Sparse Norm-Regularized Reconstructive Coefficients Learning

Bin Liu,Shuo Chen,Mingjie Qian,Changshui Zhang
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology(TNList)
Department of Automation,Tsinghua University,Beijing, P.R.China
{liubin07,chenshuo07}@mails.tsinghua.edu.cn,qian.mingjie@gmail.com,zcs@mail.tsinghua.edu.cn

***Abstract*—Inspired by the fact that the final decision rule is mainly affected by a small subset of the training samples, i.e., Support Vector Machine(SVM) shows that the decision function relies on the few samples that are on or over the margin. We propose a new framework that explicitly strengthen this intuitive fact by adding an l_1-norm regularizer. We give different formulations for our framework in different scenarios, and the experiments show that our framework can not only lead to high sparse solutions but also better performance than traditional methods.**

***Keywords*-support vector machine;l_1-norm;sparse;**

I. Introduction

Finding the sparsity of the intrinsic data structure has attracted a lot of interest in machine learning, statistics and signal processing. It is quite common that we get an abundance of data for a particular learning task. For example, the massive text data from the web pages and the easily captured image by our digital cameras. However the valuable information may not increase as the number of the training samples does. For example, *Support Vector Machine*(*SVM*) ([1]) in the supervised learning scenario has shown that there is only a small number of samples that affect the final decision boundary. In other words, the indices for the samples as support vectors are very sparse.

One method aiming at sparse solution is by utilizing l_1-norm regularization. Many works have shown that by adding an l_1-norm regularizer, one can not only find the intrinsic structure of the data sets, but also improve the performance of the learning machine. For example, in linear regression scenario the famous *Lasso* regularizer was proposed to do the model selection ([2]). The l_1-norm regularizer was also used to do feature selection in *SVM* ([3], [4], [5]). [6] used the characteristics of sparsity in image processing.

In this paper we propose a novel framework for reconstructive coefficients learning with l_1-norm regularization term. Our work is based on the assumption that: the number of the training samples that mainly affect the final decision function is very small. Our goal is trying to find the samples that are close enough to the decision boundary no matter it is labeled or unlabeled. We show that our framework can be specialized to many existed supervised and semi-supervised sparse methods.

In section 2, we introduce some related works. In section 3, We propose our new framework firstly. Then we specialize our framework to fit the supervised and semi-supervised scenario. At last we extend our framework by using multi-kernel functions. We demonstrate our experiment results and analyzing in section 4, and conclusions and future work is in the last section.

Notations: In this paper we assume that the number of labeled samples is ℓ, and the number of the unlabeled samples is u. We use N to denote the number of the whole training samples. Specifically, in supervised learning scenario all the samples are labeled, so $N = \ell$. On the other hand, in semi-supervised scenario we not only have the ℓ labeled samples but also u unlabeled samples, so $N = \ell + u$. We use K to denote the kernel matrix constructed by the whole training dataset $X = [x_1, \cdots, x_\ell, x_{\ell+1}, \cdots, x_{\ell+u}]$. We would like to emphasize that the order of the dataset X is fixed. This means that when constructing the kernel matrix K, the first ℓ sample are the ℓ labeled samples and the rest u samples are unlabeled ones.

$$K = \begin{pmatrix} \mathcal{K}(x_1, x_1) & \cdots & \mathcal{K}(x_1, x_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(x_N, x_1) & \cdots & \mathcal{K}(x_N, x_N) \end{pmatrix}$$

$\mathcal{K}(x, y)$ is the kernel function $f : X \times X \to Y$, where X denotes the original space while Y denotes the high-dimensional space, i.e. the Radio Basis Function(*RBF*) kernel. Its form is as follows:

$$\mathcal{K}(x, y) = e^{\frac{-\|x-y\|_2^2}{\sigma^2}}$$

In this paper we use K_ℓ to denote the first ℓ rows of the kernel matrix K. We will introduce some existing methods which are related to our work in this section .

