I. NEURAL NETWORK

In this question, the final cost function is:

$$J(\omega) = l(\omega) + \lambda(||\omega^{(1)}||_F^2 + ||\omega^{(2)}||_F^2) \tag{1}$$

In this case, the analytical expression for the gradient $\Delta_{\omega_1} J(\omega)$ and $\Delta_{\omega_2} J(\omega)$ can be calculated as follows:

A. 2nd Layer NNW back propagation

$$\Delta_{\omega_{kj}^{(2)}} J(\omega) = \frac{\partial l}{\partial \omega_{kj}^{(2)}} + \frac{\partial \lambda |\omega^{(2)}|_F^2}{\partial \omega_{kj}^{(2)}} \tag{2}$$

$$= \frac{\partial l}{\partial h_k} \frac{\partial h_k}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial \omega_{kj}^{(2)}} + 2\lambda \omega_{kj}^{(2)} \tag{3}$$

$$\frac{\partial l}{\partial h_k} = \Sigma_{n=1}^{n=N} \left[-\frac{y_{kn}}{h_{kn}} + \frac{1 - y_{kn}}{1 - h_{kn}} \right] \tag{4}$$

$$\frac{\partial h_{kn}}{\partial a_{kn}^{(2)}} = \frac{\partial (1 + e^{-a_{kn}^{(2)}})^{-1}}{\partial a_{kn}^{(2)}} = \frac{e^{-a_{kn}^{(2)}}}{(1 + e^{-a_{kn}^{(2)}})^2} \tag{5}$$

$$\frac{\partial a_{kn}^{(2)}}{\partial \omega_{jkn}} = z_{jn} \tag{6}$$

Therefore, combining the above four equations we have:

$$\Delta_{\omega_{kj}^{(2)}} J(\omega) = \Sigma_{n=1}^{n=N} \left[(h_{kn} - y_{kn}) z_{jn} \right] + 2\lambda \omega_{kj}^{(2)} \tag{7}$$

B. 1st Layer NNW back propagation

$$\Delta_{\omega_{ji}^{(1)}} J(\omega) = \frac{\partial l}{\partial \omega_{ji}^{(1)}} + \frac{\partial \lambda |\omega^{(1)}|_F^2}{\partial \omega_{ji}^{(1)}} \tag{8}$$

$$= \frac{\partial l}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial \omega_{ji}^{(1)}} + 2\lambda \omega_{ji}^{(1)} \tag{9}$$

$$\frac{\partial a_{jn}^{(1)}}{\partial \omega_{ijn}} = x_{in} \tag{10}$$

$$\frac{\partial l}{\partial a_j^{(1)}} = \Sigma_{k=1}^{k=K} \frac{\partial l}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial a_j^{(1)}} \tag{11}$$

$$= \Sigma_{n=1}^{n=N} \Sigma_{k=1}^{k=K} \left[(h_{kn} - y_{kn}) \cdot \omega_{jk}^{(2)} \cdot \frac{e^{-a_{jn}^{(1)}}}{(1 + e^{-a_{jn}^{(1)}})^2} \right] \tag{12}$$

Therefore, the final analytical form of the gradient is:

$$\Delta_{\omega_{kj}^{(2)}} J(\omega) = \Sigma_{n=1}^{n=N} \Sigma_{k=1}^{k=K} \left[(h_{kn} - y_{kn}) \cdot \omega_{jk}^{(2)} \cdot \frac{e^{-a_{jn}^{(1)}}}{(1 + e^{-a_{jn}^{(1)}})^2} \right] \tag{13}$$

$$\cdot x_{in} + 2\lambda \omega_{ji}^{(1)} \tag{14}$$

C. Numerical Implementation

Numerically, it is computationally faster to implement the above analytical form into matrix multiplication, the full pseudo code is listed below:

