모형이 더 좋은 성능을 보였다.

다음으로 그림 2.2.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위: 초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

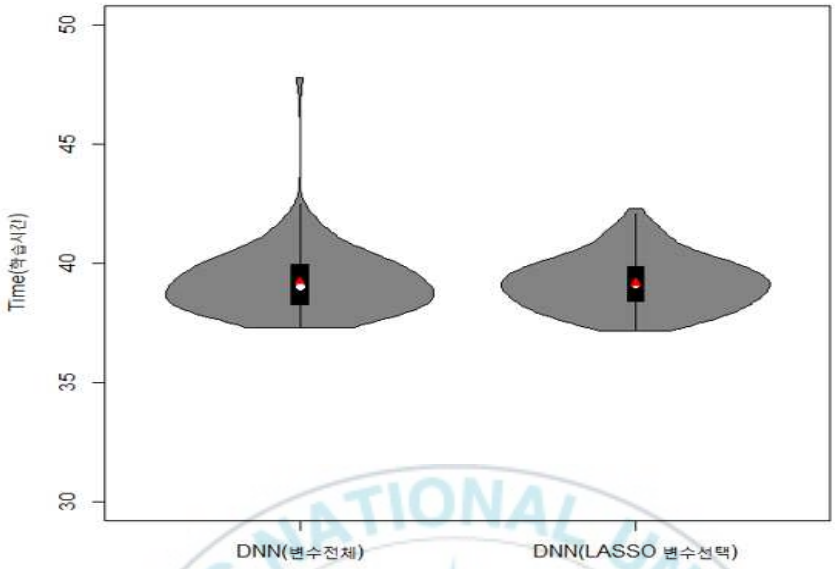

그림 2.2.3 (Turbine 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

두 모형 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 t-test를 유의수준 5%에서 수행한 결과 p-값이 0.734으로 두 심층 신경망 모형 사이의 학습 완료 시간 (단위:초)에는 차이가 없었다.

Turbine 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 단계적 변수선택 방법 및 LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 학습 완료 시간에는 통계적으로 유의한 차이가 없으나. 단계적 변수선택 방법 및 LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 전체 변수를 사용하여 학습한 모형보다 성능이 좋았다.

2.3 Pima dataset

Pima 데이터셋은 9개의 변수와 768개의 인스턴스로 구성되어 있으며, 8개의 입력변수를 통하여 당뇨병 발병 여부를 분류하는 분류 문제를 다룬다. Pima 데이터셋의 변수들에 대한 설명은 그림 2.1.1에 제시하였다.

	변수	설명
입력변수	pregnant	임신 횟수
	glucose	글루코스 내성(glucose tolerance) 실험 후 혈당수치
	pressure	확장기 혈압 (단위 : mmHg)
	triceps	상완 삼두근 피부 두께 (단위 : mm)
	insulin	혈액내 인슐린 수치 (단위 : mu U/ml)
	mass	BMI(비만도) 수치
	pedigree	당뇨병 가족력
	age	나이 (단위 : year)
출력변수	diabetes	당뇨병 여부 (0 or 1, 1 : 발병)

그림 2.3.1 (Pima 데이터셋) 변수설명

Pima 데이터셋에서 insulin 변수는 374개의 결측치 (missing value)를 포함하고 있고 triceps 변수는 227개의 결측치를 포함하고 있어서 심층 신경망 학습 전 입력변수에서 제외하였다. 따라서 입력변수의 개수가 8개에서 6개로 줄었다.

단계적 방법과 10-겹 교차검증을 이용한 LASSO 벌점회귀를 이용하여 변수를 선택하면 총 5개의 변수 pregnant, glucose, mass, pedigree,

age가 동일하게 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 2.3.1과 같다.

표 2.3.1 (Pima 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Softmax
은닉노드 (Hidden node) 수	5
배치 크기 (Batch Size)	10
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	100
활성화 함수 (Activation Function)	sigmoid

2.1절과 2.2절의 Boston, Turbine 데이터셋은 회귀문제를 다루어 출력노드를 받은 값 그대로 내보냈지만 Pima 데이터셋은 분류 문제를 다루기 때문에 출력노드에 입력받은 값들을 출력 값으로 내보내기 전 0과 1사이의 값들로 정규화하며 총합은 1이 되는 특성을 가진 소프트 맥스(softmax) 함수를 주었다.

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를 모두 이용하는 경우와 변수선택 방법과 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. Pima 데이터셋에서 단계적 변수선택 방법을 이용하여 선택한 변수와 LASSO 벌점회귀를 통해 선택된 변수가 같았기 때문에 총 세 종류의 모형이 아닌 두 종류의 모형이 만들어진다. 그림 2.3.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 단계적 변수선택 방법 및 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습

한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 정확도와 AUC 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 정확도와 AUC 값들의 평균, 원은 정확도와 AUC 값들의 중앙값을 나타낸다.

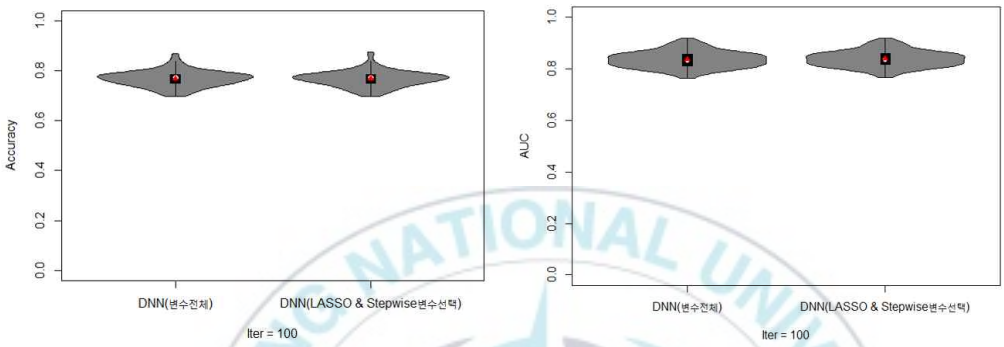

그림 2.3.2 (Pima 데이터셋) 모형별 정확도와 AUC 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 두 모형 사이의 정확도와 AUC 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 t-test를 유의수준 5%에서 수행한 결과 p-값이 정확도에 대해선 0.876, AUC에 대해선 0.542으로 두 모형 사이의 정확도와 AUC 값에는 차이가 없음을 알 수 있다.

다음으로 그림 2.3.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 단계적 변수선택 방법 및 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

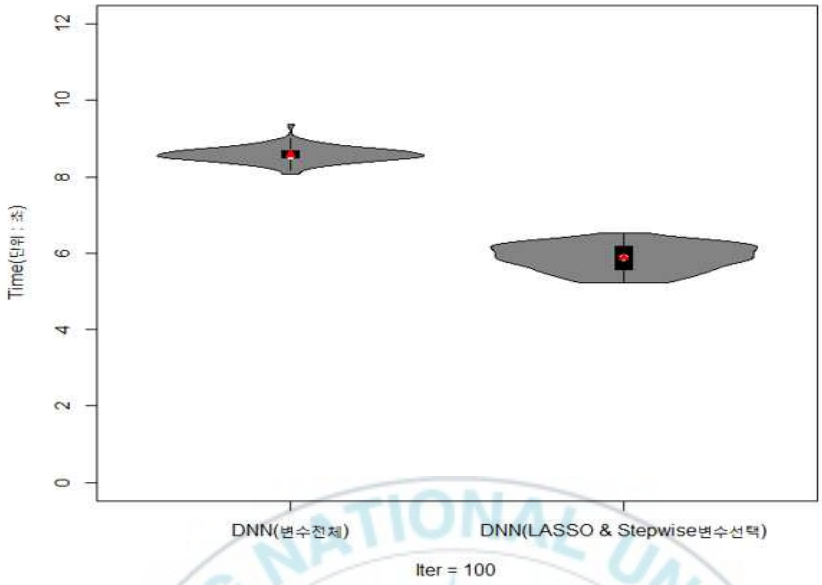

그림 2.3.3 (Pima 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

두 모형 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 t-test를 유의수준 5%에서 수행한 결과 p-값이 0.022×10^{-14}보다 작은 값이 나왔다. 따라서 단계적 변수선택 방법 및 LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 변수 전체를 사용하여 학습하는 것보다 비하여 평균적으로 2.7초 빠르게 학습되었다.

Pima 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 단계적 변수선택 방법 및 LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 정확도와 AUC 값 즉, 성능에는 통계적으로 유의한 차이가 없으나, 단계적 변수선택 방법 및 LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 전체 변수를 사용하여 학습하는 모형보다 빠르게 학습되었다.

2.4 Adult dataset

Adult 데이터셋은 15개의 변수와 32,561개의 인스턴스로 구성되어 있으며, 14개의 입력변수를 통하여 연 소득 분위가 50,000달러를 넘는지를 분류하는 분류문제를 다룬다. Adult 데이터셋의 변수들에 대한 설명은 그림 2.4.1에 제시하였다.

	변수	설명
입력변수	age	나이
	workclass	고용형태 (8가지 범주)
	fnlwgt	예상가중치
	education	교육수준 (순위형)
	education-num	교육수준 (수치형)
	marital_status	결혼상태 (7가지 범주)
	occupation	직업 (14가지 범주)
	relationship	관계 (부인, 자녀, 남편, 비가족, 기타, 비혼)
	race	인종 (5가지 범주)
	sex	성별 (남성, 여성)
	capital_gain	자본이익 기록 (수치형)
	capital_loss	자본손실 기록 (수치형)
	hours_per_week	주당 노동시간 (수치형)
	native_country	모국 (41가지 범주)
출력변수	wage	연소득 분위 ($> 50K, \leq 50K$)

그림 2.4.1 (Adult 데이터셋) 변수설명

Adult 데이터셋에서 workclass, occupation, native_country 변수에 각각 1,836, 1,843, 583개의 결측치를 포함하고 있어서 심층 신경망 학습 전 해당 인스턴스만 제외하고 변수는 제거하지 않았다. 따라서 인스턴스의 개수가 32,561개에서 30,162개로 줄었다.

단계적 방법을 이용하여 변수를 선택하면 education_num 변수를 제외한 총 13개의 변수가 선택되고 10-겹 교차검증 (cross validation)을 이용한 LASSO 벌점회귀를 통하여 변수를 선택하면 총 10개의 변수 age, workclass, education_num, marital_status, relationship, race, sex, capital_gain, capital_loss, hours_per_week가 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 2.4.1과 같다.

표 2.4.1 (Adult 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Softmax
은닉노드 (Hidden node) 수	5
배치 크기 (Batch Size)	20
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	100
활성화 함수 (Activation Function)	sigmoid

Adult 데이터셋 또한 분류 문제를 다루기 때문에 출력노드에 소프트 맥스 함수를 주어 출력결과를 내보내었다.

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를 모두 이용하는 경우와 변수선택 및 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. 그림 2.4.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, 단계적 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을

사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 정확도와 AUC 값들의 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 정확도와 AUC 값들의 평균, 원은 정확도와 AUC 값들의 중앙값을 나타낸다.

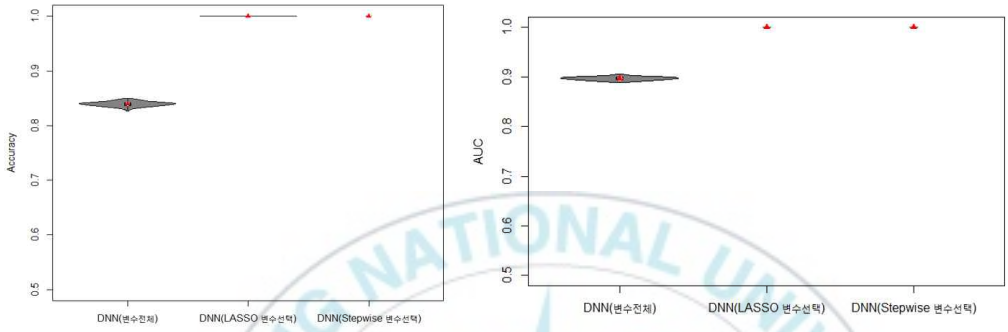

그림 2.4.2 (Adult 데이터셋) 모형별 정확도와 AUC 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 구한 모형들 사이의 정확도와 AUC 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 정확도에 대해서는 2×10^{-16}, 0.022×10^{-14}보다 작은 값이 나왔고, AUC에 대해서도 2×10^{-16}, 0.022×10^{-14}보다 작은 값이 나왔다. 따라서 3가지 모형들 사이의 성능에 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 2.4.2와 표 2.4.3과 같았다.

