2nd Project - 데이터 분석 프로젝트 결과 보고서

12141635 최유정

1. **데이터와 작업 목표**
2. **데이터: MNIST 이미지 데이터세트**
   * 손글씨 숫자 이미지 데이터세트이다. 0부터 9까지의 숫자 이미지로 구성되어 있음.
   * Train용 이미지 60,000장과 Test용 이미지 10,000장으로 구성되어 있고 각 이미지는 그 이미지가 실제 의미하는 숫자가 레이블 데이터를 가지고 있다.
   * 한 장의 이미지 데이터는 28x28 크기의 회색조 이미지여서 768개의 픽셀을 가지고 있고 각 픽셀은 0~255의 값을 취한다.
   * 다음 그림은 MNIST의 샘플 이미지들이다.

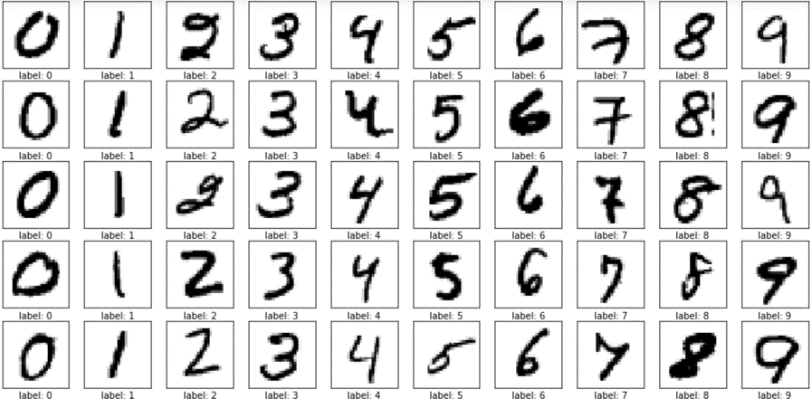
[](http://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjgm4bRs9_bAhXICqYKHZFmByAQjRx6BAgBEAU&url=http://blog.welcomege.com/mnist-database/&psig=AOvVaw2RX3_xwzHO3DhxRB-ZGfYg&ust=1529486539144717)

Figure 1. MNIST 샘플 이미지

1. **작업 목표**

60,000 개의 이미지 데이터와 레이블을 가지고 머신러닝 지도학습으로 학습시켜 입력된 이미지를 0~9까지의 10개의 digit로 분류(Classification)하는 인식률이 높은 분류기(Classifier)를 만드는 것이 작업 목표이다.

1. **구현에 사용된 초기 분석 방법**
2. **추론 모델**

* 모델: Deep Neural Network을 통한 Classification

학습과 추론에 사용되는 모델은 신경망(Neural Network) 아키텍쳐를 사용한다.

* **신경망 계층 수**
* 3개의 서로 다른 계층 수의 신경망 실험
* 은닉계층이 없는 2-layer NN, 은닉계층이 1개인 3-layer NN, 은닉계층이 2개인 4-layer NN으로 확장해가면서 정확도를 높여갈 수 있는 지 확인한다.
* **입출력 뉴런**

입력층 뉴런 수는 784개로의 입력되는 이미지의 픽셀수 (28x28) 이며, 출력층은 0-9까지 숫자의 10개의 분류를 표기하기 위해 10개의 뉴런으로 구성된다.

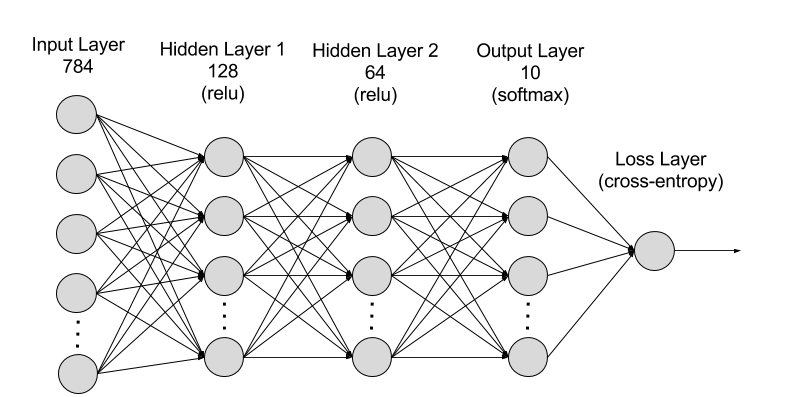
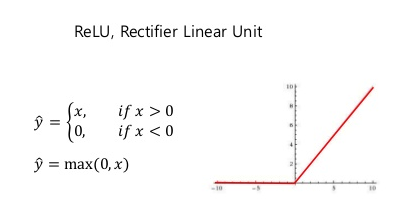
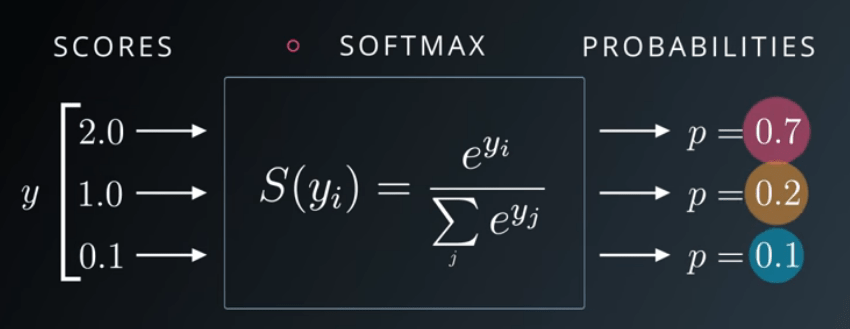
[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjS4tSuwt_bAhUFtJQKHQOMAaIQjRx6BAgBEAU&url=https://mxnet.incubator.apache.org/tutorials/python/mnist.html&psig=AOvVaw1blj_EPs069FUpKYnj-Rms&ust=1529489947628674)

Figure 2. 다중 계층 신경망 예시

* **Activation Fuction**

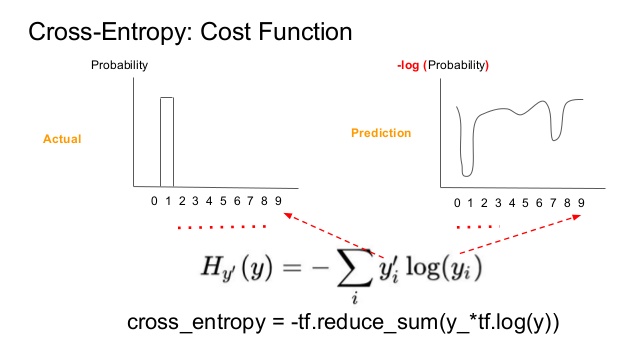
각 Layer의 Activation 함수는 은닉계층은 Normalized 되지 않아도 되므로 **ReLU**함수로 사용하고, Output Layer의 경우 10개의 가능한 출력값에 대한 확률 분포를 사용하는 것이 적합하므로 **Softmax** 함수를 사용한다.