II. Related Works

A. Sparse Support Vector Machine

Nowadays *SVM* has been widely used in many fields of machine learning. Its main idea is to construct a classifier which not only minimizes the error on the training dataset, but also maximizes the margin between different classes. Reader can refer to [1] for detailed formulation.

The *Sparse Support Vector Machine*(*SSVM*) we would like to mention was proposed by [5]. Its main idea is to constraint the cardinality of the parameter vector w as small as possible, where the cardinality of a vector denotes the number of the non-zero element of the vector. It is based on the implicit assumption that only a few number of the features are useful in constructing the goal classifier. The *SSVM* can be considered as a common *SVM* training process combining with a feature selection procedure. *SSVM* is superior to the standard *SVM* when the training dataset is dirty with noise.

B. Sparse Support Vector Regression

Support Vector Machine(*SVM*) can be easily extended for the regression scenario. The regression problem is to find a function $f : X \to Y$ that minimize the regularized risk functional ([7], [8], [9]):

$$R[f] := P[f] + C\frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i, f(x_i)), \tag{1}$$

where X, Y denotes the spaces of input and output respectively, ℓ is the number of the training samples and $L(\cdot)$ is a loss function ([4]). Usually $L(\cdot)$ takes the form of hinge loss, the regularization operator $P[\cdot]$ is to control the complexity of the model and C is a parameter to control the trade off between the two parts of the objective function $R[f]$.

If we choose the linear model then $f(x) = w^T x + b$, the regression becomes Support Vector Regression(SVR). Further more, if we import the kernel trick ([10]) then the function f becomes

$$f(x) = \sum \alpha_i \mathcal{K}(x_i, x) + b$$

and the optimal solution of w to SVM can be written in the following expression: $w = \sum \alpha_i \Phi(x_i)$. The *Sparse Support Vector Regression*(*SSVR*)([4]) changes the l_2 norm of vector α which used in the standard support vector regression to l_1 norm. The formulation ([4]) is

$$R[f] := \sum_{i=1}^{\ell} |\alpha_i| + C\frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i, f(x_i)) \tag{2}$$

if we use hinge loss then the formulation becomes

$$R[f] := \|\alpha\|_1 + C\frac{1}{\ell}\sum_{i=1}^{\ell} \max(|y_i - \sum \alpha_i \mathcal{K}(x_i, x) - b| - \epsilon, 0) \tag{3}$$

C. Semi-Supervised Learning

Semi-supervised Learning(*SSL*) is a framework of machine learning that it tries to train a classifier with a small number of labeled data and a huge number of unlabeled data. It is based on the implicit assumption that samples that are close to each other have high probability to belong to the same class.

If the abundant unlabeled data lies on manifolds ([11]), even though the number of labeled data is small we still can train a good enough classifier with the help of the manifold information. The general formulation of the *Belkin*'s manifold regularization of semi-supervised learning is as follows:

$$f^* = \arg\min_{f \in H_k} \frac{1}{\ell}\sum_{i=1}^{\ell} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2 \tag{4}$$

According to *Belkin*'s paper[12], $\|f\|_I^2$ is an appropriate penalty item that reflects the intrinsic structure of the distribution of the data x. The optimal solution of f is

$$f(x) = \sum_{i=1}^{\ell+u} \mathcal{K}(x_i, x)\alpha_i \tag{5}$$

where ℓ, u denote the number of labeled data and unlabeled data respectively. $\mathcal{K}(\cdot, \cdot)$ is the kernel function and α is the parameter vector.