표 2.4.2 (Adult 데이터셋) 정확도에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 정확도	집단
DNN (LASSO 변수선택)	0.999	a
DNN (stepwise 변수선택)	1.000	a
DNN (변수전체)	0.840	b

표 2.4.3 (Adult 데이터셋) AUC에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 AUC	집단
DNN (LASSO 변수선택)	1.000	a
DNN (stepwise 변수선택)	1.000	a
DNN (변수전체)	0.897	b

다음으로 그림 2.4.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, 단계적 변수선택 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

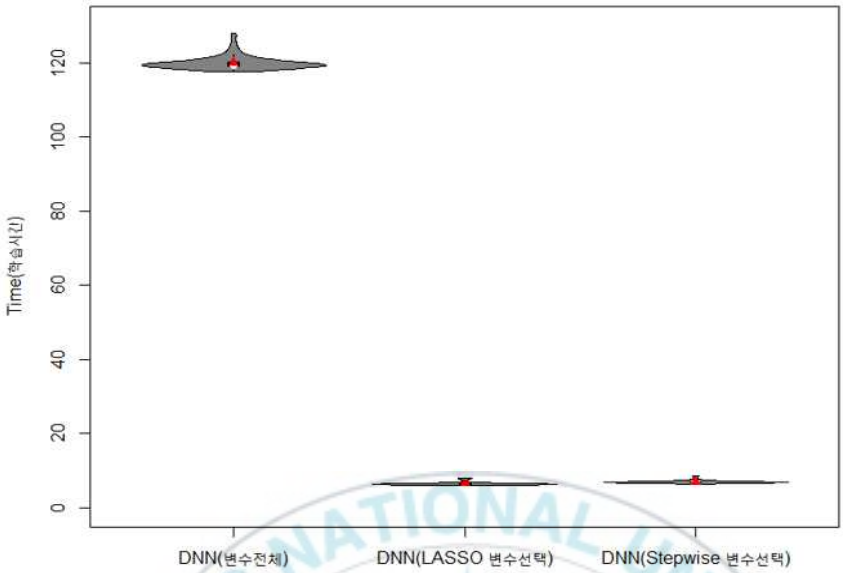

그림 2.4.3 (Adult 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

모형들 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 각각 0.022×10^{-14}, 0.022×10^{-14}보다 작은 값이 나왔다. 따라서 3가지 모형들 사이의 학습 완료 시간 (단위:초)은 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 2.4.4와 같았다.

표 2.4.4 (Adult 데이터셋) 학습 완료 시간 (단위:초)에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 학습 완료 시간 (단위:초)	집단
DNN (변수전체)	120.02	a
DNN (stepwise 변수선택)	6.853	b
DNN (LASSO 변수선택)	6.348	c

Adult 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 단계적 변수선택 방법, LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 정확도와 AUC 값 즉, 성능에는 통계적으로 유의한 차이가 있었고, 변수를 선택하여 학습하였을 때 더 좋은 성능을 보여주었다. 학습 완료 시간 또한 모형 사이에 차이가 있었다. LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우가 평균적으로 가장 빠르고 (6.348초), 단계적 변수선택 방법으로 선택된 변수들로 학습하는 경우 (6.853초)가 다음으로 빨랐다. 전체 변수를 이용하여 학습하는 경우 앞의 모형들에 비해 평균적으로 18배 느리게 (120.02초) 학습되었다.

제 3장 딥러닝 회귀 및 분류 문제에서 변수 선택의 영향 - 고차원 자료의 경우

본 장에서는 분류 문제를 다루는 고차원 자료를 이용하여 심층 신경망 모형을 SIS 방법에서 SCAD, MCP, LASSO 벌점회귀로 선택된 변수들로 학습할 때 모형의 성능과 학습 완료 시간에 어떠한 영향을 미치는지 다룬다.

본 연구에서 사용된 고차원 자료는 네 종류로 표 3.1에 제시하였다.

표 3.1 고차원 자료 구성 및 출처

자료 이름	목적	입력변수 개수	인스턴스 개수	출처
Leukemia	분류	7,129	72	Golub 등(1999)
Prostate	분류	12,600	136	Singh 등(2002)
Colon	분류	2,000	62	Alon 등(1999)
Breast cancer	분류	22,215	188	Neve 등(2006)

3.1 Leukemia dataset

Leukemia 데이터셋은 급성 백혈병 환자 72명으로부터 받은 7,129개의 유전자 발현 데이터로 구성되어 있다. 즉, 7,129개의 유전자 발현 수치변수와 72개의 인스턴스로 구성되어 있으며, 7,129개의 입력변수를 통하여 급성 림프절성 백혈병 환자와 급성 골수성 백혈병 환자를 분류하는 분류

문제를 다룬다. Leukemia 데이터셋의 변수들에 대한 설명은 그림 3.1.1에 제시하였다.

	변수	설명
입력변수	V1 ~ V7129	7,129개의 유전자 발현 데이터
출력변수	V7130	0 : 급성 림프절성 백혈병, 1 : 급성 골수성 백혈병

그림 3.1.1 (Leukemia 데이터셋) 변수설명

SIS 방법에서 SCAD 벌점회귀를 이용하면 총 4개의 변수 V1144, V2597, V4196, V4847가 선택되고 MCP 벌점회귀를 이용하면 총 2개의 변수 V1144, V4847가 선택되며 LASSO 벌점회귀를 이용하면 총 4개의 변수 V1144, V2684, V4847, V6855가 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 3.1.1과 같다.

표 3.1.1 (Leukemia 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Softmax
은닉노드 (Hidden node) 수	20
배치 크기 (Batch Size)	5
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	150 ~ 400
활성화 함수 (Activation Function)	sigmoid

3장에서 다루는 고차원 자료를 이용하여 심층 신경망 학습을 위한 초매개변수 설정은 2장에서 다루었던 소규모 및 중규모 자료에서 사용했던 초매개변수 설정과 2가지 차이점이 있다. 첫 번째로 입력변수의 개수가 고차원 자료에서 크게 증가하였기 때문에 은닉노드의 수를 5개에서 20개로 늘렸고, 두 번째로는 인스턴스의 개수가 배치학습을 하기에 너무 적어 배치크기를 20에서 5로 줄였다.

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를 모두 이용하는 경우와 SIS 방법에서 3종류의 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. 그림 3.1.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 SIS 방법에서 SCAD 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 정확도와 AUC 값들의 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 정확도와 AUC 값들의 평균, 원은 정확도와 AUC 값들의 중앙값을 나타낸다.

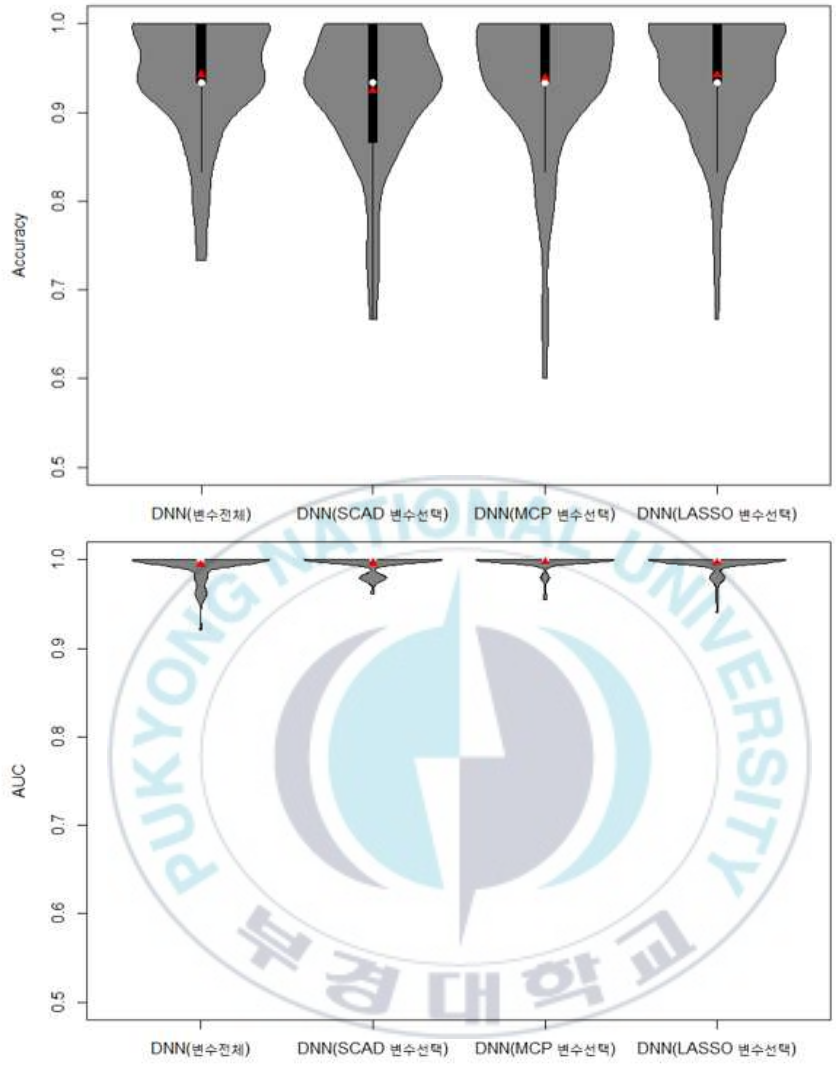

그림 3.1.2 (Leukemia 데이터셋) 모형별 정확도와 AUC 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 구한 모형들 사이의 정확도와 AUC 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 정

확도에 대해서는 0.259, 0.131, AUC에 대해서는 0.256, 0.230이 나왔다. 따라서 4가지 모형들 사이의 성능에 통계적으로 유의한 차이가 없음을 알 수 있다.

다음으로 그림 3.1.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, SIS 방법에서 SCAD 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

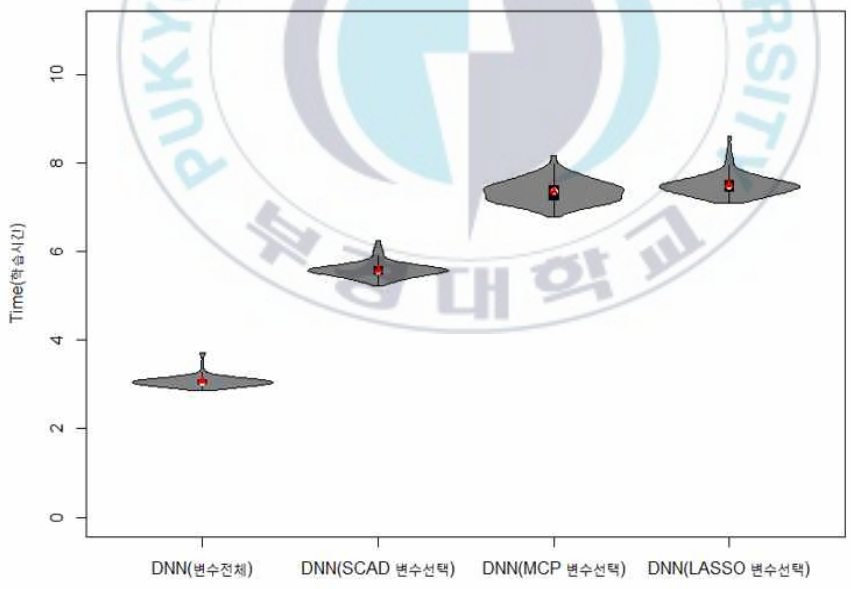

그림 3.1.3 (Leukemia 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

모형들 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 각각 2×10^{-16}, 0.022×10^{-14}보다 작은 값이 나왔다. 따라서 4가지 모형들 사이의 학습 완료 시간 (단위:초)은 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.1.2와 같았다.

표 3.1.2 (Leukemia 데이터셋) 학습 완료 시간 (단위:초)에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 학습 완료 시간 (단위:초)	집단
DNN (LASSO 변수선택)	7.509	a
DNN (MCP 변수선택)	7.365	b
DNN (SCAD 변수선택)	5.592	c
DNN (변수전체)	3.068	d

Leukemia 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 SIS 방법에서 SCAD, MCP, LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 정확도와 AUC 값 즉, 성능에는 통계적으로 유의한 차이가 없었다. 그러나 학습 완료 시간은 모형 사이에 차이가 있었다. 전체 변수를 이용하여 학습하는 경우가 평균적으로 가장 빠르고 (3.068초), SCAD 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 (5.592초)가 다음으로 빨랐으며 MCP 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 (7.365초)가 뒤를 잇고 LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 다른 모형에 비해 평균적으로 가장 느리게 (7.509초) 학습되었다.

3.2 Prostate dataset

Prostate 데이터셋은 전립선종양 환자 77명과 정상 환자 59명으로부터 받은 12,600개의 유전자 발현 데이터로 구성되어 있다. 즉, 12,600개의 유전자 발현 수치변수와 136개의 인스턴스로 구성되어 있으며, 12,600개의 입력변수를 통하여 전립선종양 환자와 정상 환자를 분류하는 분류 문제를 다룬다. Prostate 데이터셋의 변수들에 대한 설명은 그림 3.2.1에 제시하였다.

	변수	설명
입력변수	V1 ~ V12600	12,600개의 유전자 발현 데이터
출력변수	V12601	0 : 정상, 1 : 전립선종양

그림 3.2.1 (Prostate 데이터셋) 변수설명

SIS 방법에서 SCAD 벌점회귀를 이용하면 총 4개의 변수 V4483, V10431, V11052, V11200가 선택되고 MCP 벌점회귀를 이용하면 총 2개의 변수 V4483, V11052가 선택되며 LASSO 벌점회귀를 이용하면 총 6개의 변수 V4483, V6151, V8610, V10431, V11052, V11200가 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 3.2.1과 같다.

