Modifications of k q-flats for supervised learning

Arthur Szlam

***Abstract*—**
In [KL93], and then in [Man98], [Tse99] the k-means algorithm is generalized to find the k best fit q-dimensional affine spaces to a set of points in R^d; the prototype for a set of points becomes the best fit plane, rather than the centroid. Here we discuss connections with sparse approximation and manifold learning, and suggest some modifications for supervised learning problems. We give experimental evidence showing that on some standard benchmark data sets for supervised learning, the original method gives nearly state of the art classification results in a small fraction of the time as its competitors. We then show how the method can be better adapted to learning problems by using Mahalanobis metrics as prototypes, generalizing affine sets. This modification allows for a very strong classifier, but at the cost of a longer run time than the original method.

***Index Terms*—q-flats, metric learning, Mahalanobis distance, sparse representation**

I. Introduction

The k q-flats algorithm [KL93], [Man98], [Tse99] is a generalization of the k-means algorithm where we take q-dimensional affine spaces ("flats") instead of points as prototypes. Thus given a set of n points $X \in \mathbb{R}^d$, we wish to find k q-dimensional flats $\{F_1, \ldots, F_k\}$ and a partition of X into $\{K_1, \ldots, K_k\}$ minimizing the energy

$$\sum_{j=1}^{k} \sum_{x \in K_j} ||x - F_j||^2, \qquad (1)$$

where $||x - F_j||$ is just the Euclidean distance of the point x to its projection on F_j.

The k q-flats algorithm can be used as a supervised classifier by training a dictionary of flats on each class separately, and assigning each new point to the class to which its nearest flat belongs. One of the purposes of this article is to remind the reader of the strength of this simple old method; on many data sets it gives excellent classification rates, especially relative to its speed; and the entire code can be written in just a few lines in matlab. However, there is much room for improvement. The k q-flats algorithm is representational; it does not explicitly see the differences between the classes. We will try to rectify this by changing the energy functional (1) to punish configurations of the flats passing through one class that get too close to points in another class.

In fact we will go farther than this, and generalize the k q-flats method to the k metric method, which replaces affine spaces, which can be thought of as very simple positive semi-definite matrices, by Mahalanobis metrics. We will run a generalized Lloyd algorithm: iterating between updating the metrics to be most discriminatory, and partitioning each class by associating points to the metric that they are smallest in. In other words, groups of labeled points will choose a warping of space under which they have small norm, and points from other classes have large norm; and then new groups will be chosen so that each point is paired with the warping under which it has the smallest norm.

We test this idea on some standard benchmarks, and we find that changing the energy functional and especially generalizing flats to metrics improves classification performance. We will see that our method outperforms many recent metric learning methods. However, the classification performance will come at a cost in run time, and the generalized algorithm, while extremely accurate on small UCI benchmarks, fails to be fast enough for large scale applications.

To balance this out, we also introduce a much faster but slightly less accurate version of k q-flats based on binary partitions. With this method we will be able to get close to the accuracy of neural networks and SVM's on the MNIST data set, but with a training time of roughly a minute on a desktop computer, with all code written in matlab.

II. k-q-flats, sparse representation, and manifold learning

Just as in k-means, we have the Lloyd algorithm:

- Initialize flats F_j, randomly or otherwise.
- Partition X into sets K_j by assigning each x to

$$argmin_j ||x - F_j||.$$

- Update the F_j by finding the L^2 best fit affine q-flat through the points in K_j.
- Repeat until convergence.

Just as for k-means, the Lloyd algorithm is guaranteed to converge to a local minimum.

A standard variant of the algorithm where all the F_j are constrained to pass through the origin often goes by the name of projective clustering. While the two versions of the algorithm are certainly not equivalent, any q dimensional affine set is contained in a $q+1$ dimensional linear subspace, and so one often does not loose too much intuition by thinking of the F_j as actual subspaces, especially when the ambient dimension is larger.