III. Framework of Norm-Regularized Reconstructive Coefficients Learning

In last section we have reviewed several supervised and semi-supervised learning methods. We can see that the optimal solutions of *SVM*, *Belkin*'s framework and *SVR* are all taking the following form:

$$f(x) = \sum_{i=1}^{N} \alpha_i \mathcal{K}(x_i, x) \tag{6}$$

Inspired by the idea of the sparse support vector regression and the general equation (6) used in many formulations, we propose a novel learning framework. We call it "*Norm-Regularized Reconstructive Coefficients Learning*", "*NRCL*" for short. Our formulation is as follows:

$$\alpha^* = \arg\min_{\alpha} \quad \|\alpha\|_p + \lambda \sum_{i=1}^{N} L(f(x_i), y_i) + R[f] \tag{7}$$

where

$$f(x) = \sum_{i=1}^{N} \alpha_i \mathcal{K}(x_i, x) \tag{8}$$

$\mathcal{K}(x, y)$ is the kernel function. $\|x\|_p$ denotes the l_p norm of x, $R[f]$ denotes the regularization of function f. $L(x, y)$ is the loss function. If we use hinge loss for our loss function then the formulation of our framework becomes

$$\begin{aligned} \alpha^* &= \arg\min_{\alpha} \quad \|\alpha\|_p + \lambda \sum_{i=1}^{\ell} \xi_i + R[f] \\ &\text{s.t.} \quad y = K_\ell \alpha + \xi \\ &\xi_i \geq 0, \quad i = [1, \cdots, \ell] \end{aligned} \tag{9}$$

where y is the label vector, and the ith element of y is the label of x_i. K_ℓ is the first ℓ rows of the kernel matrix K which is constructed by the data $X = [x_1, \cdots, x_N]$. α is

the vector constructed by $\{\alpha_i\}_{i=1}^N$. ξ is the vector aligned by $\xi_i, i = [1, \cdots, \ell]$. If we use the square loss function, our formulation becomes

$$\alpha^* = \arg\min_{\alpha} \quad \|\alpha\|_p + \lambda \|y - K_\ell \alpha\|_2^2 + R[f] \tag{10}$$

In the following subsections we will mainly analyze our framework in the form of equation (10) as a special case.

A. Sparse Norm-Regularized Reconstructive Coefficients Learning

Our work in this subsection is mainly inspired by the good characteristic of *SVM*, where the reconstructive coefficients are enforce this characteristic explicitly. This is based on the very sparse, with the nonzero values only for the samples that are near to the decision boundary of the classifier. We thus want to assumption that only a small number of training samples contribute to the construction of the classifier.

In our framework a direct idea is using the l_0 norm of vector α in the objective function to make the final coefficients sparse. But to optimize the l_0 norm is an NP hard problem. According to [5] we can optimize the l_1 norm of vector α instead. Then the equation (7) becomes the follows:

$$\alpha^* = \arg\min_{\alpha} \quad \|\alpha\|_1 + \lambda \sum_{i=1}^{\ell} L(f(x_i), y_i) + R[f] \tag{11}$$

Also we can rewrite it in the following form when using square loss:

$$\alpha^* = \arg\min_{\alpha} \quad \|\alpha\|_1 + \lambda \|y - K_\ell \alpha\|_2^2 + R[f] \tag{12}$$

Different $R[f]$ will lead to different formulations of our framework. Inspired by the framework of *SSL* ([12]) mentioned above, one assumption is that if the dataset lies on manifolds, we can add a regularized item to guarantee the smoothness of the function on manifolds. And in order to control the complexity of model on *RKHS* (Representing Kernel Hilbert Space) ([13]), we can add another regularized item, then the equation (12) becomes

$$\begin{aligned}\alpha^* = \arg\min_{\alpha} \ & \|\alpha\|_1 + \lambda \|y - K_\ell \alpha\|_2^2 + \\ & \gamma_A \alpha^T K \alpha + \gamma_I \alpha^T K L K \alpha\end{aligned} \tag{13}$$

