MDS6232: Assignment 3

Name: WEI Qing (218012040)

Problem 1

$$h_t = tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$z_t = softmax(W_{hz}h_t + b_z)$$

$$L(x,y) = \sum_t L_t = \sum_t -logz_{t,y_t}$$

1.1 Calculate $\frac{\partial L}{\partial W_{hz}}$

$$u_t = W_{hz}h_t + b_z$$

$$z_t = softmax(u_t)$$

$$\frac{\partial z_{t,u}}{\partial u_{t,j}} = \delta(i=j)z_{t,j} - z_{t,i}z_{t,j}$$

$$\frac{\partial u_{t,l}}{\partial W_{hz,ij}} = \frac{\partial(\sum_s W_{hz,ls}h_{t,s} + b_{z,l})}{\partial W_{hz,ij}} = \delta(i=l)h_{t,j}$$

$$\frac{\partial u_{t,l}}{\partial h_{t,k}} = \frac{\partial(\sum_s W_{hz,ls}h_{t,s} + b_{z,l})}{\partial h_{t,k}} = W_{hz,lk}$$

$$\frac{\partial L}{\partial z_{t,i}} = \frac{-\delta(i=y_t)}{z_{t,i}}$$

$$\begin{aligned}
\frac{\partial L}{\partial W_{hz,ij}} &= \sum_t \frac{\partial L}{\partial z_t}\frac{\partial z_t}{\partial W_{hz,ij}} \\
&= \sum_{t,k} \frac{\partial L}{\partial z_{t,k}}\frac{\partial z_{t,k}}{\partial W_{hz,ij}} \\
&= \sum_{t,k,l} \frac{\partial L}{\partial z_{t,k}}\frac{\partial z_{t,k}}{\partial u_{t,l}}\frac{\partial u_{t,l}}{\partial W_{hz,ij}} \\
&= \sum_{t,k} \frac{\partial L}{\partial z_{t,k}}\frac{\partial z_{t,k}}{\partial u_{t,i}}h_{t,j} \\
&= \sum_t -\frac{h_{t,j}}{z_{t,y_t}}\frac{\partial z_{t,y_t}}{\partial u_{t,i}}
\end{aligned}$$

1.2 Calculate $\frac{\partial L}{\partial W_{hh}}$

$$v_t = W_{xh}x_t + W_{hh}h_t + b_h$$

$$h_t = tanh(v_t)$$

$$\frac{\partial h_{t,k}}{\partial v_{t,i}} = \delta(k=i)(1-h_{t,k}^2)$$

$$\begin{aligned}\frac{\partial h_{t+1,i}}{\partial h_{t,k}} = \frac{\partial h_{t+1,i}}{\partial v_{t+1,i}}\frac{\partial v_{t+1,i}}{\partial h_{t,k}} =& \frac{\partial h_{t+1,i}}{\partial v_{t+1,i}}\frac{\partial((W_{xh}x_{t+1})_i + \sum_s W_{hh,is}h_{t,s} + b_{h,i})}{\partial h_{t,k}} \\ =& \delta(k=i)(1-h_{t+1,k}^2)W_{hh,ik}\end{aligned}$$

$$\frac{\partial v_{t,l}}{\partial W_{hh,ij}} = \frac{\partial((W_{xh}x_t)_l + \sum_s W_{hh,ls}h_{t-1,s} + b_{h,l})}{\partial W_{hh,ij}} = I(i=l)h_{t-1,j}$$