[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiSv5GTyN_bAhVLAogKHQgZC6AQjRx6BAgBEAU&url=https://sefiks.com/2017/11/08/softmax-as-a-neural-networks-activation-function/&psig=AOvVaw1nKEwqEuRbnR_3qvYLzMqk&ust=1529492003972962)

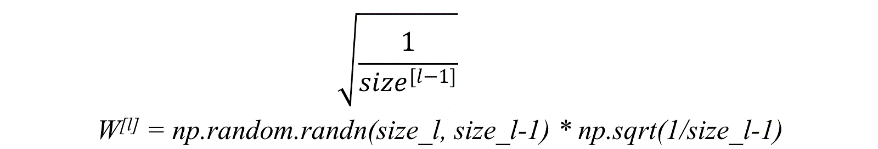
* **Cost Function**

출력 층의 Loss function은 Softmax로 구한 확률 벡터와 레이블 벡터간의 거리를 측정하기 위해 **Cross-entropy cost fuction**을 사용한다

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi0rO7Tyd_bAhUUFYgKHfIpAMQQjRx6BAgBEAU&url=https://www.slideshare.net/KhorSoonHin/gentlest-introduction-to-tensorflow-part-3&psig=AOvVaw3SmHp53eseVoyu6zKZc2um&ust=1529492455798244)

* **가중치(Weight)의 초기값 분포**

이 모델에서는 정규분포범위의 값을 랜덤하게 할당보다는 activation 함수로 선택한 ReLU에 가장 적합하다고 알려진 **He 초기값**을 사용하기로 한다. He 초기값은 앞 계층의 노드가 n개일 때 평균이 0이고 표준편차가 인 정규분포를 사용한다. He 분포 범위의 값을 랜덤하게 초기값으로 할당한다.

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiOw5qFtuHbAhUTtJQKHRTjB34QjRx6BAgBEAU&url=https://towardsdatascience.com/deep-learning-best-practices-1-weight-initialization-14e5c0295b94&psig=AOvVaw1-SzXyI62t-g-CZzwcC4fv&ust=1529555902431017)

다만, Bias 초기값은 원래 소스처럼 표준편차 0.1의 정규 분포의 랜덤 값으로 할당한다.

* **Hyper Parameters**

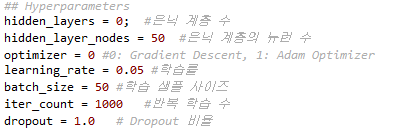
학습 속도와 인식의 정확도를 높이기 위해 실험에서 바꿔가면서 튜닝하는 하이퍼 파라미터는 다음과 같다.

* 최적화 알고리즘: SGD(**Stochastic** Gradient Descent) vs **Adam**(Adaptive Moment Estimation)
* 학습속도(Learning rate): 0.005 ~0.02
* Batch size: 60,000개 샘플 중 한번의 학습Iteration에 사용될 이미지 샘플수
* 학습 iteration 수: 500 ~ 2000
* 과적합 해소를 위한 Dropout 비율: 0% ~ 30%

1. **소스코드 및 결과 화면 설명**
   1. **소스코드 설명**

소스코드의 기본 틀은 수업시간에 제공된 파이선 MNIST example 코드를 기반으로 새로운 모델에 맞게 변경하였다. 주요 변경 내용은 다음과 같다.

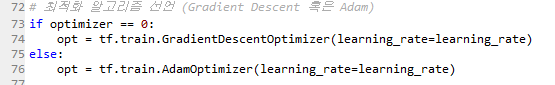
* 7개의 HyperParameter를 설정하였다.



* Neural Network의 Hidden Layer(=0, 1, 2)가 Hyperparameter 설정에 따라 동작할 수 있도록 구현하였다. 은닉계층의 뉴런수는 hidden\_layer\_nodes 수에 따라 조절이 가능하다.



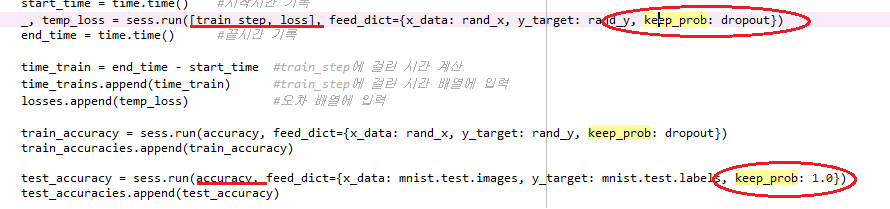
* Optimizer를 Gradient Descent 와 Adam 을 선택하여 실행할 수 있도록 구현



* 과적합(Overfitting) 해소를 위해 dropout 비율을 하이퍼파라미터로 설정하고, train 과정에는 dropout 비율을 적용하고 test 과정에서는 dropout을 하지 않게 하였다.





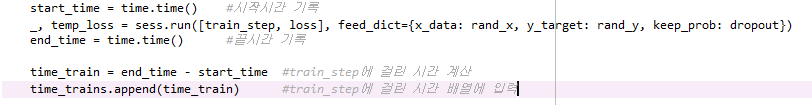


* 가중치의 초기값을 ReLU Activation에 적합하다고 알려진 He Initializer로 초기화 함. 편향값은 정규분포의 랜덤 값으로 설정





* 학습에 소요된 총 시간을 계산하는 기능을 구현



* Accuracy 그래프에 Optimizer의 performance를 측정하기 위해 loss 함수의 그래프도 추가하였고, 그래프에 결과값이 Max Accuracy와 입력된 HyperParameter들도 표시하였다.



* 학습이 완료된 신경망에 테스트데이터세트 이미지를 입력하여 레이블을 예측한 값을 이미지와 함께 보여주는 기능을 구현하였다.



* 1. **프로그램 실행 결과 화면**

실행 결과화면에는 loss 그래프를 추가하고 그래프만으로도 정확도를 판별할 수 있도록 내에 학습결과(정확도와 학습시간), 하이퍼 파라미터를 표시하였다. 또한 10개의 시험이미지에 대한 예측 결과를 이미지로 표현하였다.