1. **Initialization**:
   X,Y, stepsize,
   ω_1=rand(n$_{hid}$,x$_{dim}$)-0.5;
   ω_2=rand(y$_{dim}$,n$_{hid}$)-0.5.
2. **Forward Propagation**:
   a$_1$=ω_1*x$_{sample}$;
   z=activation(a$_1$);
   a$_2$=W$_2$*z;
   h=activation(a$_2$);
3. **Backward Propagation**:
   dω_2=$(h - y_{sample}) * z + 2\lambda * \omega_2$;
   dω_1=$(((h - y_{sample}) * \omega_2). * (exp(-a_1)./((1 + exp(-a_1)).^2))) * x_{sample} + 2 * \lambda * \omega_1$;
4. **Gradient Descent**:
   $\omega_1 = \omega_1$−stepsize$*d\omega_1$;
   $\omega_2 = \omega_2$−stepsize$*d\omega_2$;
   where $*$ represent the matrix product, while $.*$ represent the item-to-item product.

D. Batch/Stochastic Gradient Descent

In batch gradient descent, the gradient at each gradient descent is averaged over the calculated gradient (at the same evaluation point) for the entire dataset. In another word, step 3 above is the average of all $d\omega_1$ and $d\omega_2$ for all data points.
For stochastic gradient descent, the gradient descent is carried out after the gradient is evaluated for each data point.

E. Toy data

1. number of hidden nodes

For toy dataset1, two hidden units is sufficient to give a model with accuracy on testing dataset. Figure 1 shows the decision boundaries for different hidden units, figure 2 shows error rates for different hidden units.

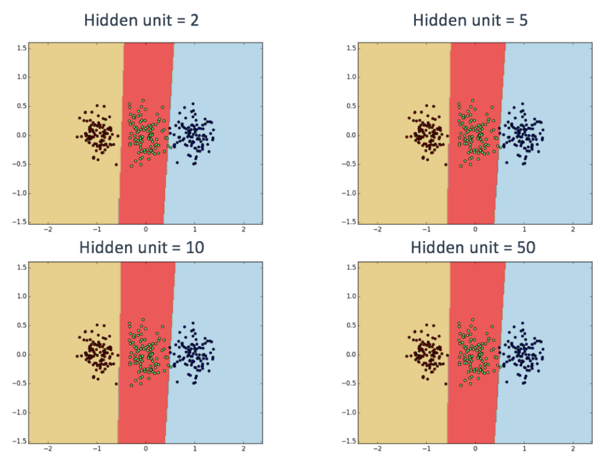
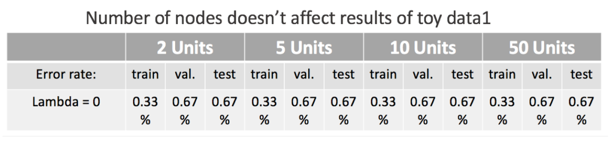


FIG. 1: Decision boundaries for toy data1



Number of nodes doesn't affect results of toy data1

	2 Units			5 Units			10 Units			50 Units		
Error rate:	train	val.	test	train	val.	test	train	val.	test	train	val.	test
Lambda = 0	0.33 %	0.67 %	0.67 %	0.33 %	0.67 %	0.67 %	0.33 %	0.67 %	0.67 %	0.33 %	0.67 %	0.67 %

FIG. 2: Error rates for toy data1

For toy dataset2, five hidden unit are required for a good model. Figure 3 shows the decision boundaries for different hidden units, figure 4 shows error rates for different hidden units.