표 3.2.1 (Prostate 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Softmax
은닉노드 (Hidden node) 수	20
배치 크기 (Batch Size)	5
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	100 ~ 600
활성화 함수 (Activation Function)	sigmoid

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를 모두 이용하는 경우와 SIS 방법에서 3종류의 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. 그림 3.2.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 SIS 방법에서 SCAD 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 정확도와 AUC 값들의 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 정확도와 AUC 값들의 평균, 원은 정확도와 AUC 값들의 중앙값을 나타낸다.

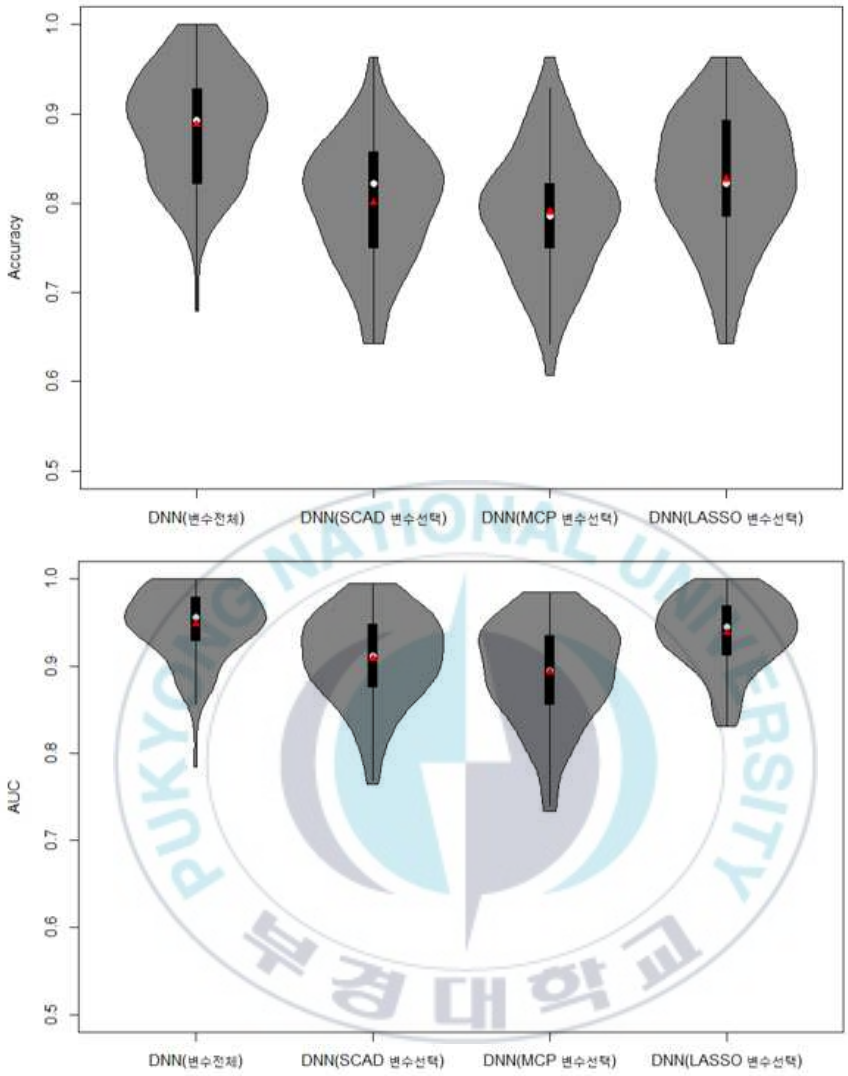

그림 3.2.2 (Prostate 데이터셋) 모형별 정확도와 AUC 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 구한 모형들 사이의 정확도와 AUC 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 정

확도에 대해서는 2×10^{-16}, 0.022×10^{-14}보다 작은 값이 나왔고, AUC에 대해서는 2.47×10^{-16}, 7.59×10^{-15}가 나왔다. 따라서 4가지 모형들 사이의 성능에 통계적으로 유의한 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.2.2, 표 3.2.3, 표 3.2.4와 같았다.

표 3.2.2 (Prostate 데이터셋) 정확도에 대한 Scheffe 다중비교 결과

심층 신경망 모형	평균 정확도	집단
DNN (변수전체)	0.876	a
DNN (LASSO 변수선택)	0.834	b
DNN (SCAD 변수선택)	0.807	bc
DNN (MCP 변수선택)	0.784	c

표 3.2.3 (Prostate 데이터셋) 정확도에 대한 Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 정확도	집단
DNN (변수전체)	0.876	a
DNN (LASSO 변수선택)	0.834	b
DNN (SCAD 변수선택)	0.807	c
DNN (MCP 변수선택)	0.784	c

표 3.2.4 (Prostate 데이터셋) AUC에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 AUC	집단
DNN (변수전체)	0.948	a
DNN (LASSO 변수선택)	0.938	a
DNN (SCAD 변수선택)	0.908	b
DNN (MCP 변수선택)	0.893	b

다음으로 그림 3.2.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, SIS 방법에서 SCAD 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

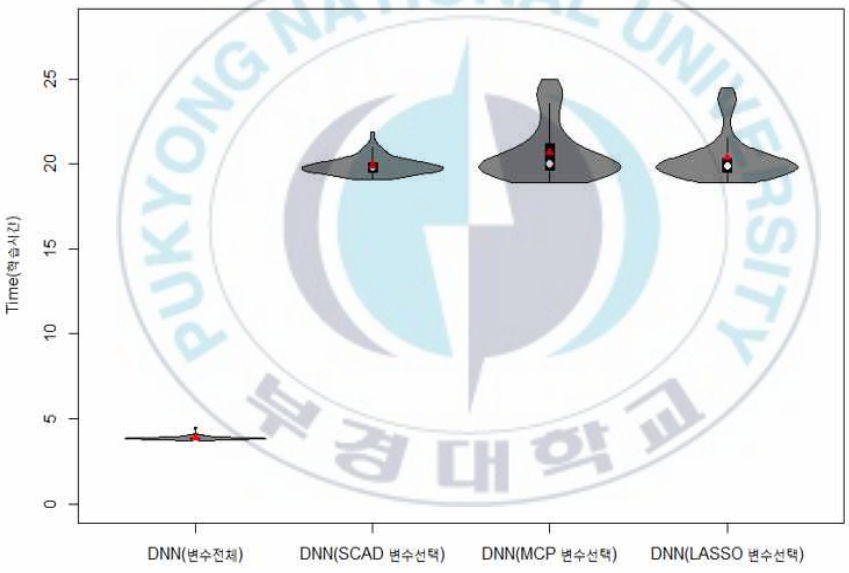

그림 3.2.3 (Prostate 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

모형들 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의

수준 5%에서 수행한 결과 p-값이 각각 2×10^{-16}, 0.022×10^{-14}보다 작은 값이 나왔다. 따라서 4가지 모형들 사이의 학습 완료 시간 (단위:초)은 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.2.5와 같았다.

표 3.2.5 (Prostate 데이터셋) 학습 완료 시간 (단위:초)에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	학습 완료 시간 (단위:초)	집단
DNN (MCP 변수선택)	20.67	a
DNN (LASSO 변수선택)	20.36	a
DNN (SCAD 변수선택)	19.89	b
DNN (변수전체)	3.854	c

Prostate 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우 SIS 방법에서 SCAD, MCP, LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우와 비교했을 때 정확도와 AUC 값 즉, 성능이 더 좋음을 확인하였다. 학습 완료 시간 또한 입력변수를 모두 사용하여 학습하였을 때 다른 모형들에 비해 평균적으로 5배 빠르게 학습함을 확인하였다.

3.3 Colon dataset

Colon 데이터셋은 대장암 환자 40명과 건강한 정상 환자 22명으로부터 받은 2,000개의 유전자 발현 데이터로 구성되어 있다. 즉, 2,000개의 유전자 발현 수치변수와 62개의 인스턴스로 구성되어 있으며, 2,000개의 입

력변수를 통하여 대장암 환자와 건강한 정상 환자를 분류하는 분류 문제를 다룬다. Colon 데이터셋의 변수들에 대한 설명은 그림 3.3.1에 제시하였다.

	변수	설명
입력변수	V1 ~ V2000	2,000개의 유전자 발현 데이터
출력변수	V2001	0 : 정상 환자, 1 : 대장암 환자

그림 3.3.1 (Colon 데이터셋) 변수설명

SIS 방법에서 SCAD와 MCP 벌점회귀를 이용하면 총 3개의 변수 V249, V1895, V1935가 선택되고 LASSO 벌점회귀를 이용하면 총 3개의 변수 V117, V249, V765가 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 3.3.1과 같다.

표 3.3.1 (Colon 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Softmax
은닉노드 (Hidden node) 수	20
배치 크기 (Batch Size)	5
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	150 ~ 500
활성화 함수 (Activation Function)	sigmoid

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를

모두 이용하는 경우와 SIS 방법에서 3종류의 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. 그림 3.3.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 SIS 방법에서 SCAD과 MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 정확도와 AUC 값들의 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 정확도와 AUC 값들의 평균, 원은 정확도와 AUC 값들의 중앙값을 나타낸다.

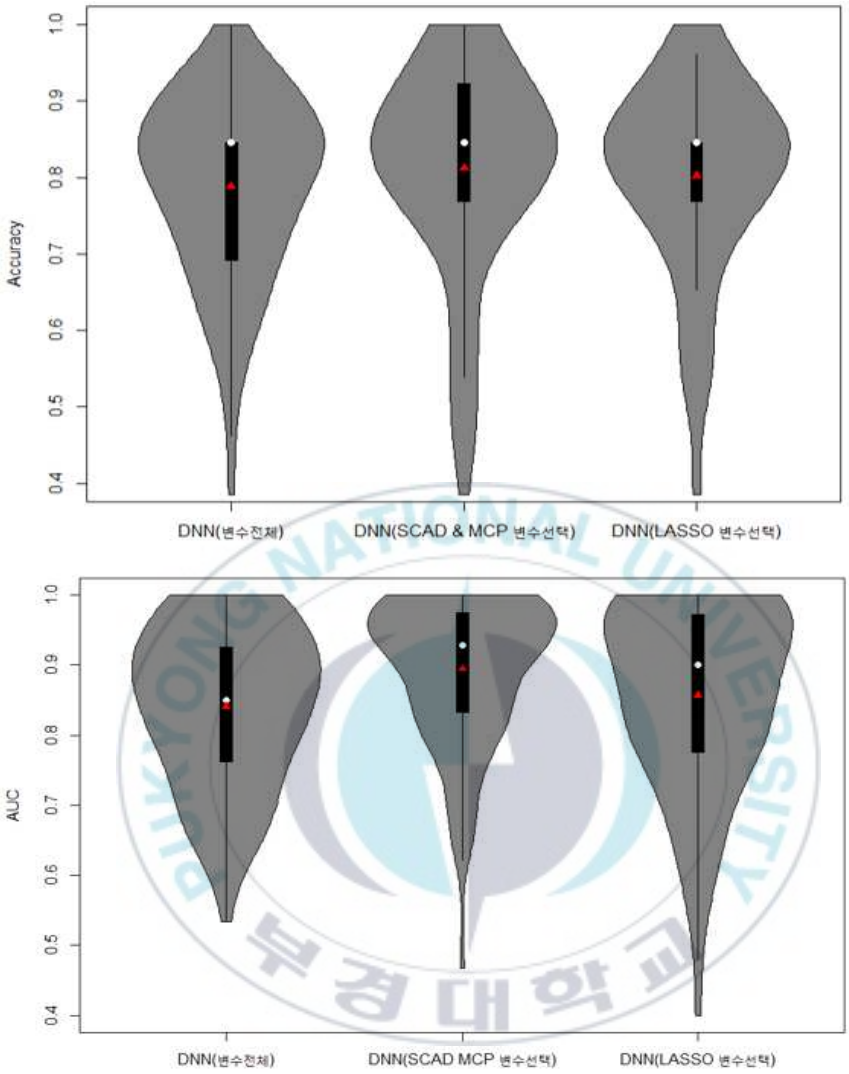

그림 3.3.2 (Colon 데이터셋) 모형별 정확도와 AUC 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 구한 모형들 사이의 정확도와 AUC 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 정

확도에 대해서는 0.415, 0.201, AUC에 대해서는 0.005, 0.002이 나왔다. 따라서 3가지 모형들 사이의 정확도에는 통계적으로 유의한 차이가 없지만, AUC에는 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.3.2와 같았다.