$$\begin{aligned}\frac{\partial L}{\partial h_{t,k}} =& \frac{\partial L}{\partial z_t}\frac{\partial z_t}{\partial h_{t,k}} + \frac{\partial L}{\partial h_{t+1}}\frac{\partial h_{t+1}}{\partial h_{t,k}} \\ =& \sum_s \frac{\partial L}{\partial z_{t,s}}\frac{\partial z_{t,s}}{\partial h_{t,k}} + \sum_i \frac{\partial L}{\partial h_{t+1,i}}\frac{\partial h_{t+1,i}}{\partial h_{t,k}} \\ =& \sum_{s,l} \frac{\partial L}{\partial z_{t,s}}\frac{\partial z_{t,s}}{\partial u_{t,l}}\frac{\partial u_{t,l}}{\partial h_{t,k}} + \sum_i \frac{\partial L}{\partial h_{t+1,i}}\frac{\partial h_{t+1,i}}{\partial h_{t,k}} \\ =& \sum_l -\frac{W_{hz,lk}}{z_{t,y_t}}\frac{\partial z_{t,y_t}}{\partial u_{t,l}} + \sum_i \frac{\partial L}{\partial h_{t+1,i}}\frac{\partial h_{t+1,i}}{\partial v_{t+1,i}}W_{hh,ik}\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial W_{hh,ij}} =& \sum_t \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial W_{hh,ij}} \\ =& \sum_{t,k} \frac{\partial L}{\partial h_{t,k}}\frac{\partial h_{t,k}}{\partial W_{hh,ij}} \\ =& \sum_{t,k,l} \frac{\partial L}{\partial h_{t,k}}\frac{\partial h_{t,k}}{\partial v_{t,l}}\frac{\partial v_{t,l}}{\partial W_{hh,ij}} \\ =& \sum_{t,k} \frac{\partial L}{\partial h_{t,k}}\frac{\partial h_{t,k}}{\partial v_{t,i}}h_{t-1,j} \\ =& \sum_{t,k} \frac{\partial L}{\partial h_{t,k}}\delta(k=i)(1-h_{t,k}^2)h_{t-1,j}\end{aligned}$$

Problem 2

$$h_t = F_\theta(h_{t-1}, x_t)$$

$$h_t = G_t(x_t, ..., x_1)$$

2.1

$$\begin{aligned}h_t =& F_\theta(h_{t-1}, x_t)\\ =& F_\theta(F_\theta(h_{t-2}, x_{t-1}), x_t)\\ =& F_\theta(F_\theta(F_\theta(h_{t-3}, x_{t-2}), x_{t-1}), x_t)\\ & ...\end{aligned}$$

keep on plugging h into the equation until x_1 is contained in the above formula.

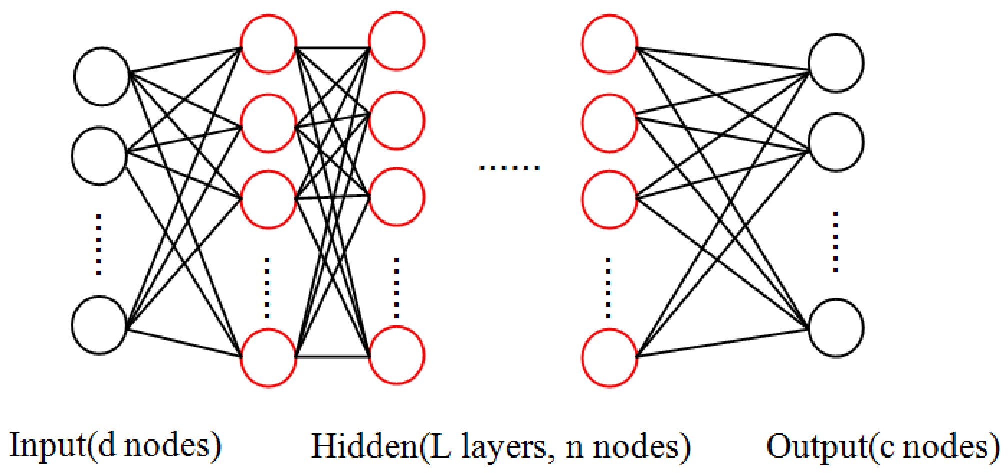
2.2

(1) As for the variation in sequence length and the generalization of the model, the RNN model with the feature of parameter sharing that could solve this problem.

