THU-70250043, Pattern Recognition (Spring 2016) **Homework: 5**

Sparse Learning, SVM and KNN

Lecturer: Changshui Zhang `zcs@mail.tsinghua.edu.cn`

Student: Qingfu Wen `wqf15@mails.tsinghua.edu.cn`

Sparse Learning

1. Please briefly describe the geometry reason of sparsity using ℓ_1 regularized optimization from unit circle diagram in Fig.8.1. And describe the possible sparsity property of outcome using ℓ_2 and $\ell_{0.5}$(No proof is needed).

   SOLUTION:

   From the Fig.8.1, we can see that ℓ_1 norm has some 'corner points' which is on the axis. These points are more likely to be the solution of our problem which are sparse(their coordinate have many 0 since they are one the axis). ℓ_2 norm does not hold sparsity because all the point on the ℓ_2's boundary have the same possibility to be a solution. Thus, $\ell_{0.5}$ have sparsity property.

2. The famous RIP(restricted isometry property) condition is commonly used in sparse recovery theory which demonstrate following property of sampling matrix $A \in R^{p\times m}$: given S a subset of all columns of A and $s = |S|$ an integer, there exists $\delta_S \in (0,1)$ that for any submatrix $A_S \in R^{p\times s}$ of A and for every y,

$$(1-\delta_S)||y||_2^2 \leq ||A_S y||_2^2 \leq (1+\delta_S)||y||_2^2$$

   If δ_S is small enough, there is overwhelming probability that through sparse learning, we can exactly recovery the original signal. If A is a gaussian random matrix, i.e. each element in the matrix is a random variable from $N(0, 1/p)$, there exist following theorem: set $r = s/m$, and further

$$f(r) = \sqrt{m/p} \cdot (\sqrt{r} + \sqrt{2(H(r))}), H(r) = -r\log r - (1-r)\log(1-r)$$

   , for each $\epsilon > 0$, the RIP constant δ_S for A satisfy:

$$P(1+\delta_S > [1+(1+\epsilon)f(r)]^2) \leq 2 \cdot \exp(-mH(r)\cdot \epsilon/2)$$

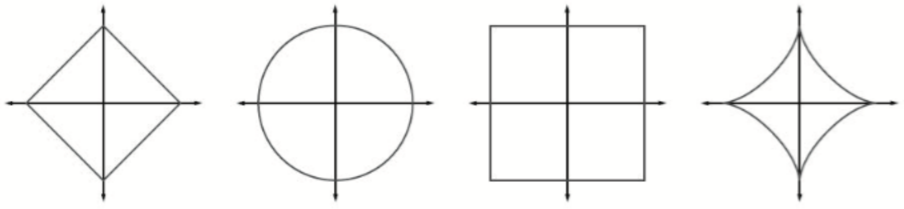
Figure 8.1: Unit circle of different norms, from left to right: ℓ_1, ℓ_2, ℓ_∞, and $\ell_{0.5}$

Please explain why gaussian random matrix can be chosen as the sampling matrix in sparse learning, i.e., why with a large probability, gaussian random matrix have small RIP constant δ_S.

3. You can use MATLAB code from online sources, for example ℓ_1 Benchmark Package(`http://www.eecs.berkeley.edu/~yang/software/l1benchmark/l1benchmark.zip`). Here we recommend using FISTA algorithm, i.e. SolveFISTA.m. Please finish programming to figure out the effectiveness of ℓ_1 optimization in solving underdetermined systems. You can test the probability of sucessful recovery using simulated data solving $Ax = b$ with different scale of A.

SOLUTION:

suppose A is a $d * n$ matrix and x is a $n * 1$ vector with k non-zero elements, then do experiments varying d, n, k.

d	n	k	error
8	10	1	$9.59 * 10^{-7}$
80	100	10	$2.13 * 10^{-7}$
800	1000	100	$2.44 * 10^{-7}$
1600	2000	200	$2.04 * 10^{-7}$
2400	3000	300	1.98
8000	10000	1000	3.56

Table 8.1: different scale of A

SVM

In this problem, we will use SVM to test on the news20 dataset. We randomly selected 1000 training samples and 1000 testing samples from the whole dataset (For the details of this dataset, refer to `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#news20.binary`). Load the given ".mat" file, **Xtrn** and **ytrn** are the training data, **Xtst** and **ytst** are the testing data. (*Hint: You can write an SVM yourself, or use the off-the-shelf svm tools such as the libsvm:* `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools`.)

1. Train and test SVM with non-linear kernels. Store the parameters that perform best.
2. Train and test linear SVM, compare its performance with the optimal non-linear kernel SVM we get in last step.
3. You can refer to these papers for the details of the SVM tools:
   a. C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011.
   b. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008), 1871-1874.

SOLUTION:

I use `SVMcgForClass.m` to find parameters γ and c with best performance of RBF kernel($c =$

kernel	accuracy
polynomial	81.1%
radial basis function	88.5%
sigmoid	76.8%
linear	88.1%

Table 8.2: different kernel of SVM

$8, \gamma = 0.25$).

KNN

Realize KNN classifier yourself, test on mnist dataset and show the experiment results:

1. Use training datasets with different scales, compare the performances of resulted KNNs, including accuracy, time and space complexity.
2. Try KNNs with different k numbers, and compare the performance.
3. Try different distance metrics, and compare the performance. we set $k = 3$ and training size= 10000.

training size	accuracy	time	memory
60000	0.994	582.51 s	0.39 GB
30000	0.992	321.50 s	0.21 GB
10000	0.992	93.79 s	0.07 GB
5000	0.986	48.48 s	0.04 GB
2500	0.983	26.34 s	0.02 GB
1000	0.977	10.83 s	0.01 GB
500	0.972	6.09 s	0.01 GB

Table 8.3: different training size of kNN

k	trainSize=1000	trainSize=10000
1	0.977	0.988
2	0.986	0.993
3	0.979	0.989
4	0.985	0.994
5	0.976	0.988
6	0.961	0.99
7	0.97	0.987
8	0.981	0.989
9	0.98	0.988
10	0.975	0.987

Table 8.4: different k of kNN

distance metrics	accuracy
euclidean	0.989
cityblock	0.988
minkowski	0.987
cosine	0.995
correlation	0.993
jaccard	0.979
hamming	0.91

Table 8.5: different distance metrics of kNN