A. *k-q-flats and sparse representation*

It is a by now classical idea that problems of denoising, compression, and learning are closely related. The first two problems have often been approached by transform coding, which leverages the sparsity of the signals of interest, perhaps images or audio, in special bases, such as wavelets. In recent years, attention has been focused on adaptively choosing a basis so that a given class of signals is sparse in the constructed basis. If such a basis can be found, the classical signal processing techniques can be applied even though the signals under study are not well represented by the classical basis functions. A standard version of the sparse representation problem goes as follows: given

n vectors $X \in \mathbb{R}^d$, which we will write as columns (i.e. X is $n \times d$), find a basis B and coefficients C minimizing the error

$$||BC - X||_{\text{FRO}} \tag{2}$$

given that

$$||C_i||_0 \leq q, \tag{3}$$

where $||\cdot||_{\text{FRO}}$ is the Frobenious norm on matrices, i.e. L^2 on the matrix considered as a vector, and $||\cdot||_0$ is the L^0 norm, i.e. the number of nonzero entries. Several authors have noticed that the sparse representation problem is very closely related to the clustering problem, for example see [Tro04]. In fact, if one restricts the entries of C to be either zeros or ones, and sets $q = 1$, the problem as written above is exactly k-means [AEB06]. If we consider only q flats passing through the origin, the k-q-flats algorithm is an attempt at a minimizing (2) subject to (3) and an additional constraint forcing C to have a block structure. Each block corresponds to a P_j, as the elements in P_j are only allowed coefficients using basis elements in F_j.

B. Manifold learning

A recent trend in machine learning has been to use the fact that data from applications often has an intrinsic dimensionality which is far smaller than the dimension of the ambient space in which it is presented. One often speaks of the data lying on a low dimensional manifold. A salient feature of a set with manifold structure is that it should be locally well approximated by affine spaces, namely the tangent spaces. If we know that there is some good approximation of the set by affine spaces, it makes sense to try to find the best one given some information theoretic constraints. This is exactly the purpose of k q-flats; the k and the q serve as bounds on the capacity.

Thus the k q-flats construction can be thought of as a method of building adapted dictionaries, which gives a notion of the distance from a point to set of points in such a way that meaningful information is abstracted; and of finding a best piecewise approximation to a data manifold, and so parameterizing a given data set. It thus sits in the intersection of the notions of sparse representation and manifold learning (for a nice look at the interplay between these sets of ideas, see [Wak06]), two of the most successful signal processing paradigms. In this article we will demonstrate its utility for supervised learning and show how it can be modified to better serve this task.

III. Sparse representation for classification

If an adaptive dictionary learning algorithm is doing its job, and we have a supervised learning problem where the classes can be expected to have fundamentally different representations, we can attempt to solve the problem by training a dictionary for each class separately, and then associating each test sample to the dictionary in which it is best represented. We record this algorithm as supervised k q-flats:

1. For each class C_i, train a dictionary $\{F_1^i, ..., F_k^i\}$ of q-dimensional affine sets on the points of class C_i.
2. For each unlabeled point, compute
$$d_{C_i}(x) = \min_j ||x - F_j^i||.$$
3. Classify x by
$$arg \min_i d_{C_i}(x).$$

There are many instances where different classes can be expected to have different representations. For example, different textures in images, and different musical instruments. In general, one might expect "manifold" type data to be well approximated by the k-q-flats algorithm, as it would be searching for an optimal set of k "secant" planes, given a bound on the "capacity" of the planes. The key here for classification is of course that the different classes are compressible in different dictionaries, not necessarily that the data as a whole is compressible. In fact the simplest example of manifold like data, say a line in $\mathbb{R}^d$, and two class problem, say everything on one side of the line is class one, and on the other is class two, would be a serious challenge for the supervised k q-flats. However, while in low dimensions we imagine planes to as intersecting, in high dimensions the vast amount of space makes this unlikely, and thus situations as the one described should be unusual; it is simply a result of the data being linearly embeddable in a very low dimensional space. The digit1 data set from [CSZ06], which consists of various images of the numeral one at different angles, jittered, and noised, has classes parameterized by the angular variable (positive angle is class one, and negative angle is class two), is a much more realistic example of what data parameterized by a single variable looks like. As a function of angle, it traces out a "Hilbert spiral", that is, each tangent line points in a new direction, and the best fit q-flats passing through different pieces of the set do not intersect.

