Piece-wise quadratic lego set for constructing data approximation potentials and their fast optimization

Alexander N. Gorban, Evgeny M. Mirkes, Andrei Zinovyev

Leicester, Paris

Abstract

Data dimension reduction by constructing low-dimensional approximators is one of the most fundamental approaches in data mining. Most efficient approximators based on quadratic energy functional are not flexible enough in many circumstances and suffer from sensitivity to outliers and dimensionality curse, which led to introducing other functional forms of energy potential that are usually computationally expensive to minimize. We suggest using piece-wise quadratic energy potentials of subquadratic growth (PQSQ potentials) that can imitate a variety of approximation metrics used in practice, such as the popular $L1$-norm or even sub-linear potentials. A family of subquadratic piece-wise potentials are almost as computationally efficient in numerical optimization as quadratic ones for the most popular data approximators (k-means, principal components, principal graphs), has guaranteed convergence to the global or local energy minimum, allows flexible choice of data approximation metrics and can be naturally robust to outlier data points. We introduce this family of potentials, provide implementations of several popular data approximators exploiting them and do benchmarking.

Keywords: data approximation, subquadraic potential, principal components, clustering

1. Introduction

Data dimension reduction by constructing low-dimensional approximators of finite set of vectors is one of the most fundamental approach in data analysis. Starting from the classical data approximators such as k-means and linear principal components (PCA), multiple generalizations have been

suggested in the last decades (principal manifolds, principal graphs, principal trees, etc.)[1].

We solve the problem of approximating a finite set of vectors $\vec{x}_i \in R^m, i = 1...N$ (data set) by a simpler object L embedded into the data space, such that for each point $\vec{x}_i$ an approximation error $err(\vec{x}_i, L)$ function can be defined. We assume this function in the form

$$err(\vec{x}_i, L) = \min_{y \in L} \sum_k u(x_i^k - y^k), \tag{1}$$

where the upper $k = 1...m$ stands for the coordinate index, and $u(x)$ is a monotonously growing symmetrical function, which we will be calling the energy potential. By data approximation we mean that the configuration of L in the data space minimizes the energy

$$\sum_i err(\vec{x}_i, L) \to \min .$$

The simplest form of the energy potential is quadratic $u(x) = x^2$, which leads to the most known data approximators: mean point (L is a point), principal points (L is a set of points) [**?**], principal components (L is a line or a hyperplane). In more advanced cases, L can posses some regular properties leading to principal curves (L is a smooth line or spline), principal manifolds (L is a smooth low-dimensional surface) and principal graphs (eg., L is a pluri-harmonic graph embedment) [2].

There exist multiple advantages of using quadratic potential $u(x)$, because it leads to the most computationally efficient algorithms usually based on a splitting schema, a variant of Expectation-Minimization approach [2]. For example, k-means algorithm solves the problem of finding the set of principal points and the standard iterative Singular Value Decomposition finds principal components. However, quadratic potential is known to be sensitive to outliers in the data set.

Iteratively reweighted least squares [3]. A Pure L1-norm Principal Component Analysis [4].

2. Piecewise quadratic potential of subquadratic growth (PQSQ)

2.1. Definition of the PQSQ potential

Let us split all non-negative numbers $x \in R_{\geq 0}$ into $p+1$ non-intersecting intervals $R_0 = [0; r_1), R_1 = [r_1; r_2), ..., R_k = [r_k; r_{k+1}), ..., R_p = [r_p; \infty)$, for

a set of thresholds $r_1 < r_2 < ... < r_p$. For convenience, let us denote $r_0 = 0, r_{p+1} = \infty$. Piecewise quadratic potential is a continuous monotonously growing function $u(x)$ constructed from pieces of centered at zero parabolas $y = b_k + a_k x^2$, defined on intervals $x \in [r_k, r_{k+1})$ in the following way (see Figure 1):

$$u(x) = \Big\{ b_k + a_k x^2, \text{if } r_k \leq |x| < r_{k+1}, k = 0...p, \tag{2}$$

$$a_k = \frac{f(r_k) - f(r_{k+1})}{r_k^2 - r_{k+1}^2}, \tag{3}$$

$$b_k = \frac{f(r_{k+1}) r_k^2 - f(r_k) r_{k+1}^2}{r_k^2 - r_{k+1}^2} \tag{4}$$

where $f(x)$ is the majorating function, which is to be approximated (imitated) by $u(x)$. For example, in the simplest case $f(x)$ can be a linear function : $f(x) = x$, in this case, $\sum_k u(x^k)$ will approximate the $L1$-based error function.

Note that accordingly to (3,4), $b_0 = 0, a_p = 0, b_p = f(r_p)$. Therefore, the choice of r_p can naturally create a "trimmed" version of energy potential $u(x)$ such that some data points (outliers) would not have any contribution to the gradient of $u(x)$, hence, will not affect the optimization procedure.