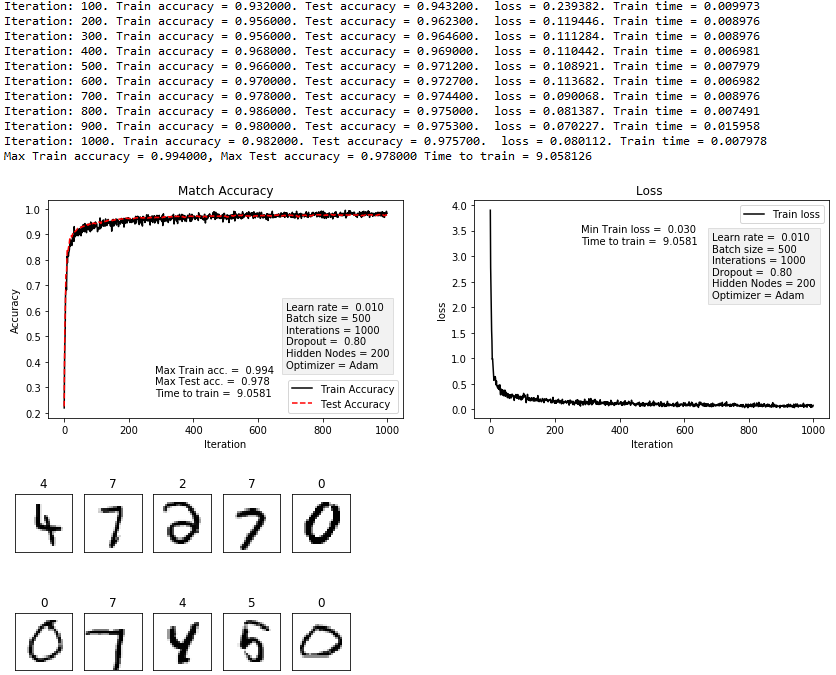


Figure 3. 프로그램 실행 결과 화면

1. **분석 결과**
   1. **단일층 신경망**

입력층과 출력층만 존재하는 신경망을 학습시킨 후 정확도를 측정하였다. 배치사이즈는 500개, 반복학습 iteration 수는 1,000으로 고정하고 학습속도(learning rate)와 손실 최소화 알고리즘 변경(Gradient Descent, Adam)에 따른 정확도와 손실오차가 어떻게 변하는 지 관찰하였다. 은닉계층이 없으므로 Hidden Layer Node수는 고려되지 않는다. 학습과 정확도 측정 결과는 다음과 같다.

* + 1. 학습속도(learning rate) 변화에 따른 정확도 및 학습 속도의 변화
* Learning rate = 0.01, 0.05, 0.10, 0.50, 1.00, 5.00, 10.00와 Optimizer = Gradient Descent의 하이퍼파라미터 사용하여 실험
* 아래 결과 그림에서 보듯이, Learning rate가 5가 넘어가면 왜곡이 심해지고 1.0에서도 학습초기 fluctuation이 있음을 볼 수 있다. Test accuracy가 가장 높은 것은 1.0, 다음은 0.5 임을 알 수 있다.

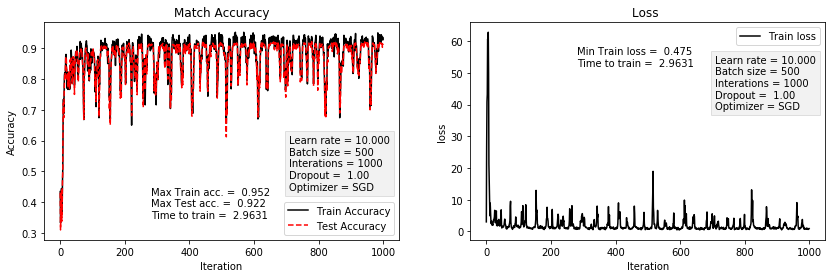
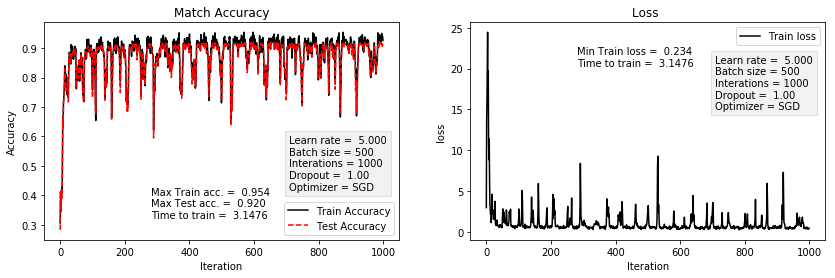
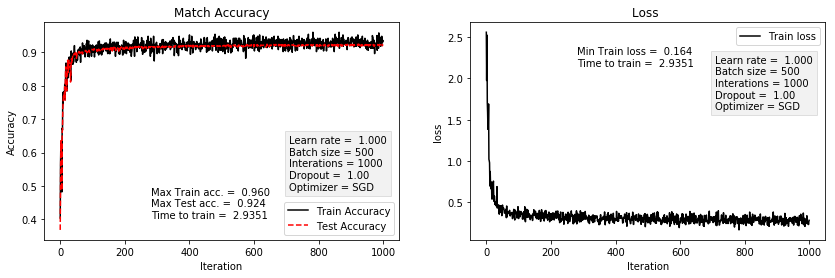
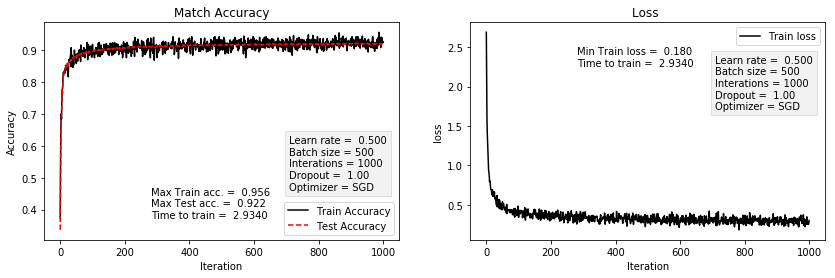
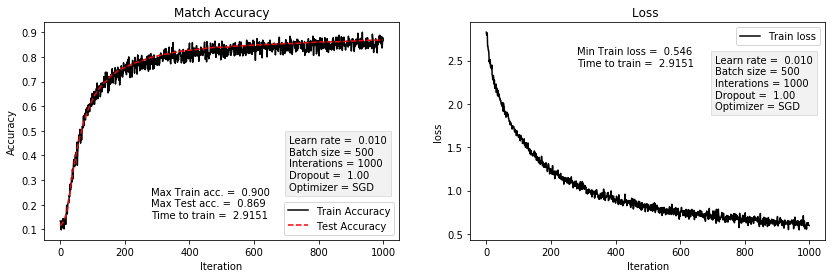
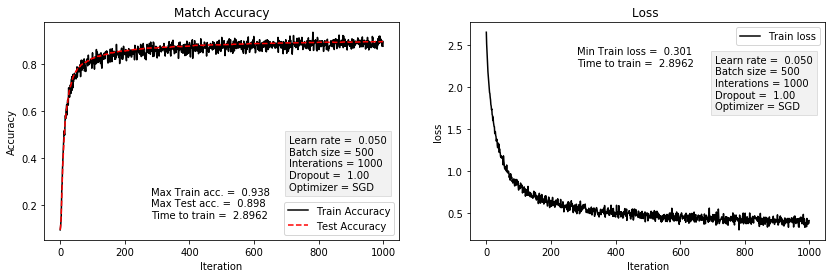
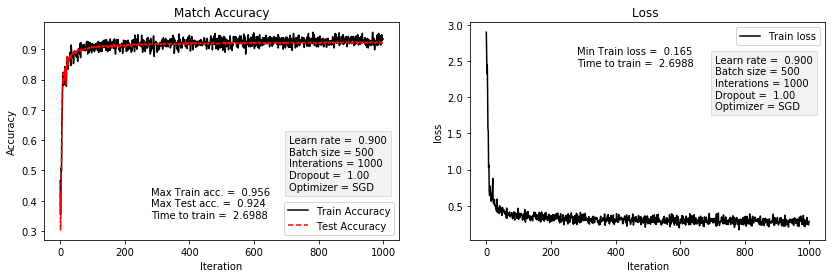


