Machine Learning: Assignment
Summer Holiday 2019

Due: There is no hard deadline. Just finish **all 4 assignments** as quickly as you can.

1. Bayes Decision Rule

(a) Suppose you are given a chance to win bonus grade points:

> There are three boxes whose appearance are all the same. Only one box contains a special prize that will grant you 1 bonus points. You pick a box, say B_1, and the teacher, who knows what is in the boxes, opens another box which does not contains the prize, say B_2.
>
> Now you are given a second chance to choose the left box, say B_3. You can either stick to B_1 or switch your choice. What is your best choice?

(i) What is the prior probability of B_1 contains prize, $\mathrm{P}(B_1 = 1)$?

(ii) What is the likelihood probability of B_2 does not contains prize if B_1 contains prize, $\mathrm{P}(B_2 = 0|B_1 = 1)$?

(iii) What is the posterior probability of B_1 contains prize given B_2 does not contain prize, $\mathrm{P}(B_1 = 1|B_2 = 0)$?

(iv) According to the Bayes decision rule, should you change your choice or not?

(b) Now let us use bayes decision theorem to make a two-class classifier. Please refer the codes in the *bayes_decision_rule* folder and main skeleton code is *run.m/run.ipynb*. There are two classes stored in *data.mat*. Each class has both training samples and testing samples of 1-dimensional feature $\mathbf{x}$.

(i) Finish the calculation of likelihood of each feature given particular class(in *likelihood.m/likelihood.py*). And calculate the number of misclassified test samples(in *run.m/run.ipynb*) using maximum likelihood decision rule. Show the distribution of $\mathrm{P}(x|\omega_i)$, and report the test error.

(ii) Finish the calculation of posterior of each class given particular feature(in *posterior.m/posterior.py*). And calculate the number of misclassified test samples(in *run.m/run.ipynb*) using optimal bayes decision rule. Show the distribution of $\mathrm{P}(\omega_i|x)$, and report the test error.

(iii) There are two actions $\{\alpha_1, \alpha_2\}$ we can take, with their loss matrix below. Show the minimal total risk ($R = \sum_x \min_i R(\alpha_i|x)$) we can get.

$\lambda(\alpha_i\|\omega_j)$	$j = 1$	$j = 2$
$i = 1$	0	1
$i = 2$	2	0

2. Text Classification with Naive Bayes

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

In this problem, you will implement a text classifier using Naive Bayes method, i.e., a classifier that takes an incoming email message and classifies it as positive (spam) or negative (not-spam/ham). The data are in *hw1_data.zip*. Since MATLAB is not good at text processing and lacks of some useful data structure, we have written some Python scripts to tranform email texts to numbers that MATLAB can read from. The skeleton code is *run.m/run.ipynb*(in *text_classification* folder).

In this assignment, instead of following the existing code, you can use any programming language you like to build up a text classifier barely from email texts. You are more encouraged to finish the assignment in this way, since you will get better understanding of where the features come from, what is the relationship between label, emails and words, and other details.

Here are some tips you may find useful:

i) **Relationship between words, document and label**. Theoretically, $P(word_i = N|\text{SPAM}) = P(word_i = N|\text{document-type}_j)P(\text{document-type}_j|\text{SPAM})$ should hold, where document-type$_j$ is the type of the document e.g. a family email will have more words about family members and house, a work email will have more words about bussiness and a game advertising email will have words like "play now". But we can not include all the document types (a not big enough data set) and that is not what naive bayes cares. For simplification, in traning we discard the documents information and mix all the words to generate $P(word_i|\text{SPAM})$ and $P(word_i|\text{HAM})$ denoting the possibility for a word in SPAM/HAM email to be $word_i$. Therefore $P(word_i = N|\text{SPAM}) = P(word_i|\text{SPAM})^N$.

ii) **Training**. Remember to add Laplace smoothing.

iii) **Testing**. When you compute $p(\mathbf{x}|y) = \prod_i p(x_i|y)$, you may experience floating underflow problem. You can use logarithm to avoid this issue..