The condition of subquadratic growth consists in the requirement $a_{k+1} \leq a_k$ and $b_{k+1} \geq b_k$. To guarantee this, the following simple condition on $f(x)$ should be satisfied:

$$f' > 0, \quad f'' x \leq f', \tag{5}$$

i.e., $f(x)$ should grow not faster than any parabola $ax^2 + cx, c > 0$.

2.2. Basic approach for optimization

The definition of approximation error (1) implies that any optimization procedure based on this approximation measure can be done independently for each coordinate. In order to use the PQSQ potential in an algorithm, two ingredients should be pre-computed:

(1) set of p interval thresholds $r_s^k, s = 1...p$ for each coordinate $k = 1...m$.

(2) Matrix of a-coefficients computed by (3) based on interval definitions: a_s^k, $s = 0...p$, $k = 1...m$ separately for each coordinate k.

Minimization of PQSQ-based functional consists in two steps:

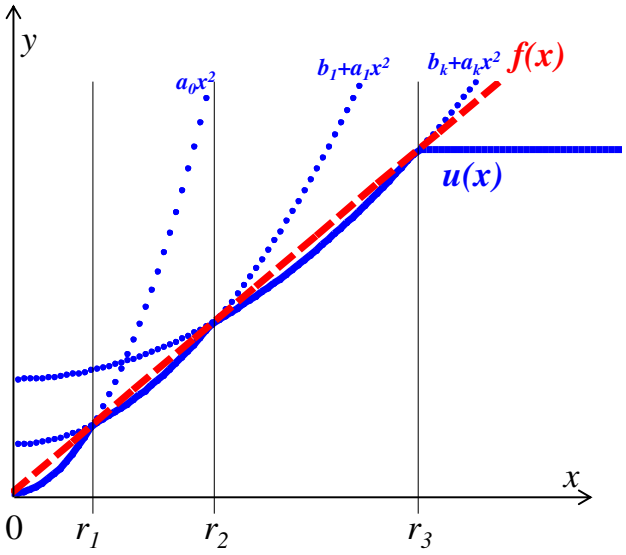

Figure 1: Trimmed piecewise quadratic potential of subquadratic growth $u(x)$ (solid blue line) defined for the majorating function $f(x)$ (red dashed line) and several thresholds r_k. Dotted lines shows the parabolas which fragments are used to construct $u(x)$.

(1) For each coordinate k, assign data point indices into non-overlapping sets $\mathcal{R}_s^k$:

$$\mathcal{R}_s^k = \{i : r_s^k \le |x_i^k - \beta_i^k| < r_{s+1}^k\}, s = 0...p, \tag{6}$$

where β is a matrix which depends on the nature of the algorithm.

(2) Minimize PQSQ-based functional for each coordinate k independently, where each set of points $\{x_{i\in\mathcal{R}_s^k}\}$ contributes to the functional quadratically with coefficient a_s^k. This is a quadratic optimization task.

(3) Repeat (1)-(2) till convergence which is guaranteed by the definition (2) of $u(x)$.

2.3. Mean value and k-means clustering in PQSQ approximation measure

Mean vector $\bar{X}_L$ for a set of vectors $X = \{x_i^k\}$, $i = 1...N, k = 1...m$ and an approximation error defined by potential $u(x)$ can be defined as a point minimizing the mean energy potential for all points in X:

$$\sum_i \sum_k f(x_i^k - \bar{X}^k) \to \min. \tag{7}$$

For Euclidean metrics L_2 ($u(x) = x^2$) it is the usual arithmetric mean.

For L_1 metrics ($u(x) = |x|$), (7) leads to the implicit equation $\#(x_i^k > \bar{X}^k) = \#(x_i^k < \bar{X}^k)$, where $\#$ stands for the number of points, which corresponds to the definition of median. At the same time this equation can not have a unique solution in case of an even number of points or when some data point coordinates coincide: therefore, definition of median is usually accompanied by heuristics used for breaking ties, i.e. to deal with non-unquiely defined rankings. This situation reflects the general problem of existence of multiple local minimuma and possible non-uniqueness of global minimum of (7) [].