A situation which is deadly to supervised k q-flats is if the classes are truly of the ambient dimension. For example consider two Gaussian clouds as the two classes of the data set. In this case, the algorithm is almost guaranteed to fail; only an exponentially small percentage of the possible orientations of any affine subspace passing through part of the cloud can see the difference between the Gaussians. The problem here is that for nice partitions of the data, there is no preferred orientation to the partition elements. The algorithm can only handle clusters if they are somehow oriented in different directions, or at least nice pieces of them are.

On the other hand, we will see experimentally that there are natural data sets with classes that so that the local orientation is an important descriptor of the class labels.

IV. Beyond representation

As mentioned above, supervised k q-flats classifies solely by best representation; it is a sophisticated form of pattern matching. We place k q-flats through the points in each class C_i so that the distance from a point $x \in C_i$ to the nearest flat to x associated to C_i is as small as possible. Supervised learning algorithms are generally expected to do more than this, and explicitly understand the differences between the classes. What we should be doing for classification is finding k q-flats for each C_i so that the distance from each point $x \in C_i$ to the nearest flat associated to C_i is smaller than the distance to the nearest flat associated to any other C_k, $k \neq i$. In this section we will explore some of the challenges (opportunities!) associated to this, and make some first useful steps.

To accomplish our goal, we of course should modify the energy in supervised k q-flats so that it penalizes enemy points from C_k getting close to flats associated to C_i. We will do this; however we also generalize the prototype we associate to a set of points from a flat to a positive semidefinite matrix A^i_j, or a Mahalanobis metric. If $x, y \in \mathbb{R}^d$, a Mahalanobis metric modifies

$$||x-y||^2 = (x-y)^T(x-y)$$

to

$$||x-y||^2_A = (x-y)^T A(x-y),$$

where A is a positive semi-definite. The distance associated to A linearly crushes some directions more than others. Several recent papers have studied methods for finding Mahalanobis metrics to improve classification by *k-nn*; see [WBS06], [DKJ$^+$07], [XNJR03], and the references therein.

Projecting onto a flat F passing through the origin is a very simple choice of A, namely $A = I - P_F$, where P_F is matrix for the projection onto the flat. We can thus greatly generalize supervised k q-flats is by choosing k metrics A^i_j, centered at the points m^i_j for each class; we can think of the A^j_i as fuzzy secant planes passing through the data manifold. The aim in the classification problem would be to choose the A^i_j so that

$$(x - m^i_j)^T A(x - m^i_j) < \text{small}, \;\; x \in C_i,$$

and in particular,

$$(x - m^i_j)^T A(x - m^i_j) < \text{very small}, \;\; x \in K^i_j;$$

and

$$(x - m^k_l)^T A(x - m^k_l) > \text{big}, \;\; x \notin C_k.$$

We emphasize to the reader the difference between this approach and the metric learning approach mentioned earlier: we are not searching for A^i_j so that points in different classes are far from each other and points in the same class are close, but rather we want A^i_j so that the norm of points in C_i belonging to K^i_j is small with respect to A^i_j, and the norm with respect to A^i_j of points in C_k, $k \neq i$ is large. As before, we take an abstraction of a number of points and collect them into a larger object that test points can be measured against; before it was an affine set, now it is a Mahalanobis metric.

We introduce the schematic algorithm which we could run on each class:

- Initialize centers of mass m^i_j, and metrics A^i_j.
- Partition C_i into sets K^i_j by associating $x \in C_i$ to the A^i_j such that $(x - m^i_j)^T A^i_j (x - m^i_j)$ is smallest.
- Solve optimization problem $\tilde{A}^i_j = \mathcal{O}$,
- Set $A^i_j = \tilde{A}^i_j$.
- Repeat until satisfied.

A reasonable choice for $\mathcal{O}$ in the third step is given by $\mathcal{O}_1$:

$$arg \min_A \sum_{x \in K^i_j} (x - m^i_j)^T A^i_j (x - m^i_j),$$

given

$$(y - m^i_j)^t A^i_j (y - m^i_j) > 1,$$

and A positive semidefinite, where y is a subsset of $X \backslash C_i$, say a small number of the y closest to the old metric corresponding to i and j (note the number 1 is irrelevant, as scaling the number will just scale the output). This is a semidefinite program in the variable A; that is, the objective and constraints are linear in A, with the exception of the constraint that A be positive semidefinite. It is a simplification of the optimization considered in [XNJR03], as we do not need to consider pairwise distances; only norms of points. We will refer to the schematic algorithm with this choice of optimization as k-metric.