Figure 4. 단일 계층 신경망, Gradient Descent 조건에서 최적 학습율 측정 결과

* Learning rate: 0.5 ~ 1.0으로 재시도하여 보았다. 대체로 정확도와 학습속도는 비슷하였고 학습속도가 0.9일 때 가장 loss최소값이 최소(0.165)가 되고 변동성 없이 찾아가면서 정확도(학습정확도 95.6%, 테스트 정확도 92.4%)가 높은 것으로 측정되었다.



* + 1. Optimizer변경 후 Leaning rate 선택에 따른 정확도 측정

마찬가지 방법으로 Adam Optimizer로 선택한 후 learing rate의 변화에 따른 정확도 변화를 확인하였다.

Loss의 최저값이 제일 작게 내려간 것은 학습속도 0.01인 경우였으며, 이때의 학습정확도(96.0%)와 시험정확도(92.8%)도 가장 높았음을 알 수 있다.

학습속도가 느린 경우 (learing\_rate = 0.01)는 최저값에 도달하는 시간도 늦을 뿐 아니라 최종 학습 후 최저값도 높았다. 이 경우 iteration을 높이면 최저값을 낮출 수 있으나 학습시간이 오래 걸릴 것으로 판단된다.

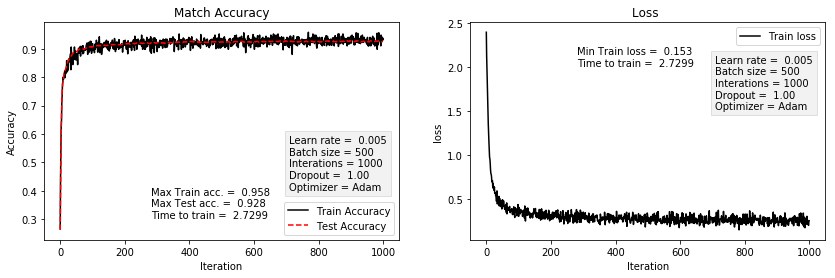
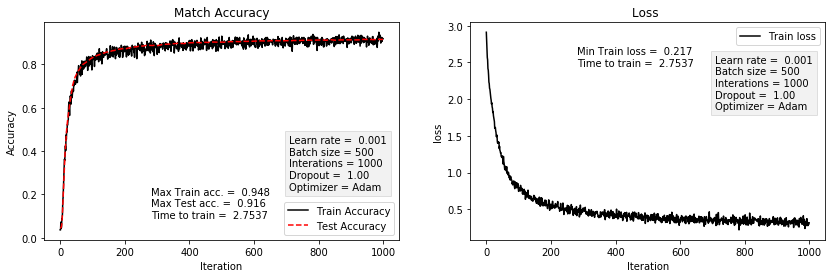
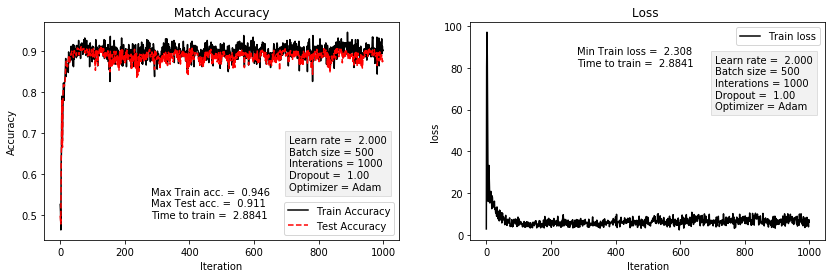
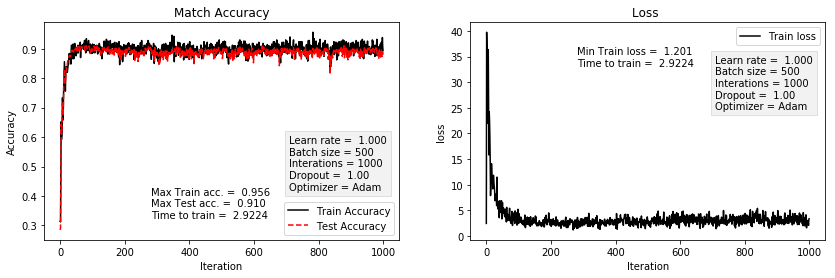
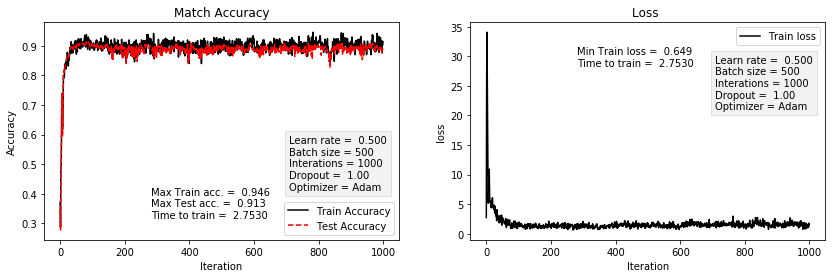
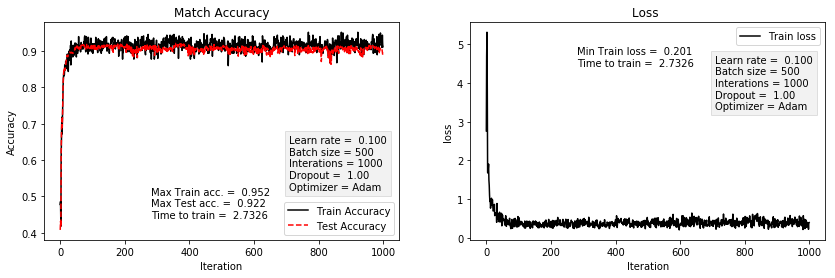
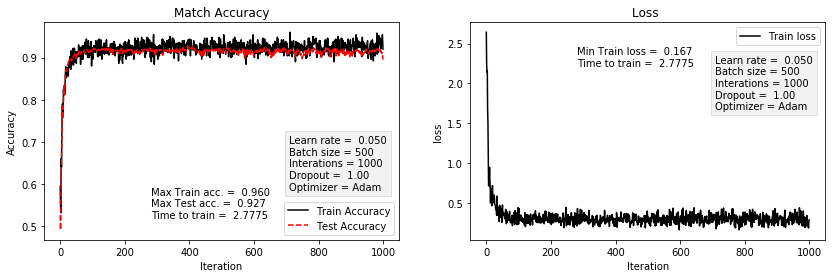
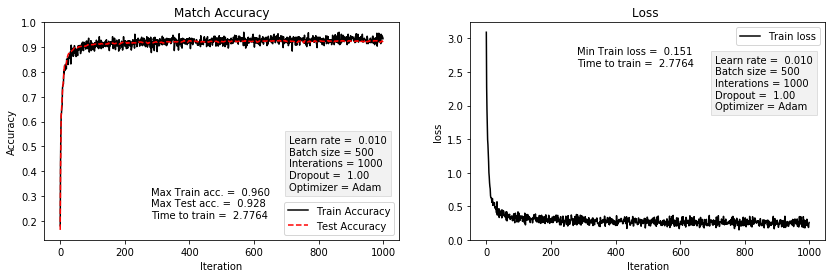