표 3.3.2 (Colon 데이터셋) AUC에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 AUC	집단
DNN (SCAD & MCP변수선택)	0.894	a
DNN (LASSO 변수선택)	0.857	ab
DNN (변수전체)	0.841	b

다음으로 그림 3.3.3은 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형, SIS 방법에서 SCAD과 MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

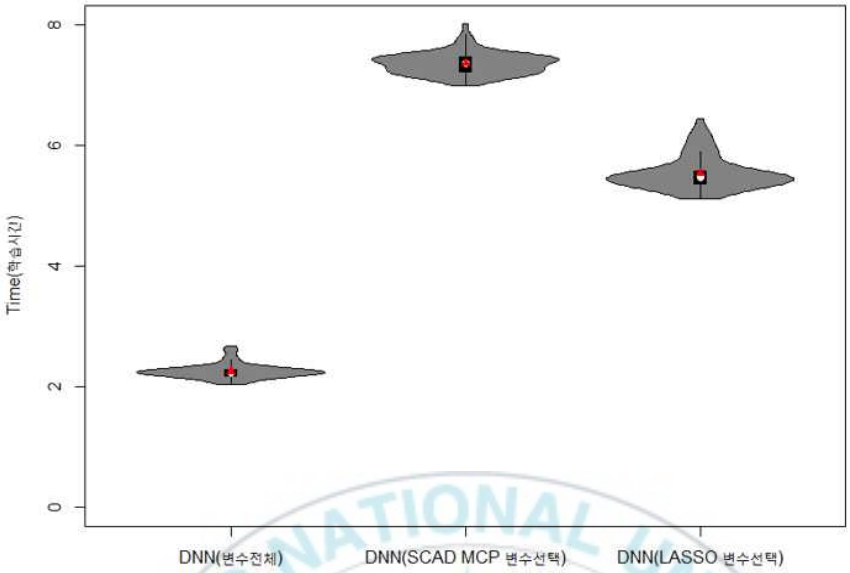

그림 3.3.3 (Colon 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

모형들 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 각각 2×10^{-16}, 0.022×10^{-14}보다 작은 값이 나왔다. 따라서 3가지 모형들 사이의 학습 완료 시간 (단위:초)은 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.3.3과 같았다.

표 3.3.3 (Colon 데이터셋) 학습 완료 시간 (단위:초)에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 학습 완료 시간 (단위:초)	집단
DNN (SCAD & MCP 변수선택)	7.361	a
DNN (LASSO 변수선택)	5.537	b
DNN (변수전체)	2.247	c

Colon 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 SIS 방법에서 SCAD, MCP, LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 정확도에는 통계적으로 유의한 차이가 없었다. 그러나 AUC 값들은 모형 사이에 차이가 있었다. SCAD와 MCP 벌점회귀를 이용하여 선택된 변수들을 사용하여 학습하였을 때 다른 모형에 비해 AUC 값이 평균적으로 4% 정도 높았다. 학습 완료 시간은 전체 변수를 이용하여 학습하는 경우가 평균적으로 가장 빠르고 (2.247초), LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 (5.537초)가 다음으로 빨랐으며 SCAD와 MCP 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 (7.361초)가 평균적으로 가장 느리게 학습되었다.

3.4 Breast cancer dataset

Breast cancer 데이터셋은 유방암 환자 75명과 정상 환자 43명으로부터 받은

22,215개의 유전자 발현 데이터로 구성되어 있다. 즉, 22,215개의 유전자 발현 수치변수와 118개의 인스턴스로 구성되어 있으며, 22,215개의 입력변수를 통하여 유방암 환자와 정상 환자를 분류하는 분류 문제를 다룬다. Breast cancer 데이터 셋의 변수들에 대한 설명은 그림 3.4.1에 제시하였다.

	변수	설명
입력변수	V1 ~ V22215	22,215개의 유전자 발현 데이터
출력변수	V22216	0 : 정상 환자, 1 : 유방암 환자

그림 3.4.1 (Breast cancer 데이터셋) 변수설명

SIS 방법에서 SCAD와 MCP 벌점회귀를 이용하면 1개의 변수 V4752가 선택되고 LASSO 벌점회귀를 이용하면 총 5개의 변수 V4752, V7404, V13543, V13795가 선택된다.

심층 신경망 학습을 위한 초매개변수 설정은 표 3.4.1과 같다.

표 3.4.1 (Breast caner 데이터셋) 심층 신경망 학습을 위한 초매개변수 설정

초매개변수	값
은닉층(Hidden layer) 수	1
가중치(Weight) 초기값	U(-0.01, 0.01)
Optimizer 종류	SGD
출력노드(Output node) 함수 종류	Softmax
은닉노드 (Hidden node) 수	20
배치 크기 (Batch Size)	5
학습률 (Learning rate)	0.001
모멘텀 (Momentum)	0.9
학습 반복 수 (Number of Iterations)	30 ~ 200
활성화 함수 (Activation Function)	sigmoid

심층 신경망 학습을 위한 초매개변수들을 설정한 후 자료의 입력변수를 모두 이용하는 경우와 SIS 방법에서 3종류의 벌점회귀를 통하여 선택된 입력변수들을 이용하는 경우에 대해 각각 심층 신경망을 학습하였다. 그림 3.4.2는 서로 다른 학습 데이터 100개를 통해 입력변수 전체를 사용하여 학습한 100개의 모형과 SIS 방법에서 SCAD과 MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형 각각에 대해 테스트 데이터 100개를 이용하여 구한 정확도와 AUC 값들의 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 정확도와 AUC 값들의 평균, 원은 정확도와 AUC 값들의 중앙값을 나타낸다.

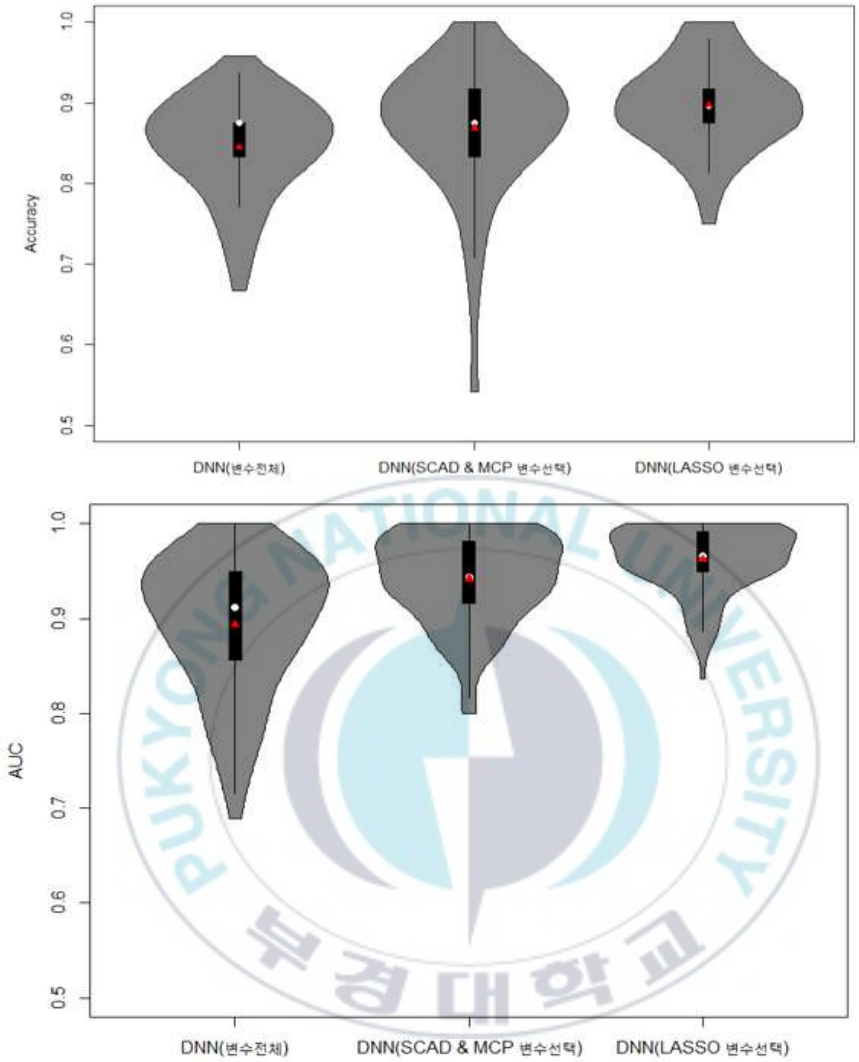

그림 3.4.2 (Breast cancer 데이터셋) 모형별 정확도와 AUC 값들에 대한 바이올린 그림

테스트 데이터를 이용하여 구한 모형들 사이의 정확도와 AUC 값들에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 정

확도에 대해서는 1.21×10^{-6}, 2.09×10^{-7}, AUC에 대해서는 2×10^{-16} 보다 작은 값과, 8.75×10^{-15}이 나왔다. 따라서 3가지 모형들 사이의 정확도와 AUC에는 통계적으로 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.4.2, 표 3.4.3, 표 3.4.4와 같았다.

표 3.4.2 (Breast cancer 데이터셋) 정확도에 대한 Scheffe 다중비교 결과

심층 신경망 모형	평균 정확도	집단
DNN (LASSO 변수선택)	0.897	a
DNN (SCAD & MCP변수선택)	0.869	b
DNN (변수전체)	0.845	b

표 3.4.3 (Breast cancer 데이터셋) 정확도에 대한 Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 정확도	집단
DNN (LASSO 변수선택)	0.897	a
DNN (SCAD & MCP변수선택)	0.869	b
DNN (변수전체)	0.845	c

표 3.4.4 (Breast cancer 데이터셋) AUC에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 AUC	집단
DNN (LASSO 변수선택)	0.962	a
DNN (SCAD & MCP변수선택)	0.942	b
DNN (변수전체)	0.893	c

다음으로 그림 3.4.3은 서로 다른 학습 데이터 100개를 통해 입력변수

전체를 사용하여 학습한 100개의 모형, SIS 방법에서 SCAD과 MCP 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형, 그리고 LASSO 벌점회귀 방법을 통하여 선택된 변수들을 사용하여 학습한 100개의 모형의 학습 완료 시간 (단위:초) 분포를 바이올린 그림으로 나타냈다. 바이올린 그림의 삼각형은 학습 완료 시간 (단위:초)의 평균, 원은 학습 완료 시간 (단위:초)의 중앙값을 나타낸다.

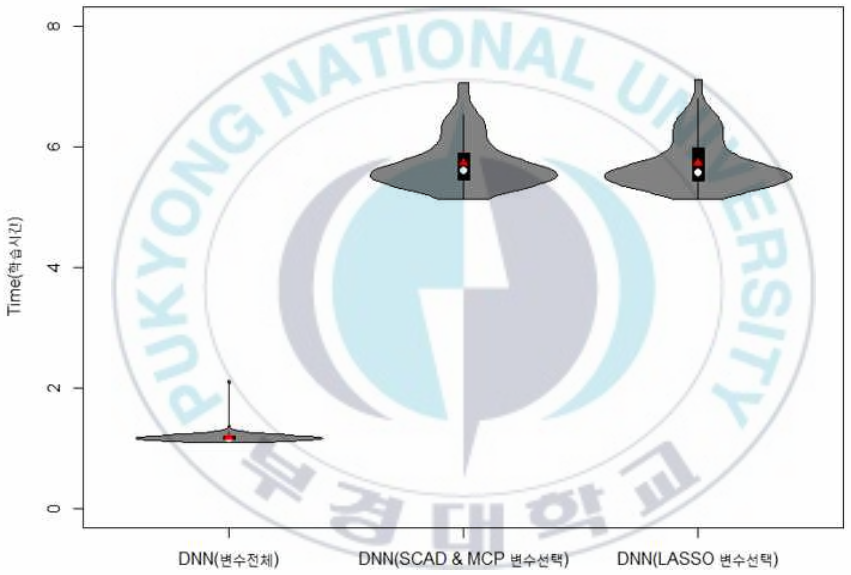

그림 3.4.3 (Breast cancer 데이터셋) 모형별 학습 완료 시간 (단위:초)에 대한 바이올린 그림

모형들 사이의 학습 완료 시간 (단위:초)에 통계적으로 유의미한 차이가 있는지 확인하기 위해 분산분석 및 Kruskal-Wallis 순위합 검정을 유의수준 5%에서 수행한 결과 p-값이 각각 2×10^{-16}, 0.022×10^{-14}보다 작은

값이 나왔다. 따라서 3가지 모형들 사이의 학습 완료 시간 (단위:초)은 차이가 있음을 알 수 있다. 사후분석으로 Scheffe, Bonferroni, Tukey 다중비교 검정을 수행한 결과 다음 표 3.4.5와 같았다.

표 3.4.5 (Breast cancer 데이터셋) 학습 완료 시간 (단위:초)에 대한 Scheffe, Bonferroni, Tukey 다중비교 결과

심층 신경망 모형	평균 학습 완료 시간 (단위:초)	집단
DNN (SCAD & MCP 변수선택)	5.734	a
DNN (LASSO 변수선택)	5.732	a
DNN (변수전체)	1.174	b

Breast cancer 데이터셋의 경우 심층 신경망 모형을 학습할 때 입력변수를 모두 사용하여 학습하는 경우와 SIS 방법에서 SCAD, MCP, LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습하는 경우를 비교했을 때 정확도와 AUC 값들에 통계적으로 유의한 차이가 있었다. LASSO 벌점회귀를 통하여 선택된 변수들을 사용하여 학습한 모형이 다른 모형에 비해 정확도가 평균적으로 3~5% 높았고, AUC 또한 평균적으로 2~7% 높았다. 그러나 학습 완료 시간은 전체 변수를 이용하여 학습하는 경우가 평균적으로 가장 빠르고 (1.174초), LASSO 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 (5.732초)와 SCAD와 MCP 벌점회귀를 통하여 선택된 변수들로 학습하는 경우 (5.734초)는 큰 차이가 없었다.