While there exist reasonably efficient methods for solving such programs, we need to solve them many times, unlike in the metric learning papers, where they only need to be solved once for given data and labels; and while the size of each problem is smaller than in the standard metric learning setting, we have found that overall the method is quite slow. On the other hand, there are many different optimization objectives for the learning of the metric, for example [WBS06], [DKJ$^+$07] which advertise faster performance than [XNJR03]; it will be interesting to see how the well the relevant simplifications of their methods perform in our context.

It is also unfortunate that for many choices of optimization objective in the schematic algorithm the LLoyd iterations are not guaranteed to converge to a local minimum. In the standard Lloyd algorithm, associating each point to its nearest center decreases the total energy, and the choice of centers also decreases the total energy. In the k metric algorithm we will use in the experiments, while associating each point to its nearest metric decreases the energy, when we update A^i_j, because we only use the nearest few $y \notin C_i$ as constraints, the constraints are different from one iteration to the next. Thus the energy can actually increase in the optimization step.

In any case, we can test Algorithm 2 with the objective above on some of the small data sets in the UCI repository, where it shows itself to be an extremely strong classifier. Allowing different regions of space to be under the influence of different Mahalanobis metrics allows a very flexible trainable geometry, but by classifiying a point based on its distance to a given metric, which embodies the opinions of many points, we keep from overtraining.

For large scale data sets it is not practical to use optimization $\mathcal{O}_1$. Since we know that on many data sets, supervised k q-flats works well, it might be possible to do something useful without searching the full generality of all positive semidefinite A, and simply try to nudge the q-flats given by supervised k q-flats in a discriminatory direction. We can use the optimization $\mathcal{O}_2$:

$$arg \min_F \sum_{x \in P^i_j} (x - m^i_j)^t (I - \mathbf{P}_F)^i (x - m^i_j)$$
$$-\alpha \sum_{y \notin P^i_j} (y - m^i_j)^t (I - \mathbf{P}_F)^i (y - m^i_j),$$

where α is a parameter. Without the semidefinite constraint, the problem is easily solved with an SVD. Because for each K^i_j we are more worried about y not in C_j which are close to it, as above, we only take the y closest to the previous flat.

This algorithm is non-negligibly slower than supervised k q-flats (if we take as many y as x, each SVD requires twice as many operations; and it does not converge as fast), but often gives non-negligible improvement in classification accuracy. We will call this method modified k q-flats.

V. Running even faster

We have discussed how to improve the classification accuracy of supervised k q-flats at the cost of run time; so it is natural for us to also discuss the reverse. A standard method for initializing k-means is to use binary subdivision. First subdivide X into two pieces with 2-means, than repeat on each piece, etc. Under certain conditions, one can give guarantees on the quality of the partition elements after several iterations; then it is not even necessary to use the partition obtained by subdivision as an initialization for the full algorithm

We can use the same technique here: simply run 2 q-flats on the whole data set, and then 2 q-flats on each of the pieces, etc. We will call this binary subdivision q-flats. One of the main costs of supervised k q-flats is computing the distances of all the points to all of the flats (however, see [BHZM07] to see progress towards a direct solution to this particular problem); and so only having two flats at each step is a major savings. On the other hand, we see that when we use while classification error using this dictionary training does increase, it is still comparable to the standard k q-flats.

It is well known that the first principal component vector of a set of points is the real-relaxed solution to the 2-means objective [DFK$^+$], [ZDG$^+$01]. Using this, we can quickly approximate the 2-means clustering by taking a single SVD. We wonder if there is some equivalent result here?

VI. Experiments

We will do two sets of experiments. The first are supervised learning benchmarks on larger data sets where we will not include k metrics; and the second are supervised learning experiments on small data sets from the UCI repository, and will not include binary subdivision q flats.