where L is the *Laplacian* matrix [14]. We call the above formulation as "*supervised SNRCL*". The *Laplacian* matrix is constructed as follows([15]): Given the data $\{x_i\}_{i=1}^N$, first we construct an undirected weighted graph $G = \{X, E\}$ with vertex set X and the edge set E. The edges are decided by the similarity of the vertexes. If the vertex x_i and x_j are close to each other, then there is an edge link the two vertexes, for detailed people can refer to paper([15]). If we give weights to the edges in the graph, then we can get the similarity matrix $W \in \mathbb{R}^{N \times N}$. The element of real symmetry matrix W measures the similarity of the vertex pair, which can be computed in different ways([15]). At last the *Laplacian* matrix L can be computed in the following way:

$$L = D - W \quad \text{where} \quad D_{i,i} = \sum_j W_{j,i} \tag{14}$$

B. Multi-Class Sparse Norm-Regularized Kernel Learning

In the above we only talked about the common regression and binary classification scenarios, it is easy for our framework to be extended to the multiple classification problem. Suppose our discriminative function changes to the following form:

$$f = K\boldsymbol{\alpha} \tag{15}$$

where $\boldsymbol{\alpha}$ is a $N \times M$ matrix, N is the number of the data and M is the number of the classes. The output of the function f is a vector. Thus our framework changes as follows:

$$\begin{aligned}\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \ & \|\text{vec}(\boldsymbol{\alpha})\|_1 + \|Y - K_\ell \boldsymbol{\alpha}\|_2^2 + \\ & \gamma_A \text{tr}(\boldsymbol{\alpha}^T K \boldsymbol{\alpha}) + \gamma_I \text{tr}(\boldsymbol{\alpha}^T K L K \boldsymbol{\alpha})\end{aligned} \tag{16}$$

Here Y becomes a indicate matrix and its size is $n \times m$. $Y_{i,j} = 1$ only x_i belongs to the jth class, otherwise $Y_{i,j} = 0$. $\text{vec}(\boldsymbol{\alpha})$ is the function that expands the matrix $\boldsymbol{\alpha}$ to a vector. $\text{tr}(\boldsymbol{\alpha})$ is to get the trace of the matrix $\boldsymbol{\alpha}$.

C. Semi-supervised Sparse Norm-Regularized Reconstructive Coefficients Learning

Obviously our framework can be easily extended to the semi-supervised learning scenario. The formulation is as follows:

$$\begin{aligned}\alpha^* = \arg\min_{\alpha} \ & \|\alpha\|_1 + \lambda \|y - K_\ell \alpha\|_2^2 + \\ & \gamma_A \alpha^T K \alpha + \gamma_I \alpha^T K L K \alpha\end{aligned} \tag{17}$$

We call it "*semi-supervised SNRCL*" for simplicity.

Belkin's framework[12] is a very effective method when the huge unlabeled data lies on manifold in semi-supervised learning. In format our framework is similar to the *Belkin*'s framwork, but we add more item $\|\alpha\|_1$ in the formulation for we not only want to use the manifold information but also to find the important unlabeled samples which is near the decision boundary, just as the *SVM* does.

D. Supervised Sparse Norm-Regularized Multi-Kernel Reconstructive Coefficients Learning

In the above we have only discussed the the decision function in the form of

$$f(x) = \sum_{i=1}^{\ell} \alpha_i \mathcal{K}(x_i, x) \tag{18}$$

where the kernel $\mathcal{K}(\cdot, \cdot)$ is a single fixed form such as *RBF* kernel. This limits the flexibility of our framework. A possible extended formulation of our framework is that we use a multiple kernel instead of the single one ([16]). We call it "*Supervised Sparse Norm-Regularized Multi-Kernel*