제 4장 요약 및 결론

본 장에서는 2장과 3장에서 회귀 및 분류 문제를 다루는 소규모 및 중규모 자료, 고차원 자료를 심층 신경망 모형에 입력변수를 모두 이용했을 때의 모형과 변수선택 방법 또는 벌점회귀를 통해 선택된 변수들을 입력변수로 이용한 모형 사이의 성능 및 학습 완료 시간을 종합적으로 비교하고 연구 결과에 대한 결론을 맺는다.

그림 4.1 자료 크기와 종류에 따른 모형 비교 결과

					모형 비교 결과	
자료 크기	자료 이름	인스턴스 개수	입력변수 개수	선택된 변수 개수	모형 성능	학습시간
소규모/중규모	Boston	506	13	9 ~ 11	유의한 차이없음	변수전체*
	Turbine	7,411	10	9	LASSO 변수선택*	유의한 차이없음
	Pima	724	6	5	유의한 차이없음	LASSO & Stepwise 변수선택*
	Adult	30,162	14	9 ~ 13	LASSO & Stepwise 변수선택*	LASSO & Stepwise 변수선택*
고차원	Leukemia	72	7,129	2 ~ 4	유의한 차이없음	변수전체*
	Prostate	136	12,600	2 ~ 6	변수전체*	변수전체*
	Colon	62	2,000	3	SCAD & MCP 변수선택*	변수전체*
	Breast cancer	118	22,215	1 ~ 5	LASSO 변수선택*	변수전체*

(* 의 의미 : 표에서 모형성능 또는 학습시간에 유의한 차이가 있는 경우 모형성능이 좋거나 학습시간이 짧은 방법을 게시하였음.)

그림 4.1에 4종류의 소규모 및 중규모 자료와 4종류의 고차원 자료를 대상으로 변수선택 방법과 벌점회귀를 사용할 때 심층 신경망 모형의 성능과 학습 완료 시간과 입력변수를 모두 이용하여 학습한 모형 경우와 비교한 결과를 작성하였다. 심층 신경망 모형을 이용하여 회귀 및 분류 문제를 다룰 때 변수선택이 모형의 성능 및 학습 완료 시간에 미치는 영향은 인스턴스의 규모와 자료의 구조에 따라 달라짐을 확인할 수 있었다. 고차원 자료의 경우에는 심층 신경망 모형의 성능은 자료에 따라 결과가 달라지는 반면에 학습 완료 시간은 변수선택을 통해 선택된 변수들을 사용하여 학습하

는 심층 신경망 모형과 비교하여 입력변수를 모두 이용하였을 때의 모형이 통계적으로 유의하게 빨리 학습함을 확인할 수 있다.

참고문헌

[1] Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D. and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences USA*, **96**, 6745-6750.

[2] Dua, D., and Graff, C. (2017). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml

[3] Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, **96**, 1348-1360

[4] Fan, J. and Lv, J. (2008). Sure independence screening for ultra-high dimensional feature space. *Journal of the Royal Statistical Society Series B*, **70**, 849-911.

[5] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E. S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531-537.

[6] Jang, D. H., Do Ha, I., Park, D. J., Park, I. H. and Lee, S. J. (2020). Statistical model development for high-dimensional big data analytics. *The Korean Data & Information Science Society*, **31**(6), 1009-1020.

[7] Kaya, H., T□FEKCİ, P., and Uzun, E. (2019). Predicting CO and NOx emissions from gas turbines: novel data and a benchmark PEMS. *Turkish Journal of Electrical Engineering & Computer Sciences*, **27**, 4783 - 4796.

[8] Kim, J. A. (2018). Outlier detection and variable selection via difference based regression model and penalized regression. *Journal of the Korean Data & Information Science Society*, **30**, 815-825.

[9] Saldana, D. F. and Feng, Y. (2018). SIS: an R package for sure independence screening in ultrahigh dimensional statistical models. *Journal of Statistical Software*, **83**(2), 1-25.

[10] Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Maonla, J., Ladd, C., Tamayo, P., Renshaw, A. A., D' Amico, A. V., Richie, J. P., Lander, E. S., Loda, M., Kantoff, P. W., Golub, T. R. and Sellers, W. R. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell* , **1**, 203-209.

[11] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267-288.

