CS446: Machine Learning	**Spring 2018**
Machine Problem 10	
Handed Out: Jan. 16, 2018	*Due: Apr. 12, 2018 (11:59 AM Central Time)*

Note: The assignment will be autograded. It is important that you do not use additional libraries, or change the provided functions input and output.

Part 1: Setup

- Connect to a EWS machine.

```
ssh (netid)@remlnx.ews.illinois.edu
```

- Load python module, this will also load pip and virtualenv. We use python 3.4 in this assignment.

```
module load python/3.4.3
```

- Reuse the virtual environment from the previous mps.

```
source ~/cs446sp_2018/bin/activate
```

- Go to your svn directory. Copy mp10 into your svn directory, and change directory to mp10.

```
cd ~/(netid)
svn cp https://subversion.ews.illinois.edu/svn/sp18-cs446/_shared/mp10 .
cd mp10
```

- Install the requirements through pip.

```
pip install -r requirements.txt
```

- Create the data directory and prevent svn from checking in the data directory. Remember not to submit any training data to your svn directory.

```
mkdir MNIST_data
svn propset svn:ignore MNIST_data .
```

Part 2: Exercise

In this exercise, we will how learn to generate images from the data distribution using VAE. Review the lecture on Variational AutoEncoder for formulation. You are asked to implement a VAE and a conditional VAE.

Part 2.1 VAE

In this part, you will implement a VAE to learn the data distribution of the MNIST dataset. Use 2-dimensional latent space for the ease of visualization. Use fully connected layers for both the encoder and decoder in the VAE. More details are provided in the function docstring.
In `main_vae.py`, the training procedure, data reader and visualization are provided for you. Fill in the functions in the file `variational_autoencoder.py`. To test your program, run `main_vae.py` and see the generated images stored in 'latent_space_vae.png'.

The visualized results will be something similar to the figure below.

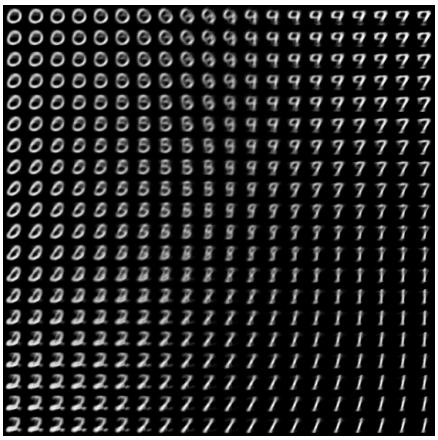

Figure 1: Generated images by VAE

Part 2.2 Conditional VAE (cVAE)

In this part, you will implement a conditional VAE to learn the data distribution of the MNIST dataset by taking the labels of the dataset into account. The encoder and decoder are conditional on the label, y. The decoder in cVAE learns $p_\theta(x|z,y)$ as opposed to $p_\theta(x|z)$ in VAE and the encoder in cVAE learns $q_\phi(z|x,y)$ as opposed to $q_\phi(z|x)$. You can derive the loss function for cVAE using what we taught in the class. The derivation of the formulations for cVAE is very similar as for VAE.

For implementation, you will use the labels as part of the input of the encoder and the decoder. More specifically, simply concatenate the input data with its label as the input of the encoder and concatenate the latent representation with the corresponding label as the input of the decoder. By concatenating the input data (n-dim) with its label (m-dim), we

create a $(n+m)$-dim vector whos first n elements are set to the input data and the rest m elements are set to the label. Similar for concatenating the latent representation with the label. In our case, the label y for each data point is a 10-dim vector specifying which class the data point belongs to. For example, if it is an image of the digit 6, the 7-th element in y is 1 and others are 0. Use the same architecture in part 2.1 for your cVAE. More details are provided in the function docstring.

In `main_cvae.py`, the training procedure, data reader and visualization are provided for you. Fill in the functions in the file `conditional_variational_autoencoder.py`. To test your program, run `main_cvae.py`.

The visualized results conditioned on 6 will be something similar to Figure 2.

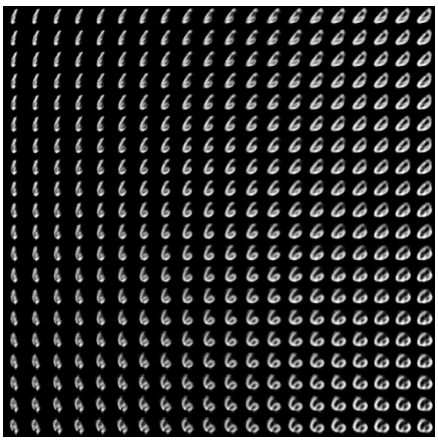

Figure 2: Generated images by cVAE

Part 3: Writing Tests

In `test.py` we have provided basic test-cases. Make sure your code can pass the simple testing cases provided in test.py. Feel free to write more. To test the code, run

```
nose2
```

Part 4: Submit

Submit your python code to the svn repository. You don't have to submit the visualized images. Submitting the code is equivalent to committing the code. This can be done with the follow command:

```
svn commit -m "Some meaningful comment here."
```

Lastly, double check on your browser that you can see your code at

```
https://subversion.ews.illinois.edu/svn/sp18-cs446/(netid)/mp10/
```