Reconstructive Coefficients Learning". The formulation of the multi-kernel function is as follows:

$$f(x) = \sum_{i=1}^{\ell} \alpha_i (\sum_{j=1}^{m} \beta_j \mathcal{K}_j(x_i, x)) \tag{19}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} \beta_j = 1, \quad \beta_j \geq 0$$

where each $\mathcal{K}_j(\cdot,\cdot)$ denotes one of the selected set of different kernel functions. This could be of two situations. One is that all the kernels within the selected set have the same form (i.e. they are all *RBF*-kernels) but different parameters. The other is that the kernels have different forms and different parameters. In fact, we can deem $\sum_{j=1}^{m} \beta_j \mathcal{K}_j(\cdot,\cdot)$ as a new kernel $\mathcal{K}'(\cdot,\cdot)$, then our formulation becomes

$$\begin{aligned} \alpha^* = \arg\min_{\alpha} \ & \|\alpha\|_1 + \lambda \|y - K'_\ell \alpha\|_2^2 + \\ & \gamma_A \alpha^T K' \alpha + \gamma_I \alpha^T K' L K' \alpha \end{aligned} \tag{20}$$

$$K' = \begin{pmatrix} \mathcal{K}'(x_1, x_1) & \cdots & \mathcal{K}'(x_1, x_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}'(x_N, x_1) & \cdots & \mathcal{K}'(x_N, x_N) \end{pmatrix}$$

E. Semi-supervised Sparse Norm-Regularized Multi-Kernel Reconstructive Coefficients Learning

It is natural for us to use multiple kernel in sparse kernel semi-supervised learning framework. We call it " *Semi-supervised Sparse Norm-Regularized Multi-Kernel Reconstructive Coefficients Learning*", and the formulation is as follows:

$$\begin{aligned} \alpha^* = \arg\min_{\alpha} \ & \|\alpha\|_1 + \lambda \|y - K'_\ell \alpha\|_2^2 + \\ & \gamma_A \alpha^T K' \alpha + \gamma_I \alpha^T K' L K' \alpha \end{aligned} \tag{21}$$

Also the framework can be used in multi-class sparse supervised kernel learning framework with little change for (21) like (16).

F. Quadratic Programming for Our Work

The l_1 norm in our formulation will significantly decelerate the computational speed of our method. In order to solve our formulation efficienttly, we rewrite $\alpha_j = u_j - v_j$, where $u_j, v_j \geq 0$, just as the paper[4]. The solution has either u_j or v_j equal to 0, which depends on the sign of α_j, so in our formulation $|\alpha_j| = u_j + v_j$. Let $u = [u_1, \cdots, u_N]$ $v = [v_1, \cdots, v_N]$, $x = [u\, v]^T$, Then our formulation changes as follows:

$$x^* = \arg\min_{x} \quad x^T H x + f^T x \tag{22}$$

$$\text{s.t.} \quad x_i \geq 0, \quad i = [1, \cdots, 2N]$$

where

$$H = \begin{pmatrix} K' & -K' \\ -K' & K' \end{pmatrix} \quad K' = \gamma_A K + \gamma_I K L K + \lambda K_\ell^T K_\ell$$

$$f = f_1 - f_2$$

$$f_1 = [\underbrace{1, \cdots, 1}_{N}, \underbrace{-1, \cdots, -1}_{N}]^T \quad f_2 = [-2\lambda y^T K_\ell, 2\lambda y^T K_\ell]^T$$

IV. Experiments

We compare the performances of the proposed sparse norm-regularized reconstructive coefficients learning methods such as *supervised SNRCL* with the *standard SVM* on five *UCI* data sets and sparse semi-supervised norm-regularized reconstructive coefficients learning methods such as *semi-supervised SNRCL*, with the *Laplacian SVM*(*LapSVM*) and *Laplacian Regularized Least Square Classification*(*LapRLSC*) on the synthetic *Two Moons Dataset* and the *Columbia Object Image Library Dataset*(*Coil20*).

A. Supervised Experiments

1) Data Sets: Our experiments were performed on the five *UCI* data sets listed in Table 1. Some of them are of binary classification setting such as *ionosphere*, *parkinson*. Others like *brown yeast*, *iris* and *wine* data sets are of multi-class classification setting. For convenience, we only do the binary classification tasks between several selected pairs of classes for the multi-class data set. In table 1 the number behind the name of data set means the selected pair of classes for binary classification. All data sets are available at [17], and the brown yeast data set is available at [18].