(2) As for the model complexity of G_t, as we can see from the formula in part 2.1, the complexity will grow as the number of historical x increases. The RNN model has the function of containing all the historical infomation in each hidden layer, which effectively avoids the increasing of model complexity.

Problem 3

The neural network mentioned in this part can be visualized as follow:



3.1

According to the figure showed above, we know that the number of parameters can be divided into three parts:

(1) input to hidden: d*n

(2) hidden to hidden: n*n*(L-1)

(3) hidden to output: n*c

Thus, the space complexity (total number of parameters to store) of this neural network is $dn + n^2(L-1) + nc$

3.2

$$\begin{aligned}\frac{\partial u}{\partial W_{ij}} &= \sum_k \frac{\partial u}{\partial y_k}\frac{\partial y_k}{\partial W_{ij}} \\ &= \sum_k \frac{\partial u}{\partial y_k} f'((Wx)_k)\frac{\partial \sum_l W_{kl}x_l}{\partial W_{ij}} \\ &= \frac{\partial u}{\partial y_i} f'((Wx)_i)x_j\end{aligned}$$

Similarly we have:

$$\begin{aligned}\frac{\partial z}{\partial x_i} &= \sum_k \frac{\partial z}{\partial y_k}\frac{\partial y_k}{\partial x_i} \\ &= \sum_k \frac{\partial z}{\partial y_k} f^{'}((Wx)_k)\frac{\partial \sum_l W_{kl}x_l}{\partial x_i} \\ &= \frac{\partial z}{\partial y_i} f^{'}((Wx)_i)W_{ki}\end{aligned}$$

According to the dimensions of variables, we need $2mn$ multiplications for computing $\frac{\partial z}{\partial W_{ij}}$ and $2m$ multiplications for $\frac{\partial z}{\partial x_i}$, respectively.

For the number of multiplications needed for BP:

(1) output to hidden: $m = c$, update weights needs $2nc$, update neurons needs $2n$;

(2) hidden to hidden: $m = n$, update weights needs $2n^2(L-1)$, update neurons needs $2n(L-1)$;

(3) hidden to input: $m = d$, update weights needs $2nd$.

Thus, in total, we need $2n[1 + d + c + (L-1)(n+1)]$ multiplications.

3.3

$$\frac{\partial u_N}{\partial W_{ji}} = \frac{\partial u_N}{\partial u_i}\frac{\partial u_i}{\partial net_i}\frac{\partial net_i}{\partial W_{ji}}$$

Here, $\frac{\partial u_N}{\partial u_i}$ should be considered.

(1) output to hidden: $m = c$, update weights needs $2nc$, update neurons needs nc;

(2) hidden to hidden: $m = n$, update weights needs $2n^2$, update neurons needs $(L-i+1)cn^{L-i}$;

(3) hidden to input: $m = d$, update weights needs $2nd$.

Thus, in total, we need $2nc + nc + 2n^2 + 2nd + \sum_{i=1}^{L-1}(L-i+1)cn^{L-i}$ Which can be simplified as: $n(3c + 2d + 2n) + c\sum_{i=1}^{L-1}(i+1)n^i$

Problem 4

Here, we have an autoencoder where one linear hidden layer and the mean squared error criterion is used to train the network.

Suppose that in the autoencoder: we have input $x_n \in \mathbb{R}^m$, output $\tilde{x}_n \in \mathbb{R}^m$.

Since the layer is linear, we can use matrix to represent the process of the encoder and decoder, so $\tilde{x}_n = U_d U_e x_n$, where U_e and U_d represent the encode and decode operation, respectively.

Loss function:

$$J = \frac{1}{N} \sum_{n=1}^{N} \|\tilde{x}_n - x_n\|$$

For the procedure of PCA:

(1) PCA projects points to the subspace orthogonally to maximizes the variance of them, so it should minimize the sum-of-squared error.

This part is consistant with the loss function of the autoencoder.