(a) It is usually useful to visualize you learnt model to gain more insight. List the top 10 words that are most indicative of the SPAM class by finding the words with highest ratio $\frac{P(word_i|\text{SPAM})}{P(word_i|\text{HAM})}$ in the vocabulary.

(b) What is the accuracy of your spam filter on the testing set?

(c) True or False: a model with 99% accuracy is always a good model. Why? (Hint: consider the situation of spam filter when the ratio of spam and ham email is 1:99).

(d) With following confusion matrix[1]:

[1]Positive and negative often substitutes as predictions in two labels problem. They are usually defined implicitly by common sence. A xxx-detector will use positive for the presence of xxx and negative for the absence of xxx, eg doping detection in the Olympics will mark atheles taking doping as posible and otherwise negative. In TP/FP/TN/FN terminology, T stands for 'True' which means predict is the same as label whihe F stans for 'False'. And P stands for 'Positive' and N stands for 'Negative'. `http://en.wikipedia.org/wiki/Precision_and_recall`

	Spam(label)	Ham(label)
Spam(predict)	TP	FP
Ham(predict)	FN	TN

compute the precision and recall of your learnt model, where precision = $\frac{TP}{TP+FP}$, recall = $\frac{TP}{TP+FN}$

(e) For a spam filter, which one do you think is more important, precision or recall? What about a classifier to identify drugs and bombs at airport? Justify your answer.

3. 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 [2] here. 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. [3]

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

iv) You can see Fig. 1 [4] for an illustration of multilayer nerual networks and its backpropagation. More details can be found in the online book *Neural Networks and Deep Learning* [5].

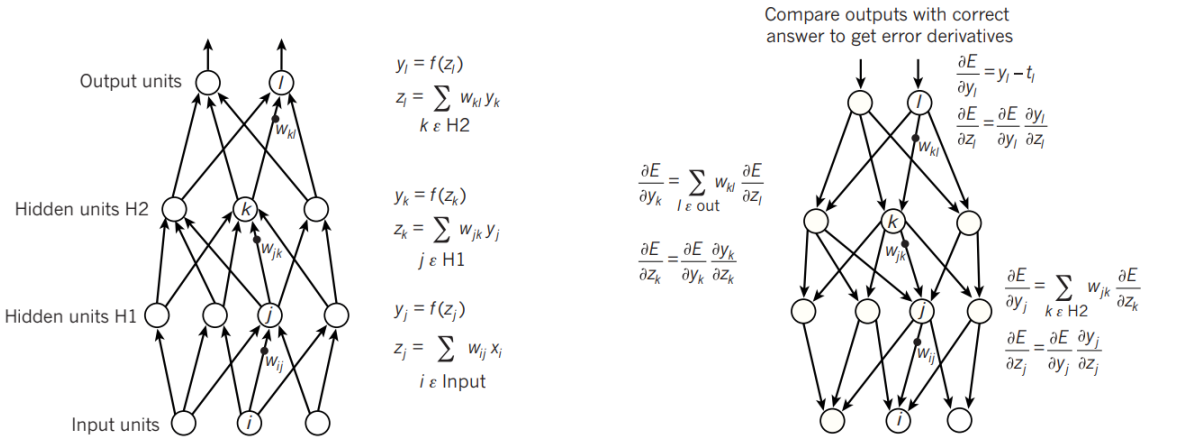

Figure 1: Multilayer neural networks and backpropagation.

[2] https://en.wikipedia.org/wiki/Softmax_function

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

[4] https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf

[5] http://neuralnetworksanddeeplearning.com/chap2.html

4. 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[6][7] 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*[8] written by Prof. Deng Cai[9].

(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 the Zhejiang University educational administration website [10]:

.

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.

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

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

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

[9] 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.

[10] `http://jwbinfosys.zju.edu.cn/default2.aspx`