2) Experimental Setup: We split randomly with 80% of the data for training and cross-validation, and 20% held-out for testing. For *standard SVM*, we use *RBF* kernel for all the data sets, the optimal σ parameter and the optimal C parameter are selected using 5-fold cross-validation with 80% for training and 20% for validation over the range $\{2^{-6}, 2^{-5}, \cdots, 2^5, 2^6\}$. For our *supervised SNRCL*, the graph is constructed with parameters $k = 15$, $\sigma_L = 1000$, and the parameters γ_A, γ_I and λ are set by grid-search from 2^{-6} to 2^6. Finally, the cardinality of the α vector is computed by counting the number of relatively large magnitude elements of the coefficient vector α, i.e. the number of the elements with $|\alpha_j| \geq \max(\alpha) \cdot 10^{-5}$. The elements that less than $\max(\alpha) \cdot 10^{-5}$ are set to zero.

Each method was tested on the held-out samples, and the final results reported were averaged over 20 trials of different random splits of the data set. We used the *LibSVM*([19]) for our *standard SVM* and our method were trained using standard convex optimization toolbox like *CVX*([20]), *MOSEK*([21]).

3) Results: The overall experimental results on five *UCI* data sets are shown in Table 1. We can see that in many cases our method *supervised SNRCL* is superior to the *standard SVM*. Though the accuracy increase is very low, the cardinality of the coefficients α is far smaller than the *standard SVM*. In contrast our method had an average sparsity than *standard SVM*.

B. Semi-supervised Experiments

1) Synthetic data: Tow Moons dataset: The dataset contains 700 samples with 3% randomly labeled for each class. We found that by tuning the parameters, both our method and *LapRLS* can reach 100% of the predicting accuracy. The first two images in Figure 1 demonstrate the original datset and the randomly labeled samples. Compare the last two images in Figure 1, we can see that the cardinality of the coefficient vector α of our *SNRCL* method is sparser than *LapRLS*. In other words, the number of the samples with non-zero coefficients of the *LapRLS* method is more than that of ours. The right image in Figure 1 shows that the samples with non-zero coefficient of the *LapRLS* method are exactly the labeled instances. However, ours are not totally the labeled samples, while using some of the unlabeled samples. For details, our method not only uses the manifold information by the unlabeled data but also selects the unlabeled samples which are important to our final decision function.

2) Columbia Object Recognition: Coil20 is a data set of gray-scale images of 20 objects ([22]). For every object there are 72 images, so in Coil20 data set the number of the class is 20 and every class contains 72 instances, and each instance is composed of 1,024 elements. In our experiment we only use 10 classes of them to do our experiment.

We applied *LapSVM*, *LapRLSC* and *semi-supervised SNRCL* algorithms to 45 binary classification problems that arise in pairwise classification of Columbia Object Images. For each pairwise classification experiment we randomly labeled 20% of the samples and the parameters $\gamma_A, \gamma_I, \lambda$ are set by grid search from 2^{-6} to 2^6. The final results reported are averaged over 20 trials as Figure 4 shows. It also shows the cardinality of the the coefficient vector α of the applied methods.

From Figure 2 we can see that our method is superior to the *LapSVM* and *LapRLSC* for most of the cases. Also our methods has the advantage of lower cardinality of coefficient vector α. In contrast our method had an average sparsity of 200% than the former two algorithms.

V. Conclusion

In this paper we have proposed a novel framework for sparse coefficients learning. In supervised learning scenario we proposed the supervised sparse norm-regularized reconstructive coefficient learning method and we compared the *standard SVM* with our method in five *UCI* datasets. We found that our method not only increase the predicting accuracy but also get sparser solutions. In semi-supervised scenario we proposed the *semi-supervised SNRCL* method. Through several experiments we have found that our method can really find the unlabeled samples that are near the decision boundaries.