(2) PCA will create a k dimensions subspace generated by k largest eigenvalues of the original input. Now there is no nonlinearty in the encoder and decoder, thus we should have $U_d = U_e^{-1}$ so that we can recover x_n, then U_e can be represented as $U_e = AU$, where U is a unitary matrix and A is a non-singular arbitrary $k \times k$ transform matrix, notice that encoder will compress the input so $k < m$. Then we get:

$$\tilde{x}_n = U_d U_e x_n = (U^T A^{-1})(AU)x_n$$

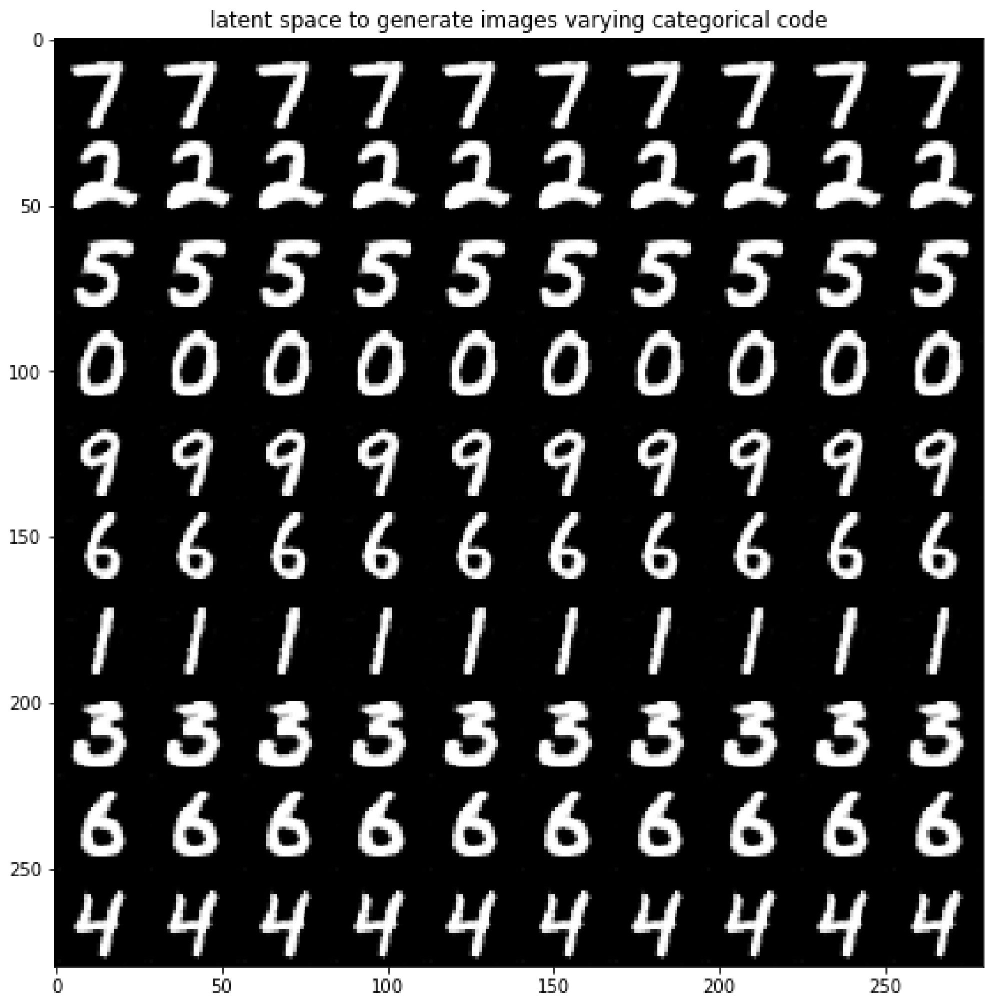
It shows that the encoder is the k dims PCA projection of x_n with an arbitrary transformation A, and the decoder projects the subspace back to the original space. So the space of these k hidden units is the same span of the first k of the principal components.

