

# 다층 퍼셉트론을 이용한 MNIST, Cifar10 데이터 셋 학습

이 한 결

서울대학교 인문대학 언어학과

[miroblog@snu.ac.kr](mailto:miroblog@snu.ac.kr)

## 요 약

본고는 다양한 하이퍼 파라미터로 다층 퍼셉트론을 만들고, 이를 활용해 MNIST와 Cifar10 데이터 셋에 대해 학습한 후, 그 성능을 비교해보았다. 실험에서는 목표(loss)함수의 종류, 정규화(regularization) 방법의 종류, 히든 레이어와 유닛의 수 등을 바꾸어 가며 성능을 측정하였다. 성능은 히든 레이어의 수보다는 히든 유닛의 수에 더 영향을 많이 받았고, Regularization의 경우 충분한 뉴런이 확보되었을 때 적용하는 것이 성능 향상에 도움이 되었다. 목표 함수로는 cross entropy가 대부분의 경우에 있어 다른 목표 함수보다 더 좋은 성능을 내었다.

## 1. 실험에 사용한 모델과 알고리즘

본 연구에서는 다층 퍼셉트론을 활용해, MNIST와 Cifar10 데이터를 학습하였다. 다층 퍼셉트론은 입력층 출력층 사이에 하나 이상의 중간층(은닉층)이 존재하는 신경망이다. 본 실험에서는 신경망의 구조와 이를 학습하는 방식에 변화를 주어 성능 차이를 알아보고자 했다.

첫번째 실험에서 사용한 하이퍼 파라미터는 아래와 같다.

Data Set	Mnist
Loss function	Cross Entropy
N_hidden layer	1, 2, 3
N_hidden units	10, 50, 100, 200, 400, 800
Activation func	Sigmoid, ReLu
Early stopping	Yes
Regularization	Drop out(0.1, 0.3, 0.5, 0.7, 0.9) Batch Normalization
Optimizer	Adam Optimizer
Output layer(for classification)	Softmax

[Experiment 1]

두번째 실험에서는 목표함수에 변화를 주었다.

Data Set	Mnist, Cifar10
Loss function	Mean Squared Error, L2 Normalization, Cross Entropy
Activation func	Sigmoid, ReLu
Early stopping	Yes
Regularization	No
Optimizer	Adam Optimizer
N_hidden layer	1
N_hidden units	100
Output layer(for classification)	Softmax

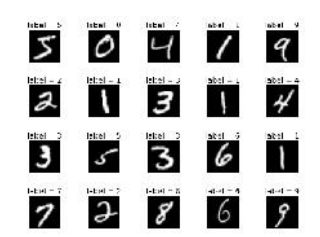

[Experiment 2]

세번째 실험에서는 앞서 진행한 두 실험에서 성능 향상에 도움이 되었던 파라미터를 활용해, Cifar10 데이터 셋을 학습하여 보았다.

Data Set	Cifar10
Loss function	Mean Squared Error, L2 Normalization, <b>Cross Entropy</b>
Activation func	<b>ReLU</b> , Sigmoid
Early stopping	Yes
Regularization	Drop out(0.1, 0.3, <b>0.5</b> , 0.7, 0.9) Batch Normalization
Optimizer	Adam Optimizer
N_hidden layer	2, 3
N_hidden units	400, 800
Output layer(for classification)	Softmax

[Experiment 3]

실험에 사용한 데이터 셋은 아래의 MNIST와 Cifar10이다.

MNIST	Cifar10
	

[Data Set Image]

데이터셋의 구체적인 스펙과 실험에 사용한 Batch의 크기는 아래와 같다.

	MNIST	Cifar10
N class	10	10
N Training sample	55000	50000
N Test sample	5000/ 10000	10000
Batch Size	100	100

## 2. 실험에 사용한 모델과 알고리즘을 선택한 이유

### 2.1 Activation : ReLu > Sigmoid

활성함수로 ReLu를 선택한 이유는 Sigmoid에 비해 계산하기가 간단하다. Sigmoid의 경우에는 지수 계산을 요한다. 네트워크의 크기가 크고, 뉴런의 수가 많아질 때 ReLu를 사용하면 학습과 평가에 소요되는 시간을 줄일 수 있다.