부록 A.1

R 코드

```
library(SIS)
library(gmodels)
library(tictoc)
library(vioplot)


data('leukemia.train', package='SIS')
dim(leukemia.train)
# dim(leukemia.train): 38x7130

data('leukemia.test', package='SIS')
dim(leukemia.test)
# dim(leukemia.test): 34x7130

leukemia_data <- rbind(leukemia.train, leukemia.test)
dim(leukemia_data)
# 72x7130


summary(leukemia_data[, 1:15])
summary(leukemia_data[, 7115:7130])


# 신경망 학습용 data set
# Train : Test = 8 : 2 나누기. (100번)
# seed 를 바꿔가며 100개의 서로 다른 train, test data set 생성.
# list() 객체에 저장.

leukemia_train_x_list_num <- list()
leukemia_train_y_list_num <- list()

leukemia_test_x_list_num <- list()
leukemia_test_y_list_num <- list()

for(i in 1:100){
  set.seed(i)
  train_index <- sample(1:nrow(leukemia_data), 0.8 * nrow(leukemia_data),
replace = FALSE)
  leukemia_train_x_num <- leukemia_data[train_index, 1:7129]
  leukemia_train_y_num <- leukemia_data[train_index, 7130]

  leukemia_test_x_num <- leukemia_data[-train_index, 1:7129]
  leukemia_test_y_num <- leukemia_data[-train_index, 7130]

  leukemia_train_x_list_num[[i]] <- leukemia_train_x_num
  leukemia_train_y_list_num[[i]] <- leukemia_train_y_num

  leukemia_test_x_list_num[[i]] <- leukemia_test_x_num
  leukemia_test_y_list_num[[i]] <- leukemia_test_y_num

}
```

```
############
## 신경망 학습 ##
############
# https://mxnet.apache.org/api/r  참고.
library(mxnet)

leukemia_model_list <- list()
leukemia_dnn_test_predict_list <- list()
leukemia_dnn_test_predict_label_list <- list()
leukemia_dnn_test_confusion_list <- list()
leukemia_dnn_test_acc_list <- list() # y_hat 과 test data set의 y 로 Accuracy 저장 list 할당.
leukemia_dnn_exectime_list <- list()
leukemia_dnn_train_logger_list <- list()
leukemia_dnn_test_logger_list <- list()
leukemia_dnn_test_auc_list <- list()

for(i in 1:100){ # 1:100
  leukemia_train_x_scale <- scale(leukemia_train_x_list_num[[i]])
  leukemia_test_x_scale <- scale(leukemia_test_x_list_num[[i]])

  leukemia_train_x_datamatrix <- data.matrix(leukemia_train_x_scale)
  leukemia_test_x_datamatrix <- data.matrix(leukemia_test_x_scale)

  leukemia_train_y <- leukemia_train_y_list_num[[i]]
  leukemia_test_y <- leukemia_test_y_list_num[[i]]

  mx.set.seed(2020)  # 가중치 초기값 고정.

  input <- mx.symbol.Variable("data")
  fc1 <- mx.symbol.FullyConnected(data = input, num.hidden = 20)  # Hyper parameter : 은닉노드 수
  act1 <- mx.symbol.Activation(data = fc1, act_type = "sigmoid")  # Hyper parameter : 활성화 함수 종류
  fc2 <- mx.symbol.FullyConnected(data = act1, num.hidden = 2)
  lro <- mx.symbol.SoftmaxOutput(data = fc2)  # Hyper parameter : 출력노드 함수 종류

  tic()

  logger <- mx.metric.logger$new()
  model <- mx.model.FeedForward.create(symbol = lro, X = leukemia_train_x_datamatrix, y = leukemia_train_y,
                                       eval.data = list(data = leukemia_test_x_datamatrix, label = leukemia_test_y),
                                       ctx = mx.gpu(), num.round = 150, optimizer = 'sgd', # Hyper parameter : iter 수, 최적화 함수 종류
                                       array.batch.size = 5, learning.rate = 0.001, momentum = 0.9, eval.metric = mx.metric.accuracy,
                                       verbose = T, epoch.end.callback = mx.callback.log.train.metric(1, logger))  # Hyper parameter : 배치 사이즈, 학습률, 모멘텀 값.

  exectime <- toc()
  exectime <- round(exectime$toc - exectime$tic, 5)

  leukemia_model_list[[i]] <- model
  leukemia_dnn_test_predict_list[[i]] <- predict(leukemia_model_list[[i]], leukemia_test_x_datamatrix)
  leukemia_dnn_test_predict_label_list[[i]] <-
```

```
max.col(t(leukemia_dnn_test_predict_list[[i]])) - 1
  leukemia_dnn_test_confusion_list[[i]] <- CrossTable(x =
leukemia_test_y_list_num[[i]], y = leukemia_dnn_test_predict_label_list[[i]])
  leukemia_dnn_test_acc_list[[i]] <- (leukemia_dnn_test_confusion_list[[i]]$t[1] +
leukemia_dnn_test_confusion_list[[i]]$t[4])  /
sum(leukemia_dnn_test_confusion_list[[i]]$t)
  auc <- performance(prediction( predict(leukemia_model_list[[i]],
leukemia_test_x_datamatrix)[2, ], leukemia_test_y_list_num[[i]], label.ordering =
c(0, 1)), measure = "auc")
  leukemia_dnn_test_auc_list[[i]] <- auc@y.values[[1]]

  leukemia_dnn_exectime_list[[i]] <- exectime
  leukemia_dnn_train_logger_list[[i]] <- logger$train
  leukemia_dnn_test_logger_list[[i]] <- logger$eval
}

leukemia_dnn_test_acc_unlist <- unlist(leukemia_dnn_test_acc_list)
leukemia_dnn_test_auc_unlist <- unlist(leukemia_dnn_test_auc_list)
leukemia_dnn_exectime_unlist <- unlist(leukemia_dnn_exectime_list)

leukemia_dnn_train_logger_unlist <- data.frame(
matrix(unlist(leukemia_dnn_train_logger_list), ncol = 100))
leukemia_dnn_test_logger_unlist <- data.frame(
matrix(unlist(leukemia_dnn_test_logger_list), ncol = 100))

# 결과저장
# write(t(leukemia_dnn_test_acc_unlist), ncolumns = 1,
"~/leukemia_DNN_result.txt")
# write(t(leukemia_dnn_test_auc_unlist), ncolumns = 1,
"~/leukemia_DNN_auc_result.txt")
# write(t(leukemia_dnn_exectime_unlist), ncolumns = 1,
"~/leukemia_DNN_time.txt")
# write(t(leukemia_dnn_train_logger_unlist), ncolumns = 100,
"~/leukemia_DNN_train_logger.txt")
# write(t(leukemia_dnn_test_logger_unlist), ncolumns = 100,
"~/leukemia_DNN_test_logger.txt")
leukemia_dnn_test_confusion_unlist <- data.frame()
for(i in 1:100){
  for(j in 1:4){
    leukemia_dnn_test_confusion_unlist[i, j] <-
unlist(leukemia_dnn_test_confusion_list[[i]]$t)[j]
  }
}
leukemia_dnn_test_confusion_unlist <- cbind(c(1:100),
leukemia_dnn_test_confusion_unlist)
colnames(leukemia_dnn_test_confusion_unlist) <- c("iter","TN", "FN", "FP", "TP")
head(leukemia_dnn_test_confusion_unlist)
tail(leukemia_dnn_test_confusion_unlist)

# write.table(leukemia_dnn_test_confusion_unlist, "~/leukemia_dnn_confusion.txt",
col.names = TRUE)


leukemia_dnn_test_acc_unlist <- read.table("~/leukemia_DNN_result.txt")
leukemia_dnn_test_acc_unlist <- cbind(leukemia_dnn_test_acc_unlist, rep("DNN(변수
전체)", 100))
names(leukemia_dnn_test_acc_unlist) <- c("Accuracy", "Model")

leukemia_dnn_test_auc_unlist <- read.table("~/leukemia_DNN_auc_result.txt")
leukemia_dnn_test_auc_unlist <- cbind(leukemia_dnn_test_auc_unlist, rep("DNN(변
수전체)", 100))
```

```
names(leukemia_dnn_test_auc_unlist) <- c("AUC", "Model")

leukemia_dnn_exectime_unlist <- read.table("~/leukemia_DNN_time.txt")
leukemia_dnn_exectime_unlist <- cbind(leukemia_dnn_exectime_unlist, rep("DNN(변수전체)", 100))
names(leukemia_dnn_exectime_unlist) <- c("Time", "Model")

# Accuracy 분포 확인.
summary(leukemia_dnn_test_acc_unlist)

win.graph()
boxplot(leukemia_dnn_test_acc_unlist[1],  main = "Leukemia data. DNN 은닉층 1, 은닉노드 20. (전체변수)",
        xlab = "Iter = 150", ylab = "Accuracy", ylim = c(0, 1))
vioplot(leukemia_dnn_test_acc_unlist[1], main = "Leukemia data. DNN 은닉층 1, 은닉노드 20. (전체변수)",
        xlab = "Iter = 150", ylab = "Accuracy", ylim = c(0, 1))
points(mean(leukemia_dnn_test_acc_unlist$Accuracy), col = "red", pch = 17) # mean 표시

# AUC 분포 확인.
summary(leukemia_dnn_test_auc_unlist)

win.graph()
boxplot(leukemia_dnn_test_auc_unlist[1],  main = "Leukemia data. DNN 은닉층 1, 은닉노드 20. (전체변수)",
        xlab = "Iter = 150", ylab = "AUC", ylim = c(0, 1))
vioplot(leukemia_dnn_test_auc_unlist[1], main = "Leukemia data. DNN 은닉층 1, 은닉노드 20. (전체변수)",
        xlab = "Iter = 150", ylab = "AUC", ylim = c(0, 1))
points(mean(leukemia_dnn_test_auc_unlist$AUC), col = "red", pch = 17) # mean 표시

# Time 분포 확인.
summary(leukemia_dnn_exectime_unlist)

win.graph()
boxplot(leukemia_dnn_exectime_unlist[1],  main = "Leukemia data. DNN 은닉층 1, 은닉노드 20. (전체변수) 학습시간",
        xlab = "Iter = 150", ylab = "Time(단위 : 초)", ylim = c(0, 4))
vioplot(leukemia_dnn_exectime_unlist[1], main = "Leukemia data. DNN 은닉층 1, 은닉노드 20. (전체변수) 학습시간",
        xlab = "Iter = 150", ylab = "Time(단위 : 초)", ylim = c(0, 4))
points(mean(leukemia_dnn_exectime_unlist$Time), col = "red", pch = 17) # mean 표시

### 2. SIS 변수선택 (leukemia data) ###
### 2. SIS 변수선택 (leukemia data) ###
### 2. SIS 변수선택 (leukemia data) ###

dim(leukemia_data)
# 72x7130

leukemia_data_x <- leukemia_data[ ,-7130]
leukemia_data_x <- data.matrix(leukemia_data_x)
leukemia_data_x <- standardize(leukemia_data_x)

leukemia_data_y <- leukemia_data[ ,7130]

## SCAD ##
## SCAD ##
```

```
## SCAD ##
leukemia_SIS_model_SCAD <- SIS(leukemia_data_x, leukemia_data_y, family =
'binomial', tune = 'bic', penalty = "SCAD",
              perm = TRUE, q = 0.9, greedy = TRUE, seed = 31)

leukemia_SIS_model_SCAD
leukemia_SIS_model_SCAD$ix
leukemia_SIS_model_SCAD$fit$beta


## MCP ##
## MCP ##
## MCP ##
leukemia_SIS_model_MCP <- SIS(leukemia_data_x, leukemia_data_y, family =
'binomial', tune = 'bic', penalty = "MCP",
                              perm = TRUE, q = 0.9, greedy = TRUE, seed =
31)

leukemia_SIS_model_MCP
leukemia_SIS_model_MCP$ix
leukemia_SIS_model_MCP$fit$beta

## LASSO ##
## LASSO ##
## LASSO ##
leukemia_SIS_model_LASSO <- SIS(leukemia_data_x, leukemia_data_y, family =
'binomial', tune = 'bic', penalty = "lasso",
                            perm = TRUE, q = 0.9, greedy = TRUE, seed =
31)

leukemia_SIS_model_LASSO
leukemia_SIS_model_LASSO$ix
leukemia_SIS_model_LASSO$fit$beta

## 각 패널티별 변수선택 확인
leukemia_SIS_model_SCAD$ix  # 1144 2597 4196 4847
leukemia_SIS_model_MCP$ix  # 1144 4847
leukemia_SIS_model_LASSO$ix  # 1144 2684 4847 6855

### 2.1 SIS SCAD 변수선택 (leukemia data) 후 DNN ###
### 2.1 SIS SCAD 변수선택 (leukemia data) 후 DNN ###
### 2.1 SIS SCAD 변수선택 (leukemia data) 후 DNN ###

# 신경망 학습용 data set
# Train : Test = 8 : 2 나누기. (100번)
# seed 를 바꿔가며 100개의 서로 다른 train, test data set 생성.
# list() 객체에 저장.

leukemia_SCAD_train_x_list_num <- list()
leukemia_SCAD_train_y_list_num <- list()

leukemia_SCAD_test_x_list_num <- list()
leukemia_SCAD_test_y_list_num <- list()

for(i in 1:100){
  set.seed(i)
  train_index <- sample(1:nrow(leukemia_data), 0.8 * nrow(leukemia_data),
replace = FALSE)
  leukemia_train_x_num <- leukemia_data[train_index, c(1144, 2597, 4196, 4847)]
  leukemia_train_y_num <- leukemia_data[train_index, 7130]
```

```
  leukemia_test_x_num <- leukemia_data[-train_index, c(1144, 2597, 4196,
4847)]
  leukemia_test_y_num <- leukemia_data[-train_index, 7130]

  leukemia_SCAD_train_x_list_num[[i]] <- leukemia_train_x_num
  leukemia_SCAD_train_y_list_num[[i]] <- leukemia_train_y_num

  leukemia_SCAD_test_x_list_num[[i]] <- leukemia_test_x_num
  leukemia_SCAD_test_y_list_num[[i]] <- leukemia_test_y_num

}


############
## 신경망 학습 ##
############
# https://mxnet.apache.org/api/r  참고.
library(mxnet)

leukemia_SCAD_model_list <- list()
leukemia_SCAD_dnn_test_predict_list <- list()
leukemia_SCAD_dnn_test_predict_label_list <- list()
leukemia_SCAD_dnn_test_confusion_list <- list()
leukemia_SCAD_dnn_test_acc_list <- list() # y_hat 과 test data set의 y 로
Accuracy 저장 list 할당.
leukemia_SCAD_dnn_exectime_list <- list()
leukemia_SCAD_dnn_train_logger_list <- list()
leukemia_SCAD_dnn_test_logger_list <- list()
leukemia_SCAD_dnn_test_auc_list <- list()

for(i in 1:100){ # 1:100
  leukemia_SCAD_train_x_scale <- scale(leukemia_SCAD_train_x_list_num[[i]])
  leukemia_SCAD_test_x_scale <- scale(leukemia_SCAD_test_x_list_num[[i]])

  leukemia_SCAD_train_x_datamatrix <- data.matrix(leukemia_SCAD_train_x_scale)
  leukemia_SCAD_test_x_datamatrix <- data.matrix(leukemia_SCAD_test_x_scale)

  leukemia_SCAD_train_y <- leukemia_SCAD_train_y_list_num[[i]]
  leukemia_SCAD_test_y <- leukemia_SCAD_test_y_list_num[[i]]