Figure 5. Adam Optimizer, 단일계층 신경망의 최적 학습률 측정 결과

* + 1. 두 Optimizier 간의 최고 매칭율 비교.

각 최적화 수단에 따른 학습률과 매치정확도를 다음 표로 비교해 보았다. 두 가지 최적화 알고리즘에 따른 학습과 시험 정확도의 차이는 크게 차이가 나지 않았으나 Adam의 경우가 최저값에 도달하는 속도가 좀 더 빠르고 정확도가 다소 높은 것으로 나타났다

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Optimizer | 최적  학습률 | 학습  정확도 | 시험  정확도 | 최저  오차값 | 학습수행시간(1000반복, 500샘플) |
| Gradient Descent | 0.9 | 95.6% | 92.4% | 0.165 | 2.69 sec |
| Adam | 0.01 | 96.0% | 92.8% | 0.151 | 2.77 sec |

표1. Optimizer간 최적 학습율과 학습정확도

* 1. **2계층 신경망 (Hidden Layer = 1)**

1개의 은닉 계층으로 learning\_rate를 바꾸어 가며 시험해 보았다. Optimizer는 Adam으로 고정하였다.

* + 1. 학습속도 변화에 따른 정확도 측정 (Optimizer = Adam으로 고정)

학습 속도를 단일층 신경망에서 얻은 0.01 주변(0.001~0.100)으로 범위를 국한하여 학습시킨 후 정확도 등을 측정한 결과, 단일층 신경망에서 얻었던 **0.01**의 학습속도에서 가장 적은 손실 오차(0.024)와 가장 높은 매칭 정확도(학습시 99.8%, 시험시 97.1%)를 나타냈다. 단일 계층 신경망과 2계층 신경망의 최고 매칭율을 보이는 학습율을 비교하여 보았다. 정확도가 상당히 높아졌으나 학습시간은 2배 가량 더 걸리는 결과가 나왔다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 신경망 종류 | 최적  학습률 | 학습  정확도 | 시험  정확도 | 최저  오차값 | 학습수행시간(1000반복, 500샘플) |
| 단일신경망(hl=0) | 0.01 | 96.0% | 92.8% | 0.151 | 2.77 sec |
| 2계층신경망(hl=1) | 0.01 | 99.8% | 97.1% | 0.024 | 5.24 sec |

표2. 은닉 계층 추가에 따른 학습수행시간, 최저오차값, 학습정확도, 최적 학습률 비교

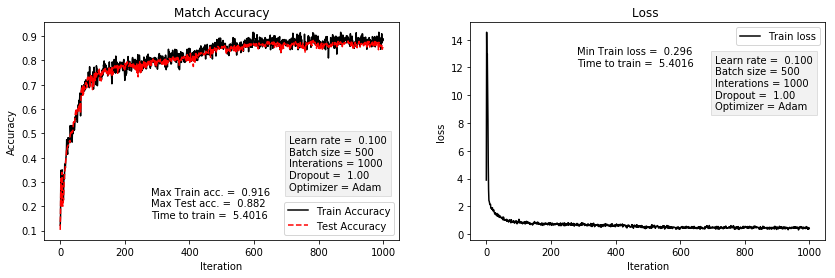
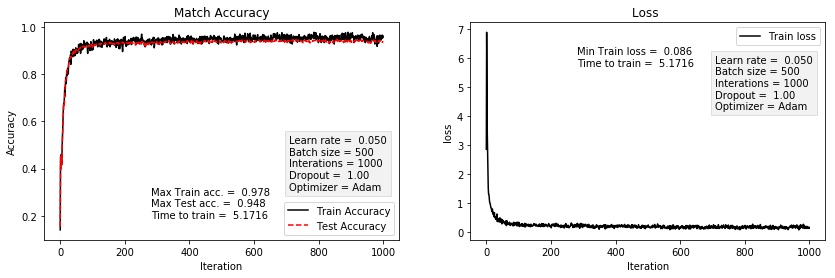
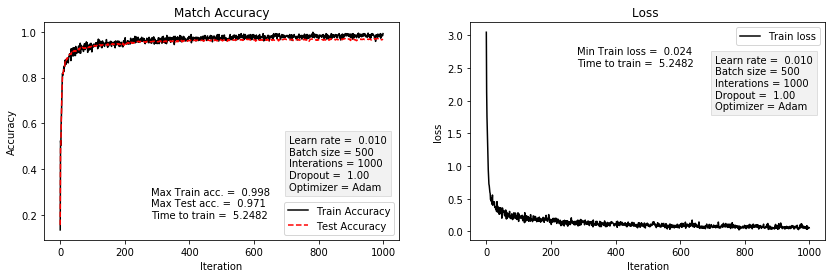
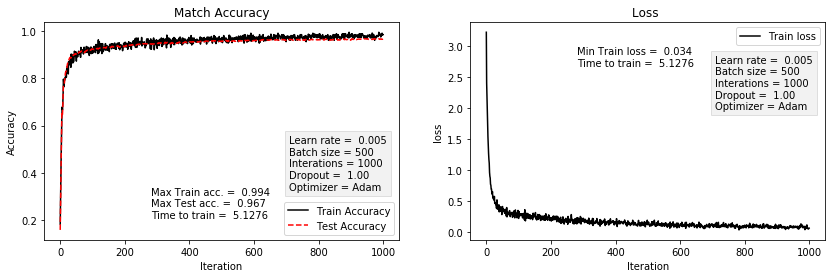
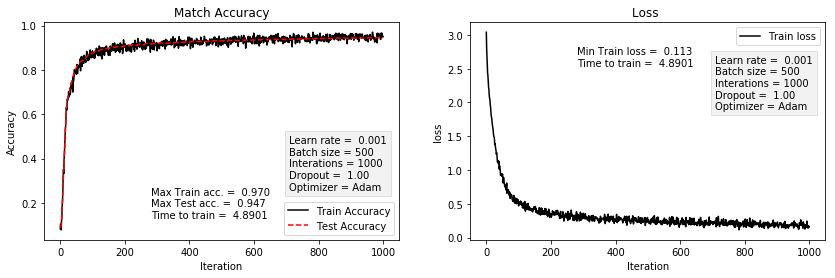