또, Sigmoid의 경우 saturation이 일어나기 쉽다. Sigmoid의 경우 함수 곡선의 양끝에서는 미분계수가 0에 가까워지는데, 이 때 saturation이 일어나 학습을 방해한다. 신경망의 층이 쌓일수록 gradient 계산이 어려워진다. [1]반면, ReLu의 경우에는 입력 값이 0보다 작을 때에만 saturation이 일어나서 sigmoid 함수보다는 학습에 용이하다.

### 2.2 Number of Hidden Layers and Nodes

본 실험에서는 다층 퍼셉트론의 은닉층 수를 1~3개로 제한했다.

은닉층이 하나도 존재하지 않는 네트워크는 입력으로 주어지는 데이터들이 linearly separable한 경우에만 잘 작동한다. 그런데 숫자 및 사물 인식에 있어서 입력들은 linearly separable하지 않기 때문에 보통에 1개 이상의 은닉층을 요구한다.

[2]다층 퍼셉트론으로 해결할 수 있는 문제의 경우, 이론상으로는 하나의 은닉층만으로도 충분하다. 다시말해, 입력으로 주어지는 훈련 샘플의 수 만큼 은닉 유닛이 은닉층에 존재하면, 해당 문제를 풀 수 있다. 층을 하나 더 쌓게되면, 은닉 유닛의 수를 대폭 감소시킬 수 있다. Huang(2003)이 제안하는 수는 아래와 같다.

1 <sup>st</sup> layer	$\sqrt{(m+2)N} + 2\sqrt{N/(m+2)},$	947
2 <sup>nd</sup> layer	$m\sqrt{N/(m+2)}.$	677

[MNIST m(# classes) = 10, N(#sample) = 55000]

본 실험에서는 개별 은닉층에 존재하는 뉴런의 수는 동일하게 구성하되 10, 50, 100, 200, 400, 800로 변화를 주어 기존 연구가 제안하는 수가 최적의 성능을 내는지 알아보고자 한다. 추가적으로 은닉층의 수에 따라서도 성능 차이를 알아볼 것이다.

### 2.3 Objective(loss) function

실험에 사용한 목표 함수는 Mean Squared Error, L2 normalization, Cross Entropy이다.

Mean Squared Error의 경우 일반적으로 regression에서 쓰이는데, 실제로 classification에서도 유용한지를 확인하고자 한다.

Cross entropy의 경우, classification문제에서 널리 쓰이는 함수이다. 예측한 값과, 실제 값이 갈아질수록 엔트로피가 낮아진다.

L2 normalization은 각 층의 weight의 제곱한 값을 loss function에 더해준다. 과적합을 방지하는데 도움이 된다.

Mean Squared Error	$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$
Cross entropy	$\xi(T, Y) = \sum_{i=1}^n \xi(t_i, y_i) = - \sum_{i=1}^n \sum_{c=1}^C t_{ic} \cdot \log(y_{ic})$
L2 normalization (with xent)	$\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2} \ w\ ^2$

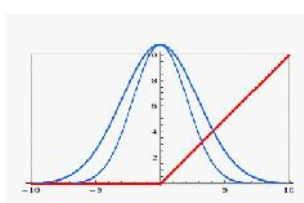
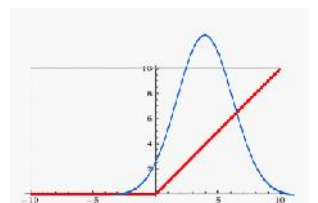
### 2.3 Regularization Technique

본 실험에서는 과적합을 방지하는 방식으로 Drop Out, Batch Normalization 방법을 사용하였다.

Drop out은 각 층에 존재하는 뉴런을 정한 확률에 따라 제거하는 방법이다. 이 방법을 사용해 학습을 하게 되면, 서로 다른 가중치로 훈련한 여러개의 네트워크를 평균하는 것과 같은 효과를 낸다.