  mx.set.seed(2020)  # 가중치 초기값 고정.

  input <- mx.symbol.Variable("data")
  fc1 <- mx.symbol.FullyConnected(data = input, num.hidden = 20)  # Hyper
parameter : 은닉노드 수
  act1 <- mx.symbol.Activation(data = fc1, act_type = "sigmoid")  # Hyper
parameter : 활성화 함수 종류
  fc2 <- mx.symbol.FullyConnected(data = act1, num.hidden = 2)
  lro <- mx.symbol.SoftmaxOutput(data = fc2)  # Hyper parameter : 출력노드 함
수 종류

  tic()

  logger <- mx.metric.logger$new()
  model <- mx.model.FeedForward.create(symbol = lro, X =
leukemia_SCAD_train_x_datamatrix, y = leukemia_SCAD_train_y,
                                       eval.data = list(data =
leukemia_SCAD_test_x_datamatrix, label = leukemia_SCAD_test_y),
                                       ctx = mx.gpu(), num.round = 300,
optimizer = 'sgd', # Hyper parameter : iter 수, 최적화 함수 종류
                                       array.batch.size = 5, learning.rate = 0.001,
```

```
momentum = 0.9, eval.metric = mx.metric.accuracy,
                                      verbose = T, epoch.end.callback =
mx.callback.log.train.metric(1, logger))  # Hyper parameter : 배치 사이즈, 학습률,
모멘텀 값.

  exectime <- toc()
  exectime <- round(exectime$toc - exectime$tic, 5)

  leukemia_SCAD_model_list[[i]] <- model
  leukemia_SCAD_dnn_test_predict_list[[i]] <-
predict(leukemia_SCAD_model_list[[i]], leukemia_SCAD_test_x_datamatrix)
  leukemia_SCAD_dnn_test_predict_label_list[[i]] <-
max.col(t(leukemia_SCAD_dnn_test_predict_list[[i]])) - 1
  leukemia_SCAD_dnn_test_confusion_list[[i]] <- CrossTable(x =
leukemia_SCAD_test_y_list_num[[i]], y =
leukemia_SCAD_dnn_test_predict_label_list[[i]])
  leukemia_SCAD_dnn_test_acc_list[[i]] <-
(leukemia_SCAD_dnn_test_confusion_list[[i]]$t[1] +
leukemia_SCAD_dnn_test_confusion_list[[i]]$t[4])  /
sum(leukemia_SCAD_dnn_test_confusion_list[[i]]$t)
  auc <- performance(prediction( predict(leukemia_SCAD_model_list[[i]],
leukemia_SCAD_test_x_datamatrix)[2, ], leukemia_SCAD_test_y_list_num[[i]],
label.ordering = c(0, 1)), measure = "auc")
  leukemia_SCAD_dnn_test_auc_list[[i]] <- auc@y.values[[1]]

  leukemia_SCAD_dnn_exectime_list[[i]] <- exectime
  leukemia_SCAD_dnn_train_logger_list[[i]] <- logger$train
  leukemia_SCAD_dnn_test_logger_list[[i]] <- logger$eval
}

leukemia_SCAD_dnn_test_acc_unlist <- unlist(leukemia_SCAD_dnn_test_acc_list)
leukemia_SCAD_dnn_test_auc_unlist <- unlist(leukemia_SCAD_dnn_test_auc_list)
leukemia_SCAD_dnn_exectime_unlist <- unlist(leukemia_SCAD_dnn_exectime_list)

leukemia_SCAD_dnn_train_logger_unlist <- data.frame(
matrix(unlist(leukemia_SCAD_dnn_train_logger_list), ncol = 100))
leukemia_SCAD_dnn_test_logger_unlist <- data.frame(
matrix(unlist(leukemia_SCAD_dnn_test_logger_list), ncol = 100))

# 결과저장
# write(t(leukemia_SCAD_dnn_test_acc_unlist), ncolumns = 1,
"~/leukemia_SCAD_DNN_result.txt")
# write(t(leukemia_SCAD_dnn_test_auc_unlist), ncolumns = 1,
"~/leukemia_SCAD_DNN_auc_result.txt")
# write(t(leukemia_SCAD_dnn_exectime_unlist), ncolumns = 1,
"~/leukemia_SCAD_DNN_time.txt")
# write(t(leukemia_SCAD_dnn_train_logger_unlist), ncolumns = 100,
"~/leukemia_SCAD_DNN_train_logger.txt")
# write(t(leukemia_SCAD_dnn_test_logger_unlist), ncolumns = 100,
"~/leukemia_SCAD_DNN_test_logger.txt")
leukemia_SCAD_dnn_test_confusion_unlist <- data.frame()
for(i in 1:100){
  for(j in 1:4){
    leukemia_SCAD_dnn_test_confusion_unlist[i, j] <-
unlist(leukemia_SCAD_dnn_test_confusion_list[[i]]$t)[j]
  }
}
leukemia_SCAD_dnn_test_confusion_unlist <- cbind(c(1:100),
leukemia_SCAD_dnn_test_confusion_unlist)
colnames(leukemia_SCAD_dnn_test_confusion_unlist) <- c("iter","TN", "FN", "FP",
"TP")
```

```
head(leukemia_SCAD_dnn_test_confusion_unlist)
tail(leukemia_SCAD_dnn_test_confusion_unlist)

# write.table(leukemia_SCAD_dnn_test_confusion_unlist,
"~/leukemia_SCAD_dnn_confusion.txt", col.names = TRUE)


leukemia_SCAD_dnn_test_acc_unlist <-
read.table("~/leukemia_SCAD_DNN_result.txt")
leukemia_SCAD_dnn_test_acc_unlist <- cbind(leukemia_SCAD_dnn_test_acc_unlist,
rep("DNN(SCAD 변수선택)", 100))
names(leukemia_SCAD_dnn_test_acc_unlist) <- c("Accuracy", "Model")

leukemia_SCAD_dnn_test_auc_unlist <-
read.table("~/leukemia_SCAD_DNN_auc_result.txt")
leukemia_SCAD_dnn_test_auc_unlist <- cbind(leukemia_SCAD_dnn_test_auc_unlist,
rep("DNN(SCAD 변수선택)", 100))
names(leukemia_SCAD_dnn_test_auc_unlist) <- c("AUC", "Model")

leukemia_SCAD_dnn_exectime_unlist <-
read.table("~/leukemia_SCAD_DNN_time.txt")
leukemia_SCAD_dnn_exectime_unlist <- cbind(leukemia_SCAD_dnn_exectime_unlist,
rep("DNN(SCAD 변수선택)", 100))
names(leukemia_SCAD_dnn_exectime_unlist) <- c("Time", "Model")

# Accuracy 분포 확인.
summary(leukemia_SCAD_dnn_test_acc_unlist)

win.graph()
boxplot(leukemia_SCAD_dnn_test_acc_unlist[1],  main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (SCAD 변수선택)",
        xlab = "Iter = 300", ylab = "Accuracy", ylim = c(0, 1))
vioplot(leukemia_SCAD_dnn_test_acc_unlist[1], main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (SCAD 변수선택)",
        xlab = "Iter = 300", ylab = "Accuracy", ylim = c(0, 1))
points(mean(leukemia_SCAD_dnn_test_acc_unlist$Accuracy), col = "red", pch =
17) # mean 표시

# AUC 분포 확인.
summary(leukemia_SCAD_dnn_test_auc_unlist)

win.graph()
boxplot(leukemia_SCAD_dnn_test_auc_unlist[1],  main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (SCAD 변수선택)",
        xlab = "Iter = 300", ylab = "AUC", ylim = c(0, 1))
vioplot(leukemia_SCAD_dnn_test_auc_unlist[1], main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (SCAD 변수선택)",
        xlab = "Iter = 300", ylab = "AUC", ylim = c(0, 1))
points(mean(leukemia_SCAD_dnn_test_auc_unlist$AUC), col = "red", pch = 17) #
mean 표시


# Time 분포 확인.
summary(leukemia_SCAD_dnn_exectime_unlist)

win.graph()
boxplot(leukemia_SCAD_dnn_exectime_unlist[1],  main = "Leukemia data. DNN 은
닉층 1, 은닉노드 20. (SCAD 변수선택) 학습시간",
        xlab = "Iter = 300", ylab = "Time(단위 : 초)", ylim = c(0, 9))
vioplot(leukemia_SCAD_dnn_exectime_unlist[1], main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (SCAD 변수선택) 학습시간",
```

```
        xlab = "Iter = 300", ylab = "Time(단위 : 초)", ylim = c(0, 9))
points(mean(leukemia_SCAD_dnn_exectime_unlist$Time), col = "red", pch = 17) #
mean 표시

### 2.2 SIS MCP 변수선택 (leukemia data) 후 DNN ###
### 2.2 SIS MCP 변수선택 (leukemia data) 후 DNN ###
### 2.2 SIS MCP 변수선택 (leukemia data) 후 DNN ###

# 신경망 학습용 data set
# Train : Test = 8 : 2 나누기. (100번)
# seed 를 바꿔가며 100개의 서로 다른 train, test data set 생성.
# list() 객체에 저장.

leukemia_MCP_train_x_list_num <- list()
leukemia_MCP_train_y_list_num <- list()

leukemia_MCP_test_x_list_num <- list()
leukemia_MCP_test_y_list_num <- list()

for(i in 1:100){
  set.seed(i)
  train_index <- sample(1:nrow(leukemia_data), 0.8 * nrow(leukemia_data),
replace = FALSE)
  leukemia_train_x_num <- leukemia_data[train_index, c(1144, 4847)]
  leukemia_train_y_num <- leukemia_data[train_index, 7130]

  leukemia_test_x_num <- leukemia_data[-train_index, c(1144, 4847)]
  leukemia_test_y_num <- leukemia_data[-train_index, 7130]

  leukemia_MCP_train_x_list_num[[i]] <- leukemia_train_x_num
  leukemia_MCP_train_y_list_num[[i]] <- leukemia_train_y_num

  leukemia_MCP_test_x_list_num[[i]] <- leukemia_test_x_num
  leukemia_MCP_test_y_list_num[[i]] <- leukemia_test_y_num

}


############
## 신경망 학습 ##
############
# https://mxnet.apache.org/api/r  참고.
library(mxnet)

leukemia_MCP_model_list <- list()
leukemia_MCP_dnn_test_predict_list <- list()
leukemia_MCP_dnn_test_predict_label_list <- list()
leukemia_MCP_dnn_test_confusion_list <- list()
leukemia_MCP_dnn_test_acc_list <- list() # y_hat 과 test data set의 y 로
Accuracy 저장 list 할당.
leukemia_MCP_dnn_exectime_list <- list()
leukemia_MCP_dnn_train_logger_list <- list()
leukemia_MCP_dnn_test_logger_list <- list()
leukemia_MCP_dnn_test_auc_list <- list()

for(i in 1:100){ # 1:100
  leukemia_MCP_train_x_scale <- scale(leukemia_MCP_train_x_list_num[[i]])
  leukemia_MCP_test_x_scale <- scale(leukemia_MCP_test_x_list_num[[i]])

  leukemia_MCP_train_x_datamatrix <- data.matrix(leukemia_MCP_train_x_scale)
  leukemia_MCP_test_x_datamatrix <- data.matrix(leukemia_MCP_test_x_scale)
```

```
  leukemia_MCP_train_y <- leukemia_MCP_train_y_list_num[[i]]
  leukemia_MCP_test_y <- leukemia_MCP_test_y_list_num[[i]]

  mx.set.seed(2020)  # 가중치 초기값 고정.

  input <- mx.symbol.Variable("data")
  fc1 <- mx.symbol.FullyConnected(data = input, num.hidden = 20)  # Hyper
parameter : 은닉노드 수
  act1 <- mx.symbol.Activation(data = fc1, act_type = "sigmoid")  # Hyper
parameter : 활성화 함수 종류
  fc2 <- mx.symbol.FullyConnected(data = act1, num.hidden = 2)
  lro <- mx.symbol.SoftmaxOutput(data = fc2)  # Hyper parameter : 출력노드 함
수 종류

  tic()

  logger <- mx.metric.logger$new()
  model <- mx.model.FeedForward.create(symbol = lro, X =
leukemia_MCP_train_x_datamatrix, y = leukemia_MCP_train_y,
                                       eval.data = list(data =
leukemia_MCP_test_x_datamatrix, label = leukemia_MCP_test_y),
                                       ctx = mx.gpu(), num.round = 400,
optimizer = 'sgd', # Hyper parameter : iter 수, 최적화 함수 종류
                                       array.batch.size = 5, learning.rate = 0.001,
momentum = 0.9, eval.metric = mx.metric.accuracy,
                                       verbose = T, epoch.end.callback =
mx.callback.log.train.metric(1, logger))  # Hyper parameter : 배치 사이즈, 학습률,
모멘텀 값.

  exectime <- toc()
  exectime <- round(exectime$toc - exectime$tic, 5)

  leukemia_MCP_model_list[[i]] <- model
  leukemia_MCP_dnn_test_predict_list[[i]] <-
predict(leukemia_MCP_model_list[[i]], leukemia_MCP_test_x_datamatrix)
  leukemia_MCP_dnn_test_predict_label_list[[i]] <-
max.col(t(leukemia_MCP_dnn_test_predict_list[[i]])) - 1
  leukemia_MCP_dnn_test_confusion_list[[i]] <- CrossTable(x =
leukemia_MCP_test_y_list_num[[i]], y =
leukemia_MCP_dnn_test_predict_label_list[[i]])
  leukemia_MCP_dnn_test_acc_list[[i]] <-
(leukemia_MCP_dnn_test_confusion_list[[i]]$t[1] +
leukemia_MCP_dnn_test_confusion_list[[i]]$t[4])  /
sum(leukemia_MCP_dnn_test_confusion_list[[i]]$t)
  auc <- performance(prediction( predict(leukemia_MCP_model_list[[i]],
leukemia_MCP_test_x_datamatrix)[2, ], leukemia_MCP_test_y_list_num[[i]],
label.ordering = c(0, 1)), measure = "auc")
  leukemia_MCP_dnn_test_auc_list[[i]] <- auc@y.values[[1]]

  leukemia_MCP_dnn_exectime_list[[i]] <- exectime
  leukemia_MCP_dnn_train_logger_list[[i]] <- logger$train
  leukemia_MCP_dnn_test_logger_list[[i]] <- logger$eval
}

leukemia_MCP_dnn_test_acc_unlist <- unlist(leukemia_MCP_dnn_test_acc_list)
leukemia_MCP_dnn_test_auc_unlist <- unlist(leukemia_MCP_dnn_test_auc_list)
leukemia_MCP_dnn_exectime_unlist <- unlist(leukemia_MCP_dnn_exectime_list)

leukemia_MCP_dnn_train_logger_unlist <- data.frame(
matrix(unlist(leukemia_MCP_dnn_train_logger_list), ncol = 100))
```

```
leukemia_MCP_dnn_test_logger_unlist <- data.frame(
matrix(unlist(leukemia_MCP_dnn_test_logger_list), ncol = 100))

# 결과저장
# write(t(leukemia_MCP_dnn_test_acc_unlist), ncolumns = 1,
"~/leukemia_MCP_DNN_result.txt")
# write(t(leukemia_MCP_dnn_test_auc_unlist), ncolumns = 1,
"~/leukemia_MCP_DNN_auc_result.txt")
# write(t(leukemia_MCP_dnn_exectime_unlist), ncolumns = 1,
"~/leukemia_MCP_DNN_time.txt")
# write(t(leukemia_MCP_dnn_train_logger_unlist), ncolumns = 100,
"~/leukemia_MCP_DNN_train_logger.txt")
