Introduction to Data Mining: Assignment #3

Summer 2020

Due: Jun 8th, 23:59:59 CST (UTC +8).

1. Neural Networks

The codes of this section are in the *neural_networks* folder.

In this problem, we will implement the feedforward and backpropagation process of the neural networks. We will use *digital.mat* as our experiment data. Finish *fullyconnect_feedforward*, *fullyconnect_backprop*, *relu_feedforward*, *relu_backprop* and the testing part in *run.m/run.ipynb*. Then we can train three layer (data, hidden-relu, loss) neural networks and report test accuracy.

Supplementary Knowledges:

i) Instead of using MSE loss function, we adopt softmax loss function here. Recall that in Assignment #2, we used logistic regression to classify two classes. And softmax regression model is a model that extends logistic regression to classify more clases than two [1]. In this problem, we have 10 classes. The softmax loss part codes are done.

ii) We can use *gradient_check.m/gradient_check.py* to check the correctness your computation. If $\frac{d}{d\theta}J(\theta) = \lim_{\epsilon\to 0}\frac{J(\theta+\epsilon)-J(\theta)}{\epsilon}$ holds, then we are in the right way. [2]

iii) Weight decay and momentum are used to update weight paramters in *get_new_weight_inc.m/get_new_weight_inc.py*.

2. K-Nearest Neighbor

The codes of this section are in the *knn* folder.

In this problem, we will play with K-Nearest Neighbor (KNN) algorithm and try it on real-world data. Implement KNN algorithm (in *knn.m/knn.py*), then answer the following questions.

(a) In *knn_exp.m/knn_exp.ipynb*, try KNN with different K (you should at least experiment K = 1, 10 and 100) and plot the decision boundary.

You are encouraged to vectorize[3][4] your code, otherwise the experiment time might be extremely long. You may find the MATLAB build-in functions *pdist2, sort, max* and *hist* useful. Also, you can use the function *eudist2*[5] written by Prof. Deng Cai[6].

[1] `http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression`

[2] `http://ufldl.stanford.edu/wiki/index.php/Gradient_checking_and_advanced_optimization`

[3] `http://www.mathworks.cn/cn/help/matlab/matlab_prog/vectorization.html`

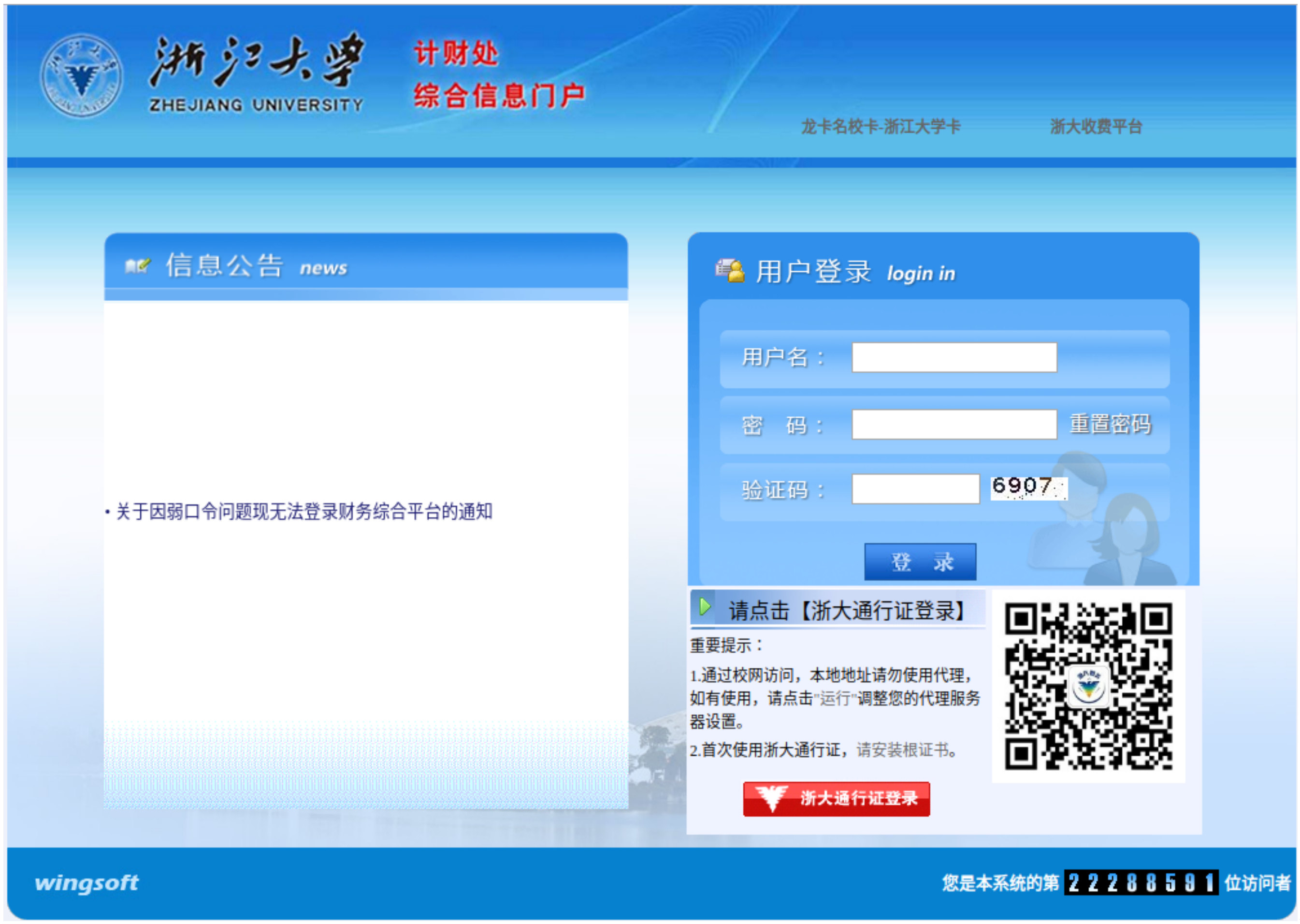
[4] `https://stackoverflow.com/questions/47755442/what-is-vectorization`