[3]한편, 기존 연구에서는 drop out 확률을 0.5로 하는 것이 linear(feed forward) network에서는 가장 좋은 regularization 효과를 거두었다고 한다. 본 실험에서는 5가지 확률([0.1, 0.3, 0.5, 0.7, 0.9])로 성능을 비교해 보았다.

[4]Batch Normalization은 활성화함수에 들어가는 batch의 logit(weighted sum)의 분포를 정규화 해준다. Data whitening과 유사한 역할을 batch layer가 대신 해준다.

No BN	With BN
	

[Distribution of Logits]

BN을 적용할때에 주의할 점이 있는데, 기존에 bias의 역할을  $\beta$ 이 하기 때문에, weighted sum만을 batch layer로 넘겨준다. 또, ReLu의 경우에는 sigmoid와는 달리 scale factor( $\alpha$ )가 logit의 분포에 영향을 못 미쳐, 사용할 필요가 없다.

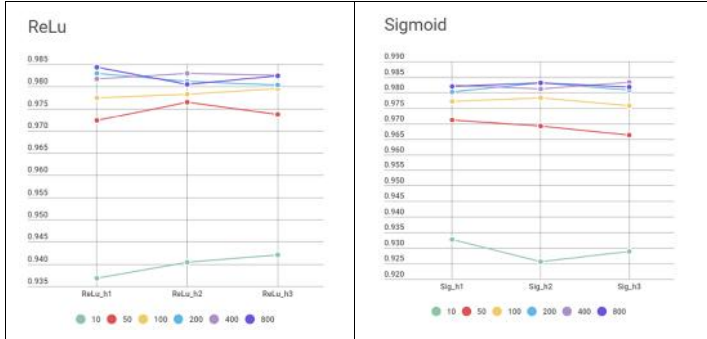
$\hat{x} = \frac{x - avg_{batch}(x)}{stdev_{batch}(x) + \epsilon}$	$BN(x) = \alpha \hat{x} + \beta$
--	----------------------------------

[Batch Normalization Equation]

3. 실험 내용 및 결과

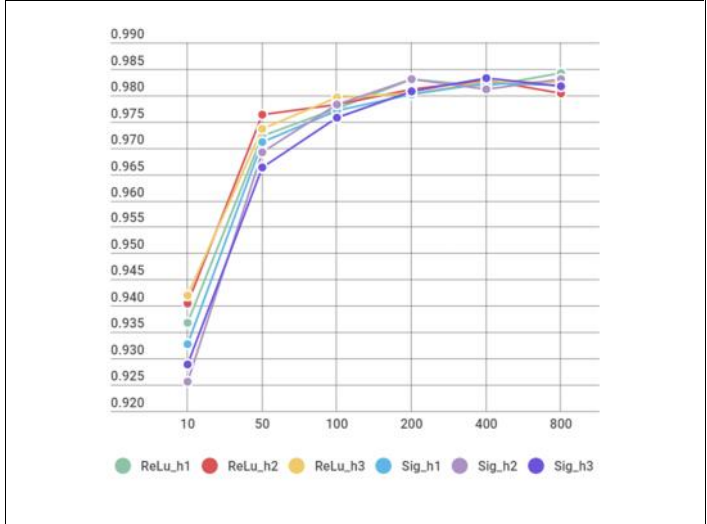
3.1 은닉층, 뉴런의 수에 따른 성능 비교

은닉유닛의 수를 고정시키고, 은닉층의 수에 변화에 따라 성능을 비교해보았다. 성능은 아래의 표와 같다.



[hidden layer on performance]

은닉층이 늘어난다고, 항상 성능향상에 도움이 되는 것은 아니었고, 성능은 은닉 유닛의 수에 더 영향을 많이 받았다. 아래는 은닉 유닛의 수에 따른 성능 비교이다.