# write(t(leukemia_MCP_dnn_test_logger_unlist), ncolumns = 100,
"~/leukemia_MCP_DNN_test_logger.txt")
leukemia_MCP_dnn_test_confusion_unlist <- data.frame()
for(i in 1:100){
  for(j in 1:4){
    leukemia_MCP_dnn_test_confusion_unlist[i, j] <-
unlist(leukemia_MCP_dnn_test_confusion_list[[i]]$t)[j]
  }
}
leukemia_MCP_dnn_test_confusion_unlist <- cbind(c(1:100),
leukemia_MCP_dnn_test_confusion_unlist)
colnames(leukemia_MCP_dnn_test_confusion_unlist) <- c("iter","TN", "FN", "FP",
"TP")
head(leukemia_MCP_dnn_test_confusion_unlist)
tail(leukemia_MCP_dnn_test_confusion_unlist)

# write.table(leukemia_MCP_dnn_test_confusion_unlist,
"~/leukemia_MCP_dnn_confusion.txt", col.names = TRUE)


leukemia_MCP_dnn_test_acc_unlist <- read.table("~/leukemia_MCP_DNN_result.txt")
leukemia_MCP_dnn_test_acc_unlist <- cbind(leukemia_MCP_dnn_test_acc_unlist,
rep("DNN(MCP 변수선택)", 100))
names(leukemia_MCP_dnn_test_acc_unlist) <- c("Accuracy", "Model")

leukemia_MCP_dnn_test_auc_unlist <-
read.table("~/leukemia_MCP_DNN_auc_result.txt")
leukemia_MCP_dnn_test_auc_unlist <- cbind(leukemia_MCP_dnn_test_auc_unlist,
rep("DNN(MCP 변수선택)", 100))
names(leukemia_MCP_dnn_test_auc_unlist) <- c("AUC", "Model")

leukemia_MCP_dnn_exectime_unlist <- read.table("~/leukemia_MCP_DNN_time.txt")
leukemia_MCP_dnn_exectime_unlist <- cbind(leukemia_MCP_dnn_exectime_unlist,
rep("DNN(MCP 변수선택)", 100))
names(leukemia_MCP_dnn_exectime_unlist) <- c("Time", "Model")

# Accuracy 분포 확인.
summary(leukemia_MCP_dnn_test_acc_unlist)

win.graph()
boxplot(leukemia_MCP_dnn_test_acc_unlist[1],  main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (MCP 변수선택)",
        xlab = "Iter = 400", ylab = "Accuracy", ylim = c(0, 1))
vioplot(leukemia_MCP_dnn_test_acc_unlist[1], main = "Leukemia data. DNN 은닉층
1, 은닉노드 20. (MCP 변수선택)",
        xlab = "Iter = 400", ylab = "Accuracy", ylim = c(0, 1))
points(mean(leukemia_MCP_dnn_test_acc_unlist$Accuracy), col = "red", pch = 17)
# mean 표시
```

```
# AUC 분포 확인.
summary(leukemia_MCP_dnn_test_auc_unlist)

win.graph()
boxplot(leukemia_MCP_dnn_test_auc_unlist[1],  main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (MCP 변수선택)",
        xlab = "Iter = 400", ylab = "AUC", ylim = c(0, 1))
vioplot(leukemia_MCP_dnn_test_auc_unlist[1], main = "Leukemia data. DNN 은닉층
1, 은닉노드 20. (MCP 변수선택)",
        xlab = "Iter = 400", ylab = "AUC", ylim = c(0, 1))
points(mean(leukemia_MCP_dnn_test_auc_unlist$AUC), col = "red", pch = 17) #
mean 표시

# Time 분포 확인.
summary(leukemia_MCP_dnn_exectime_unlist)

win.graph()
boxplot(leukemia_MCP_dnn_exectime_unlist[1],  main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (MCP 변수선택) 학습시간",
        xlab = "Iter = 400", ylab = "Time(단위 : 초)", ylim = c(0, 11))
vioplot(leukemia_MCP_dnn_exectime_unlist[1], main = "Leukemia data. DNN 은닉층
1, 은닉노드 20. (MCP 변수선택) 학습시간",
        xlab = "Iter = 400", ylab = "Time(단위 : 초)", ylim = c(0, 11))
points(mean(leukemia_MCP_dnn_exectime_unlist$Time), col = "red", pch = 17) #
mean 표시

### 2.3 SIS LASSO 변수선택 (leukemia data) 후 DNN ###
### 2.3 SIS LASSO 변수선택 (leukemia data) 후 DNN ###
### 2.3 SIS LASSO 변수선택 (leukemia data) 후 DNN ###

# 신경망 학습용 data set
# Train : Test = 8 : 2 나누기. (100번)
# seed 를 바꿔가며 100개의 서로 다른 train, test data set 생성.
# list() 객체에 저장.

leukemia_LASSO_train_x_list_num <- list()
leukemia_LASSO_train_y_list_num <- list()

leukemia_LASSO_test_x_list_num <- list()
leukemia_LASSO_test_y_list_num <- list()

for(i in 1:100){
  set.seed(i)
  train_index <- sample(1:nrow(leukemia_data), 0.8 * nrow(leukemia_data),
replace = FALSE)
  leukemia_train_x_num <- leukemia_data[train_index, c(1144, 2684, 4847, 6855)]
  leukemia_train_y_num <- leukemia_data[train_index, 7130]

  leukemia_test_x_num <- leukemia_data[-train_index, c(1144, 2684, 4847,
6855)]
  leukemia_test_y_num <- leukemia_data[-train_index, 7130]

  leukemia_LASSO_train_x_list_num[[i]] <- leukemia_train_x_num
  leukemia_LASSO_train_y_list_num[[i]] <- leukemia_train_y_num

  leukemia_LASSO_test_x_list_num[[i]] <- leukemia_test_x_num
  leukemia_LASSO_test_y_list_num[[i]] <- leukemia_test_y_num

}
```

```
############
## 신경망 학습 ##
############
# https://mxnet.apache.org/api/r  참고.
library(mxnet)

leukemia_LASSO_model_list <- list()
leukemia_LASSO_dnn_test_predict_list <- list()
leukemia_LASSO_dnn_test_predict_label_list <- list()
leukemia_LASSO_dnn_test_confusion_list <- list()
leukemia_LASSO_dnn_test_acc_list <- list() # y_hat 과 test data set의 y 로
Accuracy 저장 list 할당.
leukemia_LASSO_dnn_exectime_list <- list()
leukemia_LASSO_dnn_train_logger_list <- list()
leukemia_LASSO_dnn_test_logger_list <- list()
leukemia_LASSO_dnn_test_auc_list <- list()

for(i in 1:100){ # 1:100
  leukemia_LASSO_train_x_scale <- scale(leukemia_LASSO_train_x_list_num[[i]])
  leukemia_LASSO_test_x_scale <- scale(leukemia_LASSO_test_x_list_num[[i]])

  leukemia_LASSO_train_x_datamatrix <-
data.matrix(leukemia_LASSO_train_x_scale)
  leukemia_LASSO_test_x_datamatrix <-
data.matrix(leukemia_LASSO_test_x_scale)

  leukemia_LASSO_train_y <- leukemia_LASSO_train_y_list_num[[i]]
  leukemia_LASSO_test_y <- leukemia_LASSO_test_y_list_num[[i]]

  mx.set.seed(2020)  # 가중치 초기값 고정.

  input <- mx.symbol.Variable("data")
  fc1 <- mx.symbol.FullyConnected(data = input, num.hidden = 20)  # Hyper
parameter : 은닉노드 수
  act1 <- mx.symbol.Activation(data = fc1, act_type = "sigmoid")  # Hyper
parameter : 활성화 함수 종류
  fc2 <- mx.symbol.FullyConnected(data = act1, num.hidden = 2)
  lro <- mx.symbol.SoftmaxOutput(data = fc2)  # Hyper parameter : 출력노드 함
수 종류

  tic()

  logger <- mx.metric.logger$new()
  model <- mx.model.FeedForward.create(symbol = lro, X =
leukemia_LASSO_train_x_datamatrix, y = leukemia_LASSO_train_y,
                                       eval.data = list(data =
leukemia_LASSO_test_x_datamatrix, label = leukemia_LASSO_test_y),
                                       ctx = mx.gpu(), num.round = 400,
optimizer = 'sgd', # Hyper parameter : iter 수, 최적화 함수 종류
                                       array.batch.size = 5, learning.rate = 0.001,
momentum = 0.9, eval.metric = mx.metric.accuracy,
                                       verbose = T, epoch.end.callback =
mx.callback.log.train.metric(1, logger))  # Hyper parameter : 배치 사이즈, 학습률,
모멘텀 값.

  exectime <- toc()
  exectime <- round(exectime$toc - exectime$tic, 5)

  leukemia_LASSO_model_list[[i]] <- model
  leukemia_LASSO_dnn_test_predict_list[[i]] <-
predict(leukemia_LASSO_model_list[[i]], leukemia_LASSO_test_x_datamatrix)
```

```
  leukemia_LASSO_dnn_test_predict_label_list[[i]] <-
max.col(t(leukemia_LASSO_dnn_test_predict_list[[i]])) - 1
  leukemia_LASSO_dnn_test_confusion_list[[i]] <- CrossTable(x =
leukemia_LASSO_test_y_list_num[[i]], y =
leukemia_LASSO_dnn_test_predict_label_list[[i]])
  leukemia_LASSO_dnn_test_acc_list[[i]] <-
(leukemia_LASSO_dnn_test_confusion_list[[i]]$t[1] +
leukemia_LASSO_dnn_test_confusion_list[[i]]$t[4])  /
sum(leukemia_LASSO_dnn_test_confusion_list[[i]]$t)
  auc <- performance(prediction( predict(leukemia_LASSO_model_list[[i]],
leukemia_LASSO_test_x_datamatrix)[2, ], leukemia_LASSO_test_y_list_num[[i]],
label.ordering = c(0, 1)), measure = "auc")
  leukemia_LASSO_dnn_test_auc_list[[i]] <- auc@y.values[[1]]

  leukemia_LASSO_dnn_exectime_list[[i]] <- exectime
  leukemia_LASSO_dnn_train_logger_list[[i]] <- logger$train
  leukemia_LASSO_dnn_test_logger_list[[i]] <- logger$eval
}

leukemia_LASSO_dnn_test_acc_unlist <- unlist(leukemia_LASSO_dnn_test_acc_list)
leukemia_LASSO_dnn_test_auc_unlist <- unlist(leukemia_LASSO_dnn_test_auc_list)
leukemia_LASSO_dnn_exectime_unlist <-
unlist(leukemia_LASSO_dnn_exectime_list)

leukemia_LASSO_dnn_train_logger_unlist <- data.frame(
matrix(unlist(leukemia_LASSO_dnn_train_logger_list), ncol = 100))
leukemia_LASSO_dnn_test_logger_unlist <- data.frame(
matrix(unlist(leukemia_LASSO_dnn_test_logger_list), ncol = 100))

# 결과저장
# write(t(leukemia_LASSO_dnn_test_acc_unlist), ncolumns = 1,
"~/leukemia_LASSO_DNN_result.txt")
# write(t(leukemia_LASSO_dnn_test_auc_unlist), ncolumns = 1,
"~/leukemia_LASSO_DNN_auc_result.txt")
# write(t(leukemia_LASSO_dnn_exectime_unlist), ncolumns = 1,
"~/leukemia_LASSO_DNN_time.txt")
# write(t(leukemia_LASSO_dnn_train_logger_unlist), ncolumns = 100,
"~/leukemia_LASSO_DNN_train_logger.txt")
# write(t(leukemia_LASSO_dnn_test_logger_unlist), ncolumns = 100,
"~/leukemia_LASSO_DNN_test_logger.txt")
leukemia_LASSO_dnn_test_confusion_unlist <- data.frame()
for(i in 1:100){
  for(j in 1:4){
    leukemia_LASSO_dnn_test_confusion_unlist[i, j] <-
unlist(leukemia_LASSO_dnn_test_confusion_list[[i]]$t)[j]
  }
}
leukemia_LASSO_dnn_test_confusion_unlist <- cbind(c(1:100),
leukemia_LASSO_dnn_test_confusion_unlist)
colnames(leukemia_LASSO_dnn_test_confusion_unlist) <- c("iter","TN", "FN", "FP",
"TP")
head(leukemia_LASSO_dnn_test_confusion_unlist)
tail(leukemia_LASSO_dnn_test_confusion_unlist)

# write.table(leukemia_LASSO_dnn_test_confusion_unlist,
"~/leukemia_LASSO_dnn_confusion.txt", col.names = TRUE)

leukemia_LASSO_dnn_test_acc_unlist <-
read.table("~/leukemia_LASSO_DNN_result.txt")
leukemia_LASSO_dnn_test_acc_unlist <-
cbind(leukemia_LASSO_dnn_test_acc_unlist, rep("DNN(LASSO 변수선택)", 100))
```

```
names(leukemia_LASSO_dnn_test_acc_unlist) <- c("Accuracy", "Model")

leukemia_LASSO_dnn_test_auc_unlist <-
read.table("~/leukemia_LASSO_DNN_auc_result.txt")
leukemia_LASSO_dnn_test_auc_unlist <-
cbind(leukemia_LASSO_dnn_test_auc_unlist, rep("DNN(LASSO 변수선택)", 100))
names(leukemia_LASSO_dnn_test_auc_unlist) <- c("AUC", "Model")

leukemia_LASSO_dnn_exectime_unlist <-
read.table("~/leukemia_LASSO_DNN_time.txt")
leukemia_LASSO_dnn_exectime_unlist <-
cbind(leukemia_LASSO_dnn_exectime_unlist, rep("DNN(LASSO 변수선택)", 100))
names(leukemia_LASSO_dnn_exectime_unlist) <- c("Time", "Model")

# Accuracy 분포 확인.
summary(leukemia_LASSO_dnn_test_acc_unlist)

win.graph()
boxplot(leukemia_LASSO_dnn_test_acc_unlist[1],  main = "Leukemia data. DNN 은
닉층 1, 은닉노드 20. (LASSO 변수선택)",
        xlab = "Iter = 400", ylab = "Accuracy", ylim = c(0, 1))
vioplot(leukemia_LASSO_dnn_test_acc_unlist[1], main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (LASSO 변수선택)",
        xlab = "Iter = 400", ylab = "Accuracy", ylim = c(0, 1))
points(mean(leukemia_LASSO_dnn_test_acc_unlist$Accuracy), col = "red", pch =
17) # mean 표시

# AUC 분포 확인.
summary(leukemia_LASSO_dnn_test_auc_unlist)

win.graph()
boxplot(leukemia_LASSO_dnn_test_auc_unlist[1],  main = "Leukemia data. DNN 은
닉층 1, 은닉노드 20. (LASSO 변수선택)",
        xlab = "Iter = 400", ylab = "AUC", ylim = c(0, 1))
vioplot(leukemia_LASSO_dnn_test_auc_unlist[1], main = "Leukemia data. DNN 은닉
층 1, 은닉노드 20. (LASSO 변수선택)",
        xlab = "Iter = 400", ylab = "AUC", ylim = c(0, 1))
points(mean(leukemia_LASSO_dnn_test_auc_unlist$AUC), col = "red", pch = 17) #
mean 표시

# Time 분포 확인.
summary(leukemia_LASSO_dnn_exectime_unlist)

win.graph()
boxplot(leukemia_LASSO_dnn_exectime_unlist[1],  main = "Leukemia data. DNN 은
닉층 1, 은닉노드 20. (LASSO 변수선택) 학습시간",
        xlab = "Iter = 400", ylab = "Time(단위 : 초)", ylim = c(0, 10))
vioplot(leukemia_LASSO_dnn_exectime_unlist[1], main = "Leukemia data. DNN 은
닉층 1, 은닉노드 20. (LASSO 변수선택) 학습시간",
        xlab = "Iter = 400", ylab = "Time(단위 : 초)", ylim = c(0, 10))
points(mean(leukemia_LASSO_dnn_exectime_unlist$Time), col = "red", pch = 17)
# mean 표시
```

PUKYONG NATIONAL UNIVERSITY
부경대학교