Figure 6. 2계층 신경망의 학습속도(learning rate)변화에 따른 정확도 및 손실값 측정

* + 1. 은닉계층 뉴런 수에 따른 매칭 정확도 측정

학습속도는 앞에서 구한대로 0.01로 고정한 후, 은닉계층의 뉴런 수를 50에서 100, 150, 200으로 늘려가며 정확도 개선 효과가 있는 지 측정하였다. 결과는 다음 표 처럼 노드수가 늘어나면 매칭 정확도가 소폭 높아지고 최소 오류값은 점점 줄어드는 것을 볼 수 있으나, 학습시간은 더 길어지는 것을 확인할 수 있다,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 은닉계층 노드수  (2계층 신경망) | 최적  학습률 | 학습  정확도 | 시험  정확도 | 최저  오차값 | 학습수행시간(1000반복, 500샘플) |
| 50 | 0.01 | 0.992 | 0.966 | 0.043 | 4.89 |
| 100 | 0.01 | 1.000 | 0.974 | 0.021 | 6.12 |
| 150 | 0.01 | 1.000 | 0.977 | 0.010 | 7.59 |
| **200** | **0.01** | **1.000** | **0.978** | **0.007** | **8.69** |

표3. 은닉계층 노드수 변화에 따른 학습정확도 및 학습시간 측정

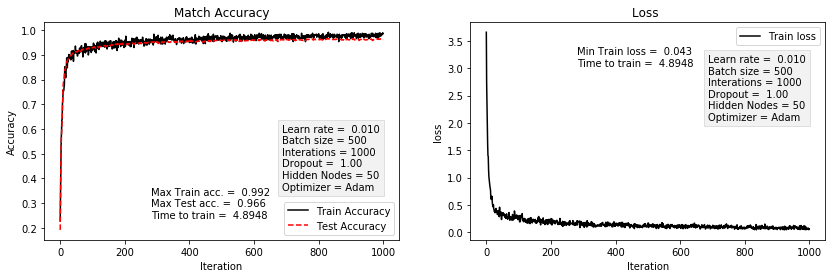
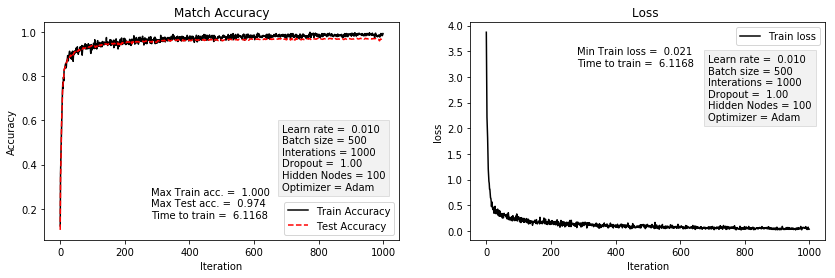
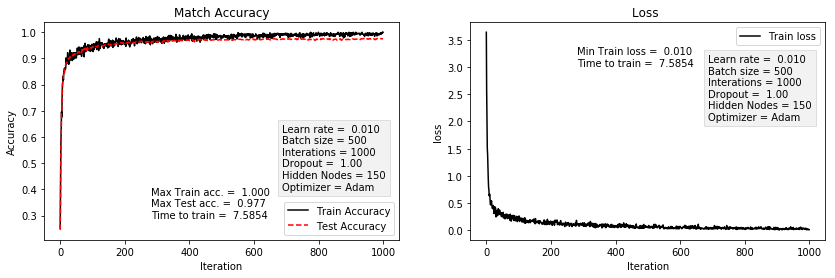
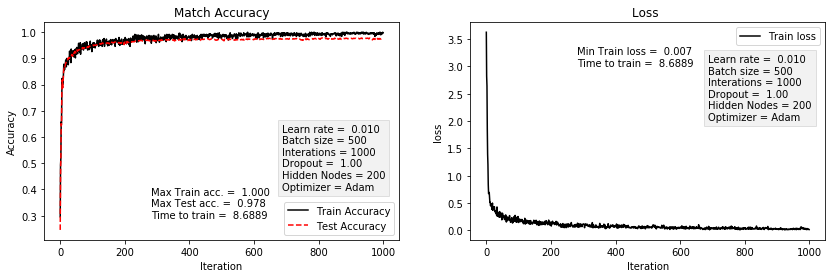


Figure 7. 2계층 신경망의 은닉계층 노드수 변화에 따른 정확도 및 손실값 측정

* 1. **3계층 신경망 (Hidden\_layers = 2)**

은닉계층을 하나 더 늘려서 매치 정확도를 향상 시킬 수 있는 지 시험을 해 보았다. 학습 진행 결과, 은닉 계층의 노드수가 같을 때 매치 정확도가 크게 개선되지 않고 극히 미미하게 개선된다는 결과가 나왔다. 그러나 학습수행시간은 더 오래 걸려 약 22%의 비용이 더 필요한 것으로 측정되었다. 2계층 신경망과 3계층 신경망의 학습결과를 다음 표에 정리해 보았다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 은닉계층  노드수 | 학습정확도 | | 시험정확도 | | 최저오차값 | | 학습수행시간 | |
| 2계층 | 3계층 | 2계층 | 3계층 | 2계층 | 3계층 | 2계층 | 3계층 |
| 50 | 0.992 | 0.996 | 0.966 | 0.968 | 0.043 | 0.037 | 4.89 | 5.88 |
| 100 | 1.000 | 1.000 | 0.974 | 0.975 | 0.021 | 0.008 | 6.12 | 7.65 |
| 150 | 1.000 | 1.000 | 0.977 | 0.978 | 0.010 | 0.013 | 7.59 | 9.07 |
| **200** | **1.000** | **1.000** | **0.978** | **0.977** | **0.007** | **0.007** | **8.69** | **11.04** |