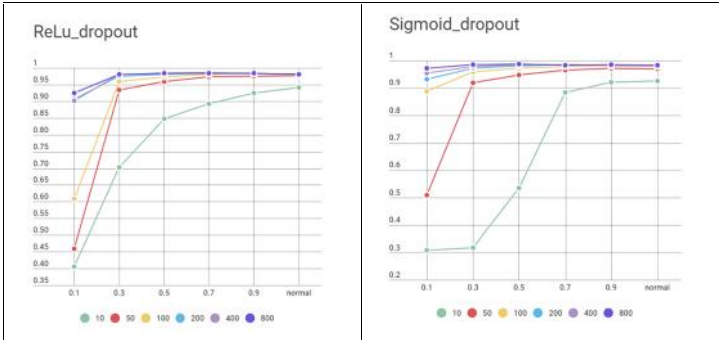


[hidden unit on performance]

또, 은닉 유닛의 수가 100을 넘어선 지점부터는, 은닉 유닛 증가가 성능 개선에 큰 도움을 주지는 못했다.

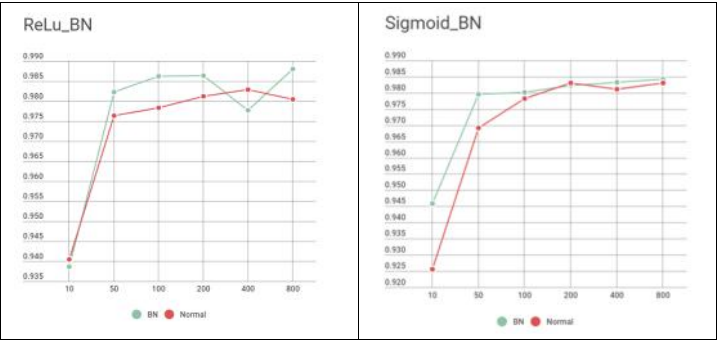
3.2 Drop out, Batch Normalization 성능 비교

과적합 방지와, 성능 개선에 Dropout이 어느정도 도움이 되는지 알아보고자 했다. 아래는 은닉 유닛의 수에 따른, Dropout의 성능을 비교한 표이다.



Dropout 실험은 [0.1, 0.3, 0.5, 0.7, 0.9]의 keep rate로 ReLu와 Sigmoid에 대해서 진행했다. 은닉 유닛이 턱 없이 적은 경우(10개), drop out은 성능 향상에 방해가 되었고, 은닉 유닛의 수가 충분히 많은 경우에만 drop out이 도움이 되었다. 은닉 유닛의 수가 800에 가까워 질수록 기존 연구가 제안하는 drop out rate가 0.5일 때 가장 좋은 성능을 보였다.

Batch Normalization도 drop out과 동일한 방법으로 성능을 비교해 봤다.

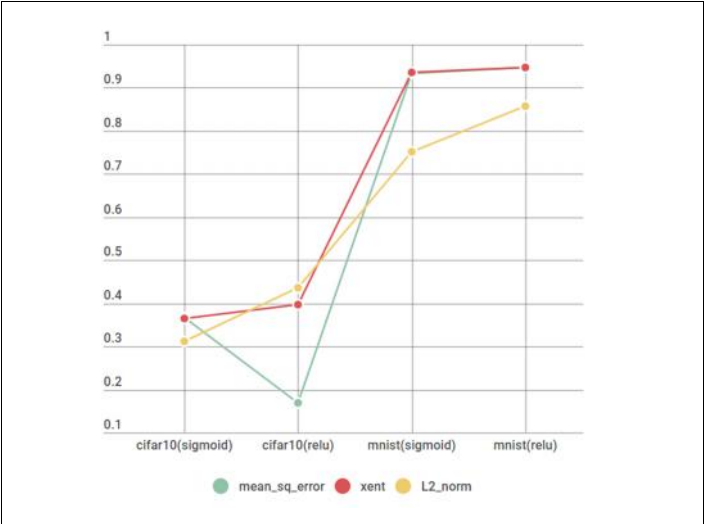


[BN on performance]

BN은 은닉 유닛의 수에 관계없이, 대체적으로 적용했을 때 성능이 개선되었다. 특히 활성화함수가 ReLu일때 성능이 눈에 띄게 향상되었다.