Then it is proved that this linear autoencoder is equivalent to PCA.

Problem 5

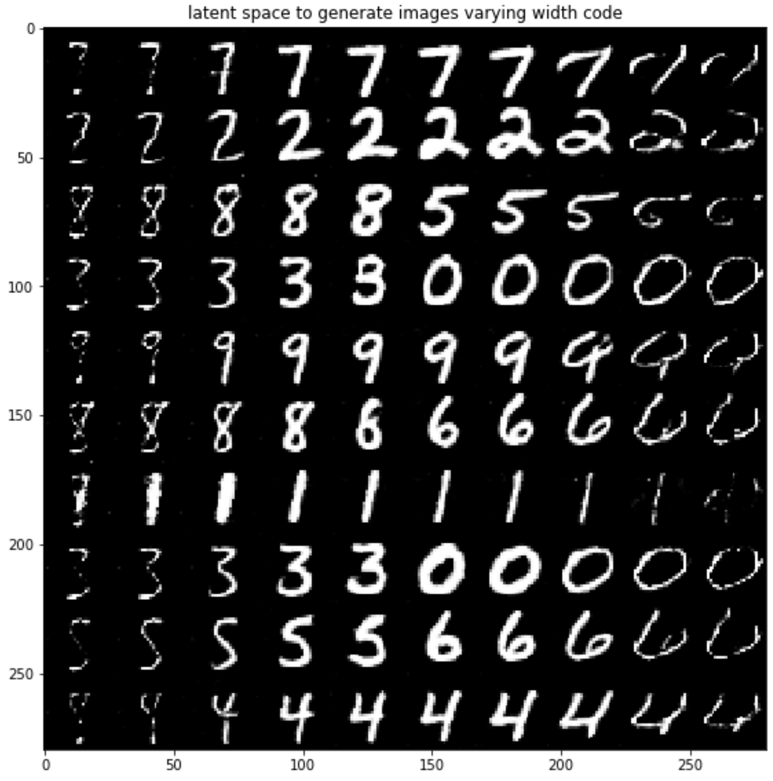
Part of the running results of the infoGAN using the framework Tensorflow are listed as below:

(1) Use fixed z and c and varying categorical code to generate diffferent digits.

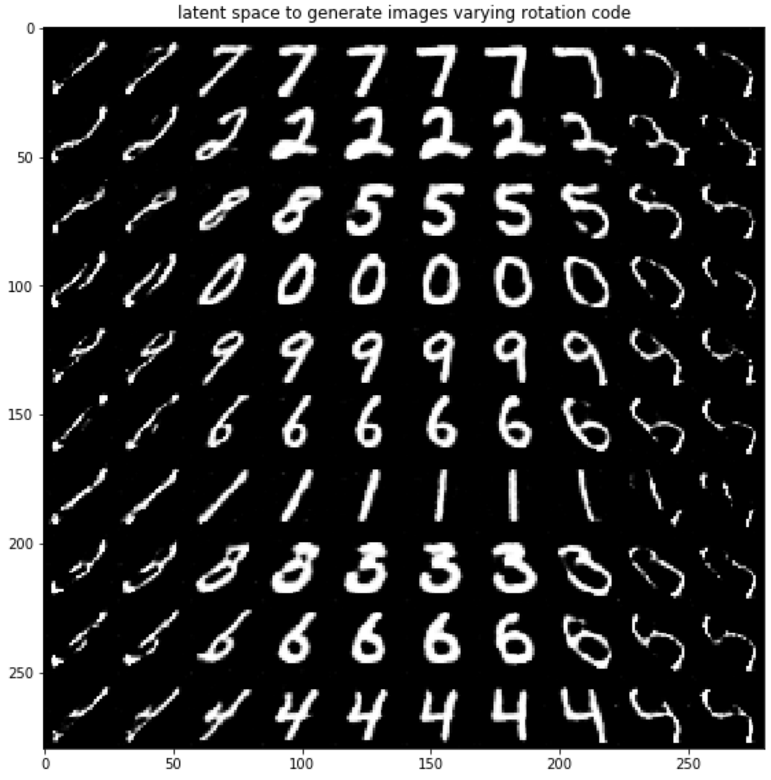


(2) Use fixed z and d and varying continous code to generate images with varying rotation angles and width. c1