[5] `http://www.cad.zju.edu.cn/home/dengcai/Data/code/EuDist2.m`

[6] Prof. Deng Cai is an expert on MATLAB, you can find all his code at `http://www.cad.zju.edu.cn/home/dengcai/Data/data.html`. You can learn how to write fast MATLAB code by reading his code.

(b) We have seen the effects of different choices of K. How can you choose a proper K when dealing with real-world data ?

(c) Now let us use KNN algorithm to hack the CAPTCHA of a website[7] that we are all familiar with:



• 关于因弱口令问题现无法登录财务综合平台的通知

重要提示：

1.通过校网访问，本地地址请勿使用代理，如有使用，请点击"运行"调整您的代理服务器设置。

2.首次使用浙大通行证，请安装根证书。

您是本系统的第 2 2 2 8 8 5 9 1 位访问者

.

Finish *hack.m/hack.py* to recognize the CAPTCHA image using KNN algorithm.

You should label some training data yourself, and store the training data in *hack_data.mat/hack_data.npz*. Helper functions *extract_image* and *show_image* are give for your convenience.

Remember to submit *hack_data.mat/hack_data.npz* along with your code and report.

3. Decision Tree and ID3

Consider the scholarship evaluation problem: selecting scholarship recipients based on gender and GPA. Given the following training data:

Draw the decision tree that would be learned by ID3 algorithm and annotate each non-leaf node in the tree with the information gain attained by the respective split.

[7] http://cwcx.zju.edu.cn/WFManager/login.jsp

Gender	GPA	Scholarship	Count
F	Low	+	10
F	High	+	95
M	Low	+	5
M	High	+	90
F	Low	-	80
F	High	-	20
M	Low	-	120
M	High	-	30

4. K-Means Clustering

The codes of this section are in the *kmeans* folder.

Finally, we will run our first unsupervised algorithm – k-means clustering. Implement k-means algorithm (in *kmeans.m/kmeans.py*), then answer the following questions.

Note that there are different kind of methods to setup initial cluster centers for k-means algorithm, we will use a simple one – randomly choose K samples from dataset as initial cluster centers.

(a) Run your k-means algorithm on *kmeans_data.mat* with the number of clusters K set to 2. Repeat the experiment 1000 times. Use *kmeans_plot.m/kmeans_plot.py* to visualize the process of k-means algorithm for the two trials with largest and smallest SD (sum of distances from each point to its respective centroid).

(b) You should observe the issue that the outcome of k-means algorithm is very sensitive to cluster centroids initialization form the above experiment. How can we get a stable result using k-means?

(c) Run your k-means algorithm on the digit dataset *digit_data.mat* with the number of clusters K set to 10, 20 and 50. Visualize the centroids using *show_digit.m/show_digit.py*. You should be able to observe that k-means algorithm can discover the patterns in dataset without any label information.

(d) Another important application of k-means is Vector quantization[8]. Vector quantization is a classical quantization technique from signal processing. It works by dividing a large set of points (vectors) into groups, then representing the data points by their group centroid points, as in k-means and some other clustering algorithms.

Here we will use vector quantization to do image compression. By clustering image pixel value into K groups, we can represent each pixel with $\log(K)$ bits, instead of 24 bits (RGB, each channel has 8bit depth).

Finish *vq.m/vq.ipynb*. Compress images with K set to 8, 16, 32 and 64. I have provided you some sample images, however use your own photos is encouraged.

What is the compress ratio if we set K to 64 (Optionally, you can compute the compress ratio using Huffman encoding) ?

[8] `https://en.wikipedia.org/wiki/Vector_quantization`

Please submit your homework report to at `http://courses.zju.edu.cn:8060/course/11827/` in pdf format, with all your code in a zip archive.