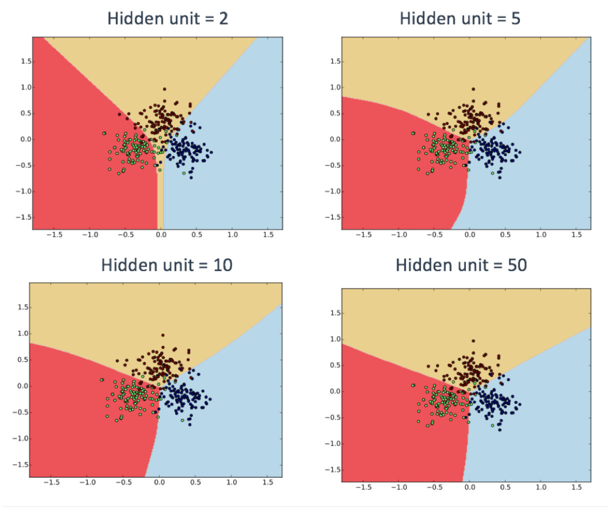
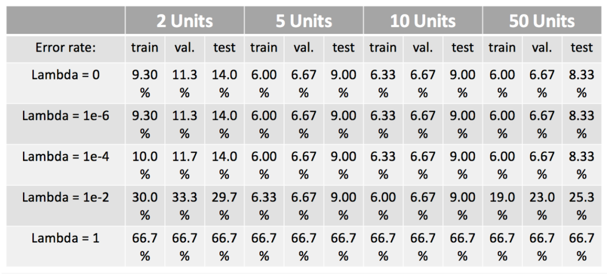


FIG. 3: Decision boundaries for toy data2



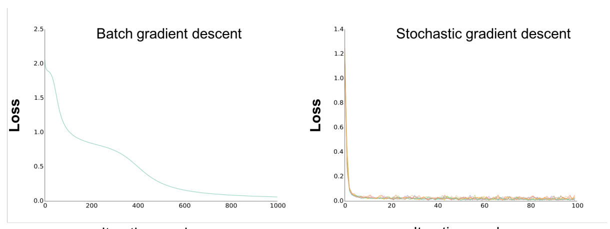
	2 Units			5 Units			10 Units			50 Units		
Error rate:	train	val.	test	train	val.	test	train	val.	test	train	val.	test
Lambda = 0	9.30 %	11.3 %	14.0 %	6.00 %	6.67 %	9.00 %	6.33 %	6.67 %	9.00 %	6.00 %	6.67 %	8.33 %
Lambda = 1e-6	9.30 %	11.3 %	14.0 %	6.00 %	6.67 %	9.00 %	6.33 %	6.67 %	9.00 %	6.00 %	6.67 %	8.33 %
Lambda = 1e-4	10.0 %	11.7 %	14.0 %	6.00 %	6.67 %	9.00 %	6.33 %	6.67 %	9.00 %	6.00 %	6.67 %	8.33 %
Lambda = 1e-2	30.0 %	33.3 %	29.7 %	6.33 %	6.67 %	9.00 %	6.00 %	6.67 %	9.00 %	19.0 %	23.0 %	25.3 %
Lambda = 1	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %	66.7 %

FIG. 4: Error rates for toy data2

The distribution of toy dataset2 is more complicated than toy datasets1. This implies that complex data requires more hidden units for better prediction.

2. batch and stochastic gradient

Figure 5 shows the how the average loss function evolves with iterations. It clearly shows that stochastic gradient descent algorithm is much faster than batch gradient descent in terms of iteration numbers. However, gradient descent has uncertainty when iteration number is large, and cannot give us a stable model. For this reason, we choose to use batch gradient method to evaluate our model performances.



Neural network with 5 hidden units, and step size of 1. Batch gradient descent spent more iteration to reach minimum, while stochastic gradient descent spent way less time.

FIG. 5: Gradient VS Batch gradient descent for toy model 1

3. regularization

As we can see from figure 4, λ doesn't have effects on error rate when it is small (about $< 10^{-2}$), it has negative value to the model when it is 1. Also, our models are very good ($< 1\%$ test error) even when $\lambda = 0$.

F. MNIST data

1. Stochastic VS Batch

Figure 6 shows the training process for MNIST dataset with stochastic gradient descent. Figure 7 compares stochastic and batch gradient descent for MNIST data. It's obvious that stochastic algorithm is much faster, but is unstable. We use batch algorithm for the rest of our analysis.