A. Larger data sets

We work on two large data sets. First, the MNIST, which is available at `http://yann.lecun.com/exdb/mnist/` with an extensive list of the results on the data set with various classifiers. The data consists of $70,000$ 28×28 images of the digits 0 through 9. The first $60,000$ are usually considered the training set, and the other $10,000$ are the test set. We also work on the Isolet data set, available at `http://archive.ics.uci.edu/ml/machine-learning-databases/isolet/`. This data set consists of 617 sonic features extracted from 50 speakers saying each letter of the alphabet twice. The first forty speakers are the training set, and the last ten are the test set; the task is to identify which letter is being spoken.

sup. k q-flats

	5	10	20
10	3.2	2.3	2.2
50	2.2	1.7	1.8
100	2.0	1.6	1.9

mod. k q-flats

	5	10	20
10	3.1	2.2	1.7
50	2.2	1.6	1.7
100	2.0	1.5	1.6

bin. sub. q-flats

	5	10	20
8	3.7	2.5	2.4
64	2.3	1.7	1.8
128	2.1	1.7	2.0

Fig. 1. Classification errors on MNIST in percent, averaged over 10 runs, using the standard 60000/10000 train/test split. The rows are the number of q-flats, and the columns are the values of q. Nearest neighbor baseline is 2.4%, a good SVM 1.2%.

On each of the large data sets, we test the performance of the algorithms mentioned in the supervised. For each data set, we use the standard splits for the supervised problem. We reduce the MNIST to 50 dimensions using principal components, and the isolet data set to 300 dimensions, and both are projected onto their respective unit spheres. For the supervised tests: we run supervised k q-flats with $q = 5, 10, 20$ and $k = 10, 50, 100$ on MNIST, and with $q = 5, 10, 20$ and $k = 2, 5, 10$ on isolet. We initialize supervised k q-flats by picking k points in each class, and choosing the initial flats as the best fit affine space to the $q+2$ nearest neighbors of the chosen points. Note that very few neighbor calculations are actually done, and no special nearest neighbor searcher is used. Modified k q-flats and binary subdivision k q-flats are initialized in the same way. The parameter α in the modified k q-means is set to .2, and we use twice the number of $y \notin C_j$ as $x \in K^i_j$ in the computation of the energy; in addition, the covariance matrix of the x and the y are normalized to unit norm. There is no image processing of the data set prior to running the benchmarks, and the algorithms we have described have no idea that they are being fed images. We stop the Lloyd algorithm after 40 iterations if a local minimum has not yet been found.

Some comments on the results: first, the algorithms tend to do better with more principal components, even up to not reducing the dimension at all; and usually below a certain dimension, the classification performance becomes quite bad. However, there is a steep cost in runtime if the dimension is not first reduced with PCA. In both data sets we used the least number of principal components before sup. k q-flats

	5	10	20
10	56	60	65
50	158	146	131
100	234	187	163

mod. k q-flats

	5	10	20
10	139	154	195
50	413	475	862
100	653	709	1460

bin. sub. q-flats

	5	10	20
8	38	40	43
64	65	62	68
128	69	70	71

Fig. 2. Computation time in seconds for training the q-flats for all the classes on MNIST, averaged over 10 runs. The rows are the number of q-flats, and the columns are the values of q.

sup. k q-flats

	5	10	20
2	5.7	5.1	4.8
5	5.9	5.3	5.9
10	5.8	5.8	5.8

mod. k q-flats

	5	10	20
2	5.9	4.6	4.3
5	5.7	4.9	5.1
10	5.8	5.3	5.1

bin. sub. q-flats

	5	10	20
2	6.0	5.1	5.0
4	5.7	5.5	5.5
8	6.1	5.6	5.6

Fig. 3. Classification errors on isolet in percent, averaged over 10 runs, using the standard 6328/1559 train/test split. The rows are the number of q-flats, and the columns are the values of q. Nearest neighbor baseline error is 8.6%, and SVM is 3.3%

the performance started to drop off. If faced with an unknown data set, our experience suggests to use as many dimensions as you have time for.

B. Smaller data sets

We also test the performance of Algorithm 3 on some small data sets from the UCI database [AN07]. We use the Iris, Wine, Balance-Scale, and Ionosphere data sets, all available at `http://archive.ics.uci.edu/ml/`. Each data set is less than 600 points, and the first three have 3 classes, and the Ionosphere has 2. The Iris data set lives in $\mathbb{R}^3$, Wine in $\mathbb{R}^{13}$, Balance-Scale in $\mathbb{R}^4$, and Ionosphere in $\mathbb{R}^{33}$.