The implementation of our methods is quite slow when the number of the training samples is very large. Future work we will design a efficient algorithm to solve this problem.

Acknowledgment

The work is funded by Tsinghua National Laboratory for Information Science and TechnologyTNListCross-discipline Foundation.

References

[1] V. Vapnik, *The nature of statistical learning theory*. springer, 2000.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[3] K. Bennett and O. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software*, vol. 1, no. 1, pp. 23–34, 1992.

[4] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *The Journal of Machine Learning Research*, vol. 3, pp. 1229–1243, 2003.

[5] A. Chan, N. Vasconcelos, and G. Lanckriet, "Direct convex relaxations of sparse svm," in *Proceedings of the 24th international conference on Machine learning*. ACM New York, NY, USA, 2007, pp. 145–153.

[6] J. Mairal, M. Elad, G. Sapiro *et al.*, "Sparse representation for color image restoration," *IEEE Transactions on Image Processing*, vol. 17, no. 1, p. 53, 2008.

[7] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM New York, NY, USA, 1992, pp. 144–152.

[8] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," *Advances in Neural Information Processing Systems 9*, p. 281, 1997.

[9] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[10] B. Scholkopf, "The kernel trick for distances," *Advances in Neural Information Processing Systems*, pp. 301–307, 2001.

[11] M. Hirsch, C. Pugh, and M. Shub, "Invariant manifolds," 1977.

[12] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

[13] R. Rosipal and L. Trejo, "Kernel partial least squares regression in reproducing kernel hilbert space," *The Journal of Machine Learning Research*, vol. 2, pp. 97–123, 2002.

[14] B. Mohar, "The Laplacian spectrum of graphs," *Graph theory, combinatorics, and applications*, vol. 2, pp. 871–898, 1991.

Figure 1. Results on Two Moons Dataset: The first image(Sequence is from the left side to the right side) demonstrates the whole rand sampled dataset and the second one shows the labeled samples and the third one illustrates the samples whose coefficients are non-zero in LapRLS and the last one gives the results of SNRCL

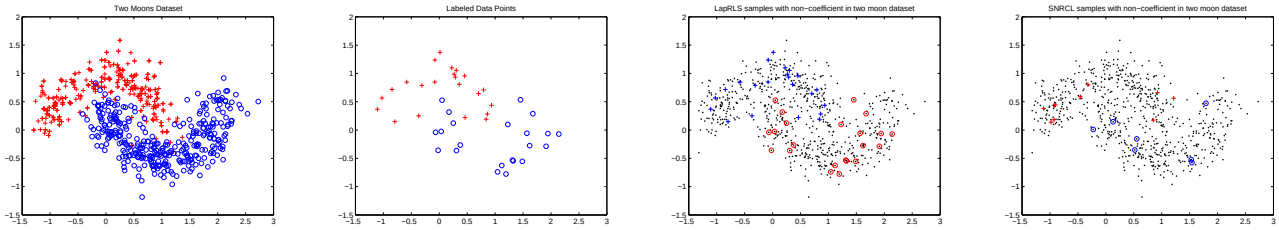
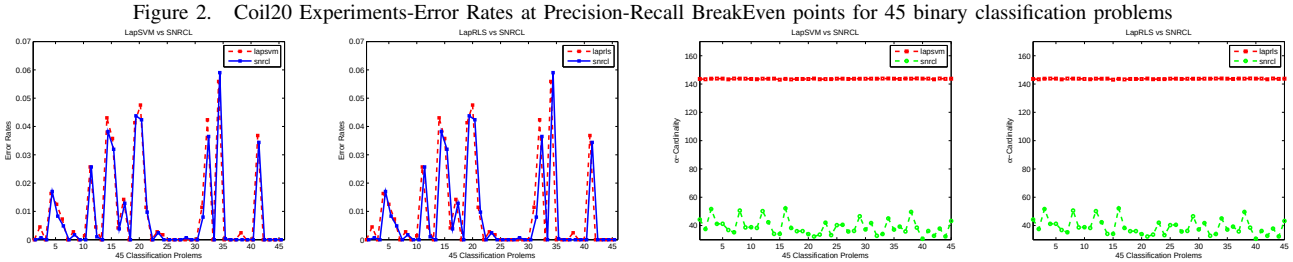