표4. 2계층 신경망과 3계층 신경망의 은닉계층 노드수 변화에 따른 학습정확도 및 학습시간 측정 비교

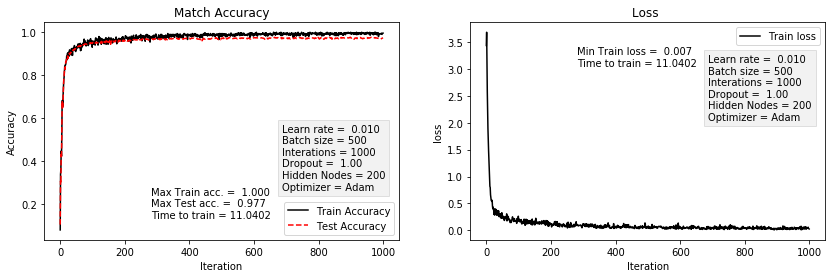
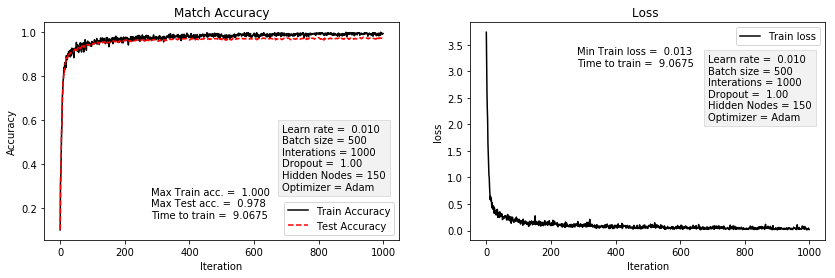
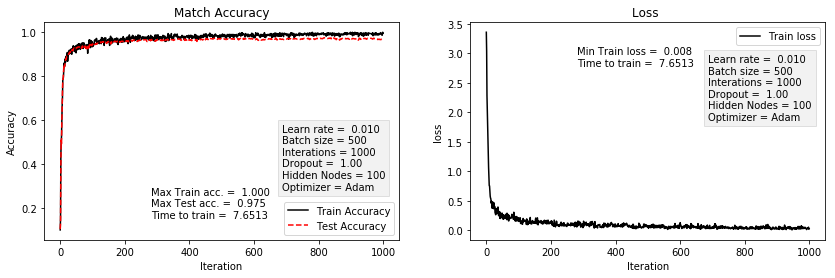
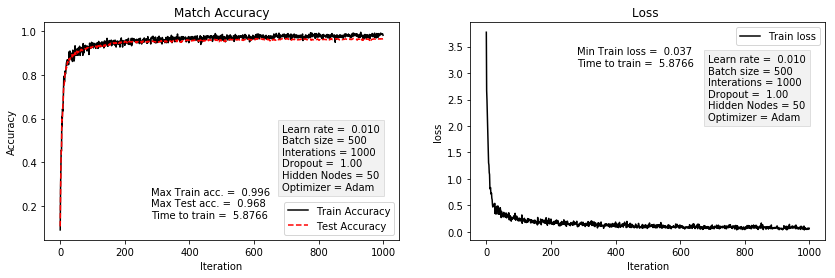


Figure 7. 3계층 신경망의 은닉계층 노드수 변화에 따른 정확도 및 손실값 측정

결론적으로, 150개 이상의 노드 수를 가진 2계층 신경망과 3계층 신경망에서 100%의 학습 매칭 정확도와 97.78%의 시험 매칭 정확도로 가장 높은 성능을 보이는 것으로 나타났다. 다만 **학습 속도를 고려하면, 200개의 노드를 가진 2계층 신경망이 최적의 모델로 판단된다.**

* 1. **2계층 신경망의 과적합 문제 해결**

2~3계층 신경망의 노드 수가 증가할수록 학습 정확도와 시험 정확도의 차이가 iteration이 진행될 수록 벌어지는 경향이 나타난다. 이러한 과적합(Overfitting) 문제를 dropout 비율을 조정해 가면서 개선될 수 있는 지 확인해 보았다. 결과는 학습 정확도와 시험 정확도의 차이는 아래 그래프처럼 많이 개선되었다. 하지만, 학습 정확도만 낮아져 격차가 줄어들 뿐, 시험 정확도는 크게 차이가 나지 않는다. 아래 2계층 NN의 2가지 dropout 시험 결과를 그래프로 표시하였다.

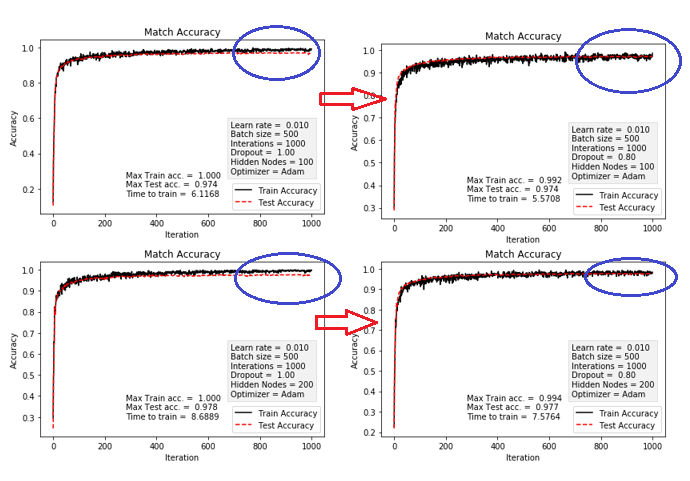


Figure 8. 2계층 신경망의 은닉계층 노드 드롭아웃을 통한 과적합 문제 해결(노드사용율 80%)

잠정적으로 결정한 최종 모델인 노드수 200 인 경우 시험 정확도는 0.977로 Dropout 했을 경우와 드롭아웃이 없는 경우(0.978)과 거의 같다. 단지 학습 정확도가 1.000에서 0.994로 오히려 낮아졌다. 이는 드롭아웃으로 학습의 속도가 느려졌기 때문으로 판단된다. Iteration 수 1,000~4,000까지 늘려가면서 학습을 더 많이 진행하여 결과를 살펴보았다. 결과는 아래 표에서 보듯이 과적합 문제도 줄어들면서 시험정확도가 지금까지 최고 수치인 **0.982**로 높아진 것을 볼 수 있다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복학습횟수 | 노드사용율 | 학습정확도 | 시험정확도 | 최저오차값 | 학습수행시간 |
| 1,000 | 1.0 | 1.000 | 0.978 | 0.007 | 8.69 |
| 1,000 | 0.8 | 0.994 | 0.977 | 0.030 | 7.58 |
| 2,000 | 0.8 | 0.998 | 0.980 | 0.010 | 14.94 |
| 3,000 | 0.8 | 1.000 | 0.981 | 0.006 | 22.42 |
| 4,000 | 0.8 | 1.000 | 0.982 | 0.005 | 31.32 |