For each of the small data sets, we run supervised k q-flats, modified k q-flats, and k metrics. The test results were averaged over 20 random 50/50 splits (train/test). The choice of q and k and whether or not to project onto the unit sphere are chosen by leave-10%-out cross validation, with q and k chosen from $\{1, 2\}$; we stop the Lloyd algorithm at 40 iterations, as before. In the computation of the metric A^i_j, we use twice the number of $y \notin C_j$ as $x \in K^i_j$, and the algorithm is initialized using the results from supervised k q-flats. We display the results, in figure 5. The results of k metrics are among the best or better than the best of the equivalent experiments in [WBS06] and [DKJ+07]. We also display the run times, per training, in figure 6 (note that the train size is larger for the timing experiments to compare with [WBS06]). As above, supervised k q-flats is coded in matlab; and the SDP in Algorithm 3 is run using sedumi. While we are not faster than [WBS06] and [DKJ+07], we are competitive, and each of those papers use specialized methods for the solution of their optimization problems which we can almost certainly appropriate.

sup. k q-flats

	5	10	20
2	8	6	4
5	6	5	5
10	8	8	10

mod. k q-flats

	5	10	20
2	58	38	28
5	55	41	36
10	45	42	50

bin. sub. q-flats

	5	10	20
2	8	6	4
4	11	8	7
8	13	10	9

Fig. 4. Computation time in seconds for training the q-flats for all the classes on isolet. The rows are the number of q-flats, and the columns are the values of q.

	k q-flats	mod. k q-flats	k-met
Wine	14.2	12.4	2.9
Balance	16.5	15.7	5.6
Ionosphere	10.0	9.6	8.5
Iris	5.6	3.7	3.5

Fig. 5. Average classification errors (in percent) of various algorithms on data sets from the UCI repository over 20 random 50/50 (test/train) runs.

	k q-flats	mod. k q-flats	k met.
Wine	.02	.02	4.9
Balance	.02	.04	12.6
Ionosphere	.02	.04	20.2
Iris	.01	.02	2.3

Fig. 6. Average slowest (over $k, q \in \{1, 2\}$) runtime in seconds of various algorithms on data sets from the UCI repository over 20 random 70/30 (test/train) runs.

VII. Conclusions and future work

We have argued that an old clustering algorithm, viewed as a supervised classifier, can in many ways hold its own against the current state of the art. The method is interesting not only because of the results it gives on benchmarks, but also because it sits at the intersection of sparse representation and manifold learning. We have demonstrated that the method can be sped up a great deal without loosing too much accuracy as a classifier. Finally, we have further shown experimentally that modifiying the method by generalizing projection onto planes to an inner product against a semi-definite matrix allows a classifier which often outperforms metric learning based algorithms which use a single metric.

There of course remains much to explore here. It seems clear that we can speed up the k metrics algorithm, perhaps by a better choice for the optimization objective, even perhaps some of the objectives discussed in[WBS06], [DKJ+07]. We have not talked about kernelization at all, although that could be a powerful extension.

References

[AEB06] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[AN07] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[BHZM07] Ronen Basri, Tal Hassner, and Lihi Zelnik-Manor. Approximate nearest subspace search with applications to pattern recognition. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[CSZ06] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning.* MIT Press, Cambridge, MA, 2006.

[DFK+] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Mach. Learn.*, 56(1-3):9–33.

[DKJ+07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007.

[KL93] Nanda Kambhatla and Todd K. Leen. Fast non-linear dimension reduction. In *NIPS*, pages 152–159, 1993.

[Man98] P. S. Bradley & O. L. Mangasarian. k-plane clustering. Technical Report MP-TR-1998-08, 1998.

[Tro04] J. Tropp. *Topics in sparse approximation.* PhD thesis, Computational and Applied Mathematics, The University of Texas at Austin, May 2004.

[Tse99] P. Tseng. Nearest q-flat to m points. Technical report, Seattle, WA, 1999.

[Wak06] M. Wakin. *The Geometry of Low-dimesnional Signal Models.* PhD thesis, Rice university, August 2006.

[WBS06] Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA, 2006.

[XNJR03] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information, 2003.

[ZDG+01] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering, 2001.