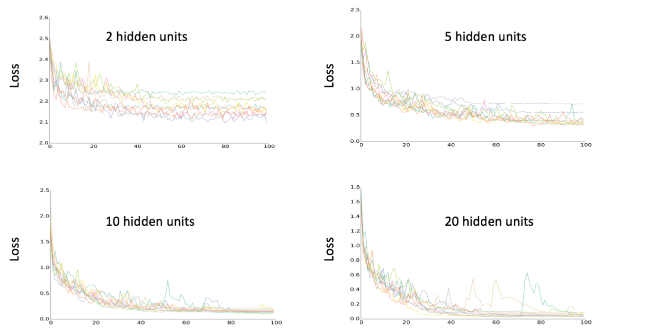


FIG. 6: Stochastic gradient descent for different hidden units, MNIST dataset

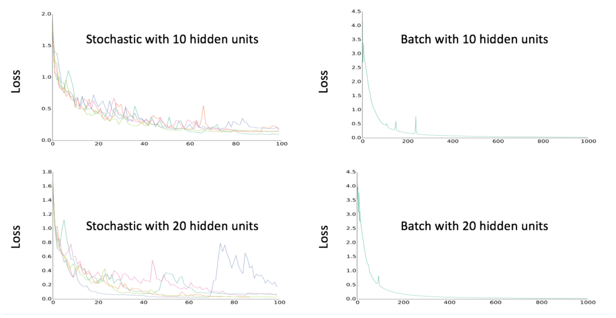
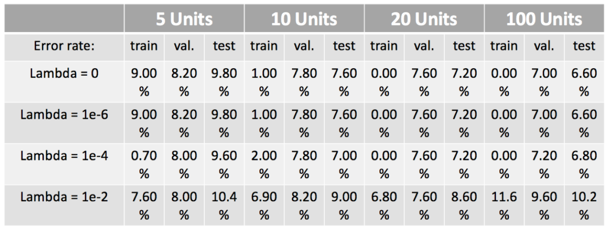


FIG. 7: Training processes for gradient and stochastic gradient descent, MNIST dataset

2. Hidden units

Figure 8 is a comprehensive list of how hidden units and regularization parameter affect model performances. For MNIST datasets, we see that 5 hidden units is not sufficient to give a best model. The performance saturated at a hidden units of 20.



	5 Units			10 Units			20 Units			100 Units		
Error rate:	train	val.	test	train	val.	test	train	val.	test	train	val.	test
Lambda = 0	9.00 %	8.20 %	9.80 %	1.00 %	7.80 %	7.60 %	0.00 %	7.60 %	7.20 %	0.00 %	7.00 %	6.60 %
Lambda = 1e-6	9.00 %	8.20 %	9.80 %	1.00 %	7.80 %	7.60 %	0.00 %	7.60 %	7.20 %	0.00 %	7.00 %	6.60 %
Lambda = 1e-4	0.70 %	8.00 %	9.60 %	2.00 %	7.80 %	7.00 %	0.00 %	7.60 %	7.20 %	0.00 %	7.20 %	6.80 %
Lambda = 1e-2	7.60 %	8.00 %	10.4 %	6.90 %	8.20 %	9.00 %	6.80 %	7.60 %	8.60 %	11.6 %	9.60 %	10.2 %

FIG. 8: Error rates with different parameters, MNIST dataset

3. Regularization and Step size

In our analysis, we tried different λs of 10^{-6}, 10^{-4}, 10^{-2}, and 1. When λ is small (10^{-6}, or 10^{-2}), the error rates are almost the same as when $\lambda = 0$. However, when λ is large (10^{-2} or 1), the performance of the model get worse. In addition, our model have optimal performance even when $\lambda = 0$ (hidden units = 20). In summary, in our particular script, tuning λ doesn't have much value, so we stick to $\lambda = 0$ when reporting the best model.

Also, we tested different step sizes of 0.5, 1 and 2. The results are very similar. Since the error rate for testing data is already low (7.2% for the best parameters), we reason it's not valuable to test more different step sizes. We choose 1 as our step size for reporting the best model.

4. Model selection

The best model has 20 hidden units, lambda = 0, step size = 1, and with batch gradient descent algorithm. Train error = 0. Validate error = 7.6%. Test error = 7.2%.