표5. 200개 노드를 가진 2계층 신경망의 노드사용율 0.8일때 반복학습횟수별 학습결과 측정

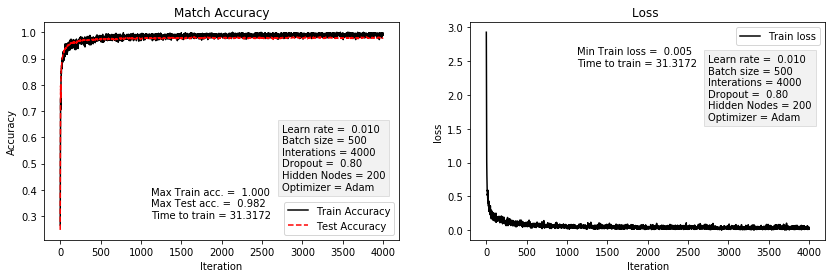
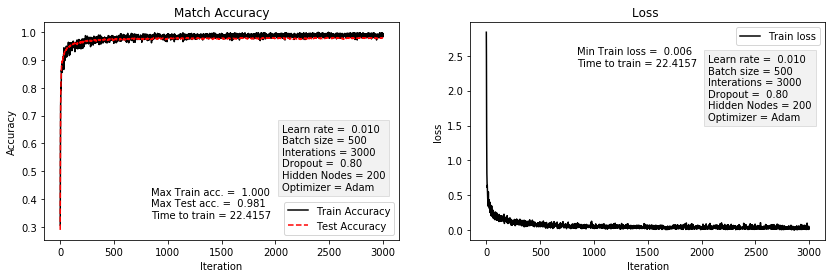
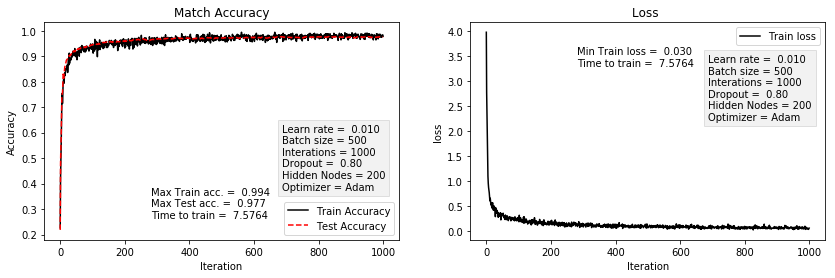


Figure 9. 은닉계층이 드롭아웃된 학습의 반복횟수 증가에 따른 정확도 측정

(노드수 200,노드사용율 80%, 반복횟수 1,000~4,000)

그러나, 드롭아웃을 하지 않고 단순히 반복 학습 횟수만 4,000으로 늘렸을 때는 오히려 시험 정확도가 0.980으로 낮아지고 시험 정확도와 학습 정확도의 차이가 벌어지는 것을 알 수 있었다. 단순히 반복학습 횟수를 늘리는 것이 정확도를 높이지는 않는 것을 알 수 있었다.

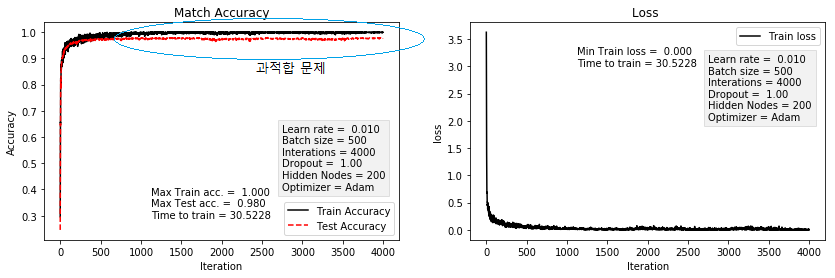
****

Figure 10. 드롭아웃 없는 학습의 반복횟수 증가에 따른 정확도 측정(노드 200, 반복횟수 4,000)

1. **최종 최적화된 모델**

실험 결과 최종 최적화된 모델은 다음과 같이 결정하였다.

1. **학습모델**

다중 신경망 모델 (2계층: Hidden Layer = 1)

1. **모델 파라미터**

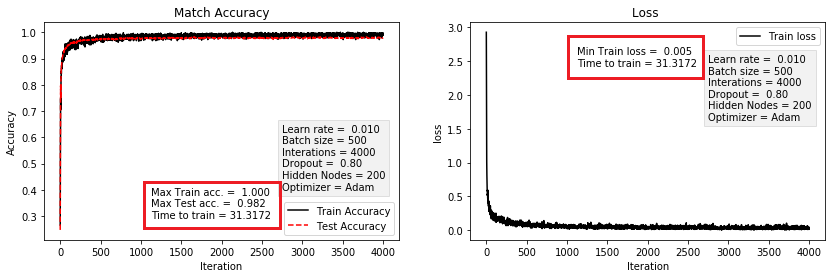
|  |  |
| --- | --- |
| 하이퍼파라미터 | 값 |
| 노드수 | 입력층: 780  **은닉층: 200**  출력층: 10 |
| 은닉층 Activation 함수 | **ReLU** |
| 출력층 Activation 함수 | **Softmax** |
| 손실함수 | **Cross Entropy** |
| 손실값 최소화 알고리즘 | **Adam Optimizer** |
| 가중치 초기값 분포 | **He normal**  ReLU에 적합, 표준편차 (은닉계층:, 출력계층: ) |
| 학습율(Learning rate) | **0.01** |
| 학습 배치 사이즈(batch size) | **500** |
| 반복학습수 (iteration) | **4,000** |
| 드롭아웃에서 사용할 뉴런의 비율 | **0.8 (80%)** |

표6. 최종 최적화된 모델의 하이퍼파라미터

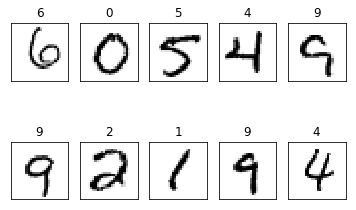
1. **학습 결과**

**시험 정확도: 0.982 (98.2%), 손실오차: 0.005**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 반복학습횟수 | 드롭아웃비율 | 학습정확도 | 시험정확도 | 최저오차값 | 학습수행시간 |
| 4,000 | 0.8 | 1.000 | 0.982 | 0.005 | 31.32초 |



최종 최적화된 모델로 학습 완료 후 10개의 시험데이터세트의 이미지들의 레이블의 예측한 결과를 그림으로 표시해 보았다. 정확하게 인식된 것을 볼 수 있다.



1. **결론**

이번 실험을 진행하면서, Deep Neural Network의 개념을 이해하게 되었다. 머신러닝에 많이 사용되고 있는 Python 언어와 구글의 Tensorflow의 기초 활용법도 익히게 되었다.

다중 신경망 모델을 통해 데이터를 분석(Classification, 추론, Clusteriung)작업 수행하는 데에는 가중치 초기값 설정, 배치 정규화, 활성화 함수(Sigmoid, ReLU 등), 손실함수, 손실 오류 최소화 알고리즘(Gradient Descent, Adam 등), 과적합과 그 해결 방법 등 고려사항 들이 많고 다양하게 연구되고 있다는 것을 알게 되었다. 또한 그러한 기법이나 알고리즘도 중요하지만, 정확도를 높이고 오차를 줄이는 최적화 작업이 정해진 방법론 보다는 많은 시간과 노력, 그리고 데이터에 대한 이해와 시행착오, 직관 등이 복합적으로 필요한 매우 어려운 작업이라는 것을 알게 되었다. 실생활에 존재하는 다양한 데이터들을 분석하여 가치 있는 예측을 할 수 있도록 다양한 분석 모델을 경험해 보고 싶다.