Figure 2. Coil20 Experiments-Error Rates at Precision-Recall BreakEven points for 45 binary classification problems



UCI data sets	SVM accuracy	SNRCL accuracy	SVM α-Card	SNRCL α-Card
Wine 12	0.9208 ± 0.0419	$\mathbf{0.9338} \pm 0.0467$	44.5900 ± 28.2403	$\mathbf{36.1600} \pm 2.6236$
Wine 13	0.9524 ± 0.0593	$\mathbf{0.9790} \pm 0.0307$	$\mathbf{11.4400} \pm 3.2335$	22.3000 ± 1.7871
Wine 23	0.9313 ± 0.0556	$\mathbf{0.9426} \pm 0.0434$	24.8800 ± 6.5641	$\mathbf{18.1200} \pm 1.4658$
ionosphere	$\mathbf{0.9431} \pm 0.0262$	0.9123 ± 0.0488	114.6800 ± 33.2860	$\mathbf{24.9200} \pm 2.0288$
parkinson	$\mathbf{0.8410} \pm 0.0606$	0.8285 ± 0.0487	94.3400 ± 13.8426	$\mathbf{53.7400} \pm 8.8475$
iris 12	1.0000 ± 0.0000	1.0000 ± 0.0000	3.2400 ± 0.5175	$\mathbf{2.4600} \pm 0.6455$
iris 13	1.0000 ± 0.0000	1.0000 ± 0.0000	3.2000 ± 0.5345	$\mathbf{2.7200} \pm 0.6713$
iris23	1.0000 ± 0.0000	1.0000 ± 0.0000	3.1800 ± 0.3881	$\mathbf{2.6400} \pm 0.6928$
yeast 34	0.9306 ± 0.0239	$\mathbf{0.9314} \pm 0.0212$	67.6600 ± 21.4786	$\mathbf{22.9200} \pm 1.7594$
yeast 45	0.9086 ± 0.0487	$\mathbf{0.9219} \pm 0.0319$	60.6000 ± 18.2130	$\mathbf{9.3800} \pm 1.2103$
yeast 35	0.9007 ± 0.0404	$\mathbf{0.9340} \pm 0.0383$	79.7800 ± 9.9083	$\mathbf{50.7200} \pm 2.6189$

Table I
RESULTS ON FIVE UCI DATA SETS

[15] D. Cvetkovic, M. Doob, and H. Sachs, *Spectra of graphs: Theory and applications*. Academic press New York, 1980.

[16] S. Sonnenburg, G. Ratsch, and C. Schafer, "A general and efficient multiple kernel learning algorithm," *Advances in Neural Information Processing Systems*, vol. 18, p. 1273, 2006.

[17] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998.

[18] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero norm with linear models and kernel methods," *The Journal of Machine Learning Research*, vol. 3, pp. 1439–1461, 2003.

[19] C. Chang and C. Lin, "LIBSVM: a library for support vector machines, 2001," *Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm*, 2001.

[20] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," *web page and software). http://stanford. edu/ boyd/cvx*.

[21] A. MOSEK, "The MOSEK Optimization Tools Version 2.5. Users Manual and Reference, 2002."

[22] S. Nene, S. Nayar, and H. Murase, "Columbia object image library (coil-20)," *Department of Computer Science, Columbia University, Technical Report CUCS-006-96*.