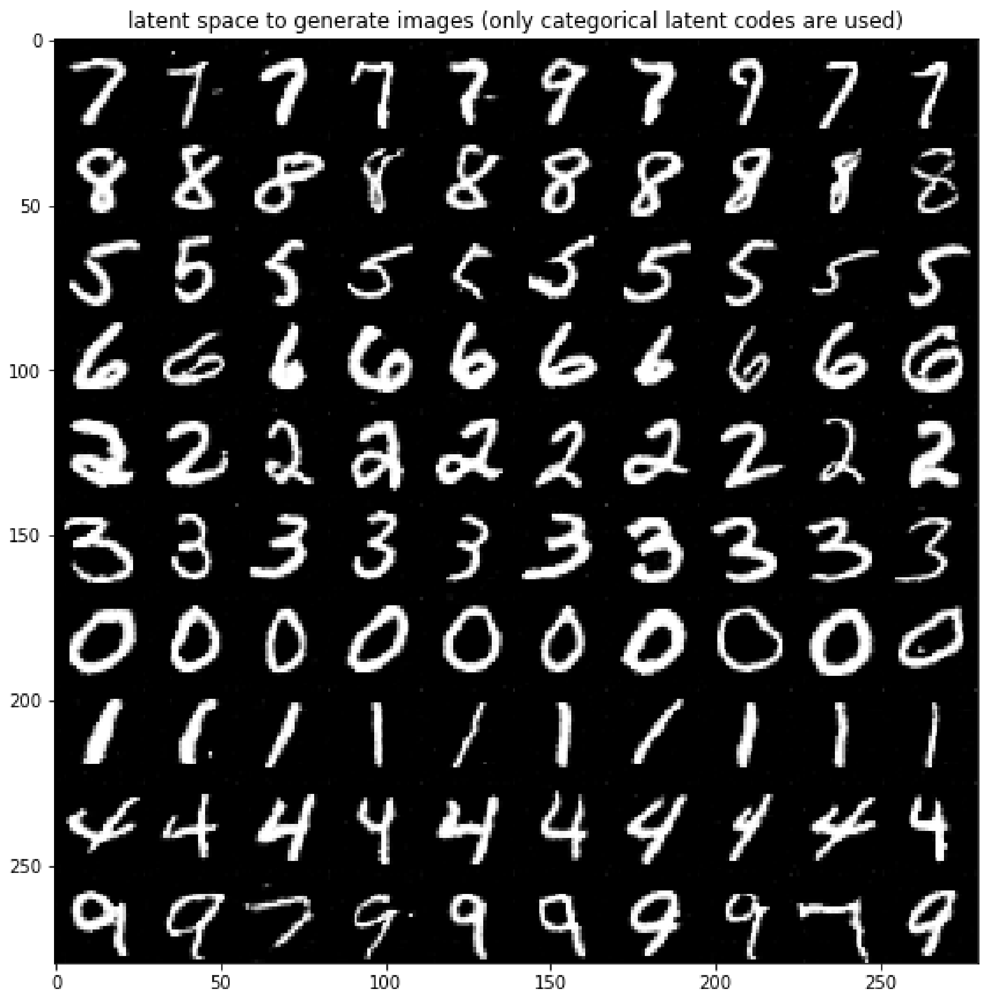
latent space to generate images varying width code



latent space to generate images varying rotation code



(3) Removed the continous code and retrain the Model, then vary the categorical codes and z codes



Conclusion: The model can generate different digits well, use the model with only categorical codes to classify the images, it can reach the accuracy of 92%