3.3 목표 함수에 따른 성능 비교

Loss 함수로 mean squared error, cross entropy, L2 normalization(with xent)의 3가지를 비교했다.



[objective function on performance]

Cifar10에서는 Regression에 쓰이는 mean squared error가 성능이 가장 나빴으며, cross entropy와 l2 norm은 성능이 비슷했다. 한편, MNIST에서는 mean squared error와 cross entropy가 성능이 비슷했고, 오히려 l2 norm의 성능이 이들 둘에 비해 훨씬 떨어졌다. cross entropy가 본 실험에서 가장 높은 성능을 보였다.

### 3.4 다른 데이터 셋, Cifar10 학습

아래는 MNIST데이터 셋에서 가장 좋은 결과를 냈던 5개의 네트워크 구조이다.

Mnist Top 5 Accuracy				Cifar Top 5 Accuracy	
Activation Function	Regularization	Hidden Layer	Hidden Unit	Accuracy	Iteration
ReLu	batch_normalization	2	800	0.988	9850
ReLu	batch_normalization	3	400	0.9871	15350
ReLu	batch_normalization	3	800	0.9868	7100
Sigmoid	drop_out(0.5)	2	800	0.9867	23050
ReLu	batch_normalization	2	400	0.9864	11500

[MNIST- Top 5 Architecture]

아래는 MNIST에서 좋은 성능을 보여주었던 하이퍼 파라미터의 조합으로 Cifar10을 학습시킨 결과이다.

Mnist Top 5 Accuracy				Cifar Top 5 Accuracy	
Activation Function	Regularization	Hidden Layer	Hidden Unit	Accuracy	Iteration
ReLu	batch_normalization	3	hu_800	0.5842	11400
ReLu	batch_normalization	2	hu_800	0.5757	9400
ReLu	batch_normalization	2	hu_400	0.567	7400
ReLu	batch_normalization	3	hu_400	0.5669	8400
Sigmoid	drop_out(0.5)	2	hu_800	0.4221	18400

앞선 MNIST와는 달리 sigmoid/drop\_out(0.5)/2-layer/800-unit을 쓴 네트워크의 정확도가 다른 네트워크에 비해 현저히 떨어졌다.

## 4. 실험 결과의 분석 내용 및 결론

### 4.1 활성화함수

대부분의 경우에 있어, ReLu가 Sigmoid에 비해 성능이 좋았으며, 특히 층의 개수와 뉴런의 개수가 많아질수록 효과적인 것으로 나타났다.

### 4.2 은닉층, 은닉 유닛의 수

실험 결과, 은닉층의 수보다는 은닉 유닛의 수가 성능에 훨씬 더 많은 영향을 미치는 것으로 파악되었다. 기존 연구에서는 첫 번째 층에서는 947개, 두 번째 층에서는 677개를 쓰도록 제안을 하는데, 실제로 실험에서도 은닉 유닛의 수가 400, 800일 때 성능이 제일 좋았다.

### 4.3 Drop out, Batch Normalization

은닉 유닛의 수가 10개 50개로 비교적 적을 때, Drop out을 쓰는 것은 오히려 정확도를 떨어뜨렸다. 한편

뉴런의 수가 800으로 충분할 때에는 기존 연구가 제안하는 drop out rate 0.5일 때 성능이 제일 좋았다.

### 4.4 Loss function

본 실험에서는 cross entropy가 가장 안정적으로 좋은 성능을 내었다. 특히, cifar10를 학습할 때에 mean squared error를 loss function으로 사용하는 것은 적합하지 않는 것으로 보인다.

## 5. 참고문헌

- [1] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15, 315-323.
- [2] Stathakis, D. (2009). How many hidden layers and nodes? *International Journal of Remote Sensing*, 30(8), 2133-2147.
- [3] Heubert, Jay P, Neisser, Ulric, & Beatty, Alexandra. (2001). Understanding Dropouts: Statistics, Strategies, and High-Stakes Testing.
- [4] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.