CS 534 Machine Learning Homework 5

Si Chen

1. The code is in hw5-1point.ipynb

a) For radius= a, V_1= $\frac{S\times a^d}{d}$,

For radius = $a-\varepsilon$, V_2= $\frac{S\times(a-\varepsilon)^d}{d}$

The fraction of the volume which lies at values of the radius between $a-\varepsilon$ and a, where 0< $\varepsilon < a$ is

f = $\frac{V_1-V_2}{V_1}$ = $\frac{\frac{S\times a^d}{d} - \frac{S\times(a-\varepsilon)^d}{d}}{\frac{S\times a^d}{d}}$ =$1-(1-\frac{\varepsilon}{a})^{\mathrm{d}}$

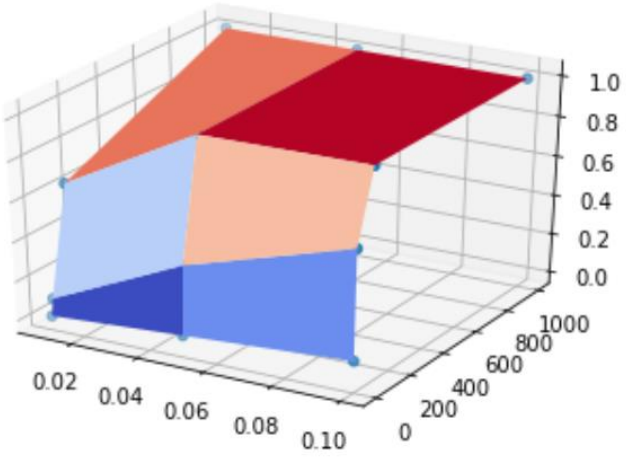
For 0<k<1, $\lim_{\mathrm{d}\to\infty} k^{\mathrm{d}}$=0

$0 < \varepsilon < a$,$0 < 1-\frac{\varepsilon}{a} < 1$, so, $\lim_{\mathrm{d}\to\infty}(1-\frac{\varepsilon}{a})^{\mathrm{d}}$ =0

$\lim_{\mathrm{d}\to\infty}$ f =$\lim_{\mathrm{d}\to\infty}(1-(1-\frac{\varepsilon}{a})^{\mathrm{d}}) = 1-\lim_{\mathrm{d}\to\infty}\left(1-\frac{\varepsilon}{a}\right)^{\mathrm{d}} = 1$

b) When $\frac{\varepsilon}{a}$ =0.01,0.05,0.1, and d=1,10,100,1000

F is shown in the figure below.



c) We can draw from the plot that

I. When $\frac{\varepsilon}{a}$ does not change, f increases while d increases, when d->∞, f->1

II. When d does not change, f increases while $\frac{\varepsilon}{a}$ increases.

2. The code is in hw5-2pca nmf.ipynb

a) I remove the missing data and normalize the data.

b) **5** components were needed to capture 95% of the variance in the data.
From the first 3 principal components, we can figure out which original features are important. I set a variance threshold of 0.35 and find all the features of which variances are above 0.35

These original features are “apps received”,”num FT undergrad”,”out-of-state tuition”,”est personal costs”

c) If the data are not normalized, since the original features are on different scale, the mapping will fit the features with the biggest values, thus the singular values are very high, leading to information lost. Therefore, in order to avoid capturing only the features with the biggest values, we need to normalize the data before performing PCA.

d) NMF squared error is 8.3490597287, PCA squared error is 2.79289880217. The squared error of PCA is less than that of NMF.
I set a variance threshold of 0.8 and find all the features of which variances are above 0.8 These original features are “apps received”, ”apps accepted”, ”num FT undergrad”, ”in-state tuition”, ”out-of-state tuition”.

e) The squared error of PCA is less than that of NMF, which means when using the same number of components, PCA contains more accurate information than NMF. Besides, the most important features of PCA and NMF are almost the same.

3. The code is in hw5-3 simulated neural network.ipynb

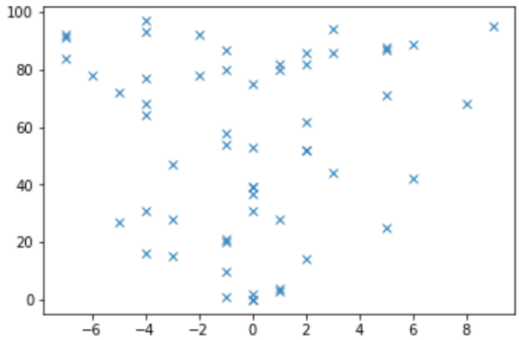
a) This depends on the training data set. If the samples are distributed relatively near the function $y=x^2$, we need less samples. Generally, the more the better. Of course, to some extent, the performance will converge and reach the best without change when more samples are added. In my case, I use 1000 random samples.

b) I choose the alpha as 0.00774 and hidden_layer_sizes as(8,5,6). Other hyperparameters are default values.

MLPClassifier(activation='relu', alpha=0.0077426368268112694, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(8,5,6), learning_rate='constant', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)

Firstly I use cross validation to choose the alpha and the first hidden layer size. With these two values, I use cross validation to choose the best second and third hidden layers’ sizes.

c) I generate 2000 test samples and the accuracy is 0.9985 The performance is quite good.



The figure above show the predicted positive values, which indicate $y>=x^2$.

4. The code is in hw5-4 Thyroid NN.ipynb
    a) At first I partition the data into 85%-15% train-test split. Since the samples are unbalanced, I preprocess the data by standardizing. Let the mean of these 6 features: 'age','TSH','T3','TT4','T4U','FTI' be zero, and the variance be unit. This is in the consideration of computing.
    b) Since neural network has so many hyperparameters to choose, I only choose the alpha as 0.27825594022071259 and hidden_layer_sizes as(4,7,4,4). Other hyperparameters are default values.

    MLPClassifier(activation='relu', alpha=0.27825594022071259, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(4,7,4,4) learning_rate='constant', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)

    Firstly I used cross validation to choose the alpha and the first hidden layer size. With these two values, I used cross validation to choose the best second hidden layer size. I use F2 score as the assessment strategy. Then, I tried to test the deeper layer size using only a few values. I found that 4 layers were the best.
    c) I use the svm and random forest hyperparameters from previous homework4. The first table shows the difference of performance metrics between Neural Networks, SVM and random forest. Neural Networks show good performance on train set, and not bad on test set, just like random forest. SVM is not as good as the other two. Maybe it is because random forest is not likely to overfit and NN can implement deep learning.

	evaluation	NN train	NN test	SVM polynomial train	SVM polynomial test	random forest test
0	misclassification	0.001681	0.014286	0.003361	0.019048	0.009524
1	F1 score	0.962264	0.700000	0.923077	0.600000	0.750000
2	F2 score	0.962264	0.744681	0.912548	0.638298	0.697674

The second table shows the number of parameters and computation time. NN has the most parameters and longest computation time.

	evaluation2	NN	SVM polynomial	random forest
0	number of parameters	21	2	4
1	computation time	0.367445	0.153196	0.195137