For PQSQ approximation measure (2) it is difficult to write down an explicit formula for computing the mean value corresponding to the global minimum of (7). In order to find a point $\bar{X}_{PQSQ}$ minimizing mean $PQSQ$ potential, a simple iterative algorithm can be used:

Algorithm 1 Computing PQSQ mean value

procedure PQSQ MEAN VALUE
  define intervals $r_s^k, s = 0...p, k = 1...m$
  compute coefficients a_s^k
  initialize $\bar{X}_{PQSQ}$ *: eg., by arithmetic mean*
repeat till convergence of $\bar{X}_{PQSQ}$:
  for each *coordinate* k
  define sets of indices

$$\mathcal{R}_s^k = \{i : r_s^k \leq |x_i^k - \bar{X}_{PQSQ}^k| < r_{s+1}^k\}, s = 0...p$$

  compute new approximation for $\bar{X}_{PQSQ}$:

$$\bar{X}_{PQSQ}^k \leftarrow \frac{\sum_{s=1...p} a_s^k \sum_{i \in \mathcal{R}_s^k} x_i^k}{\sum_{s=1...p} a_s^k |\mathcal{R}_s^k|}$$

  end for
  goto *repeat till convergence*

The suggested algorithm converges only to the local minimum which depends on the initial point approximation.

Based on the PQSQ approximation measure and the algorithm for computing the PQSQ mean value (1), one can construct the PQSQ-based k-means clustering procedure in the usual way, splitting estimation of cluster centroids given partitioning of the data points into k disjoint groups, and then re-calculating the partitioning using the PQSQ-based proximity measure.

2.4. Principal Component Analysis (PCA) in PQSQ metrics

Accordingly to the classical definition of the first principal component, it is a line best fit to the data set X [5]. Let us define a line in the parametric form $\vec{y} = \vec{V}u + \vec{\delta}$, where $u \in R^1$ is the parameter. Then the first principal component will be defined by vectors $\vec{V}, \vec{\delta}$ satisfying

$$\sum_i \sum_k f(x_i^k - V^k u_i - \delta^k) \to \min, \tag{8}$$

where

$$u_i = \arg\min_s \sum_k f(x_i^k - V^k s - \delta^k). \tag{9}$$

The standard first principal component (PC1) corresponds to $u(x) = x^2$ when the vectors $\vec{V}, \vec{\delta}$ can be found by a simple iterative splitting algorithm for Singular Value Decomposition (SVD). If X does not contain missing values then $\vec{\delta}$ is the vector of arithmetic mean values. By contrast, computing $L1$-based principal components ($u(x) = |x|$) represents a much more challenging optimization problem [4]. Several approximative algorithms for computing $L1$-norm PCA have been recently suggested and benchmarked []. There was no general efficient algorithm suggested for computing PCA in case of arbitrary approximation measure for some monotonous function $u(x)$.

Computing PCA based on PQSQ approximation error is only slightly more complicated than computing the standard $L2$ PCA by SVD. We provide a pseudo-code (**Algorithm 2**) of a simple iterative algorithm (similar to **Algorithm 1**) with guaranteed convergence.

Computation of second and further principal components follows the standard deflation approach: projections of data points onto the previously computed component are subtracted from the data set, and the algorithm is applied to the residues. However, as it is the case in any non-quadratic metrics, the resulting components can not be orthogonal anymore. Moreover, unlike $L2$-based principal components, the algorithm 2 does not always converge to a unique global minimum; the computed components can depend on the initial estimate of $\vec{V}$. The situation is somewhat similar to the standard k-means algorithm. Therefore, in order to achieve the least possible approximation error to the linear subspace, $\vec{V}$ can be initialized randomly or by

Algorithm 2 Computing PQSQ PCA

procedure PQSQ First Principal Component
  define intervals $r_s^k, s = 0...p, k = 1...m$
  compute coefficients a_s^k
  $\vec{\delta} \leftarrow \bar{X}_{PQSQ}$
  initialize $\vec{V}$ *: eg., by L2-based PC1*
  initialize $\{u_i\}$ *: eg., by* $u_i = \frac{\sum_k V^k(x_i^k - \delta^k)}{\sum_k (V^k)^2}$
repeat till convergence of $\vec{V}$:
  normalize $\vec{V}$ *:* $\vec{V} \leftarrow \frac{\vec{V}}{||\vec{V}||}$
  for each *coordinate* k
  define sets of indices

$$\mathcal{R}_s^k = \{i : r_s^k \leq |x_i^k - V^k u_i - \delta^k| < r_{s+1}^k\}, s = 0...p$$

  end for
  for each *data point* i and *coordinate* k
  find all $s_{i,k}$ *such that* $i \in \mathcal{R}_{s_{i,k}}^k$
  if *all* $a_{s_{i,k}}^k = 0$ ***then*** $u_i' \leftarrow 0$ **else**

$$u_i' \leftarrow \frac{\sum_k a_{s_{i,k}}^k V^k (x_i^k - \delta^k)}{\sum_k a_{s_{i,k}}^k (V^k)^2}$$

  end for
  for each *coordinate* k

$$V^k \leftarrow \frac{\sum_s a_s^k \sum_{i \in \mathcal{R}_s^k} (x_i^k - \delta^k) u_i}{\sum_s a_s^k \sum_{i \in \mathcal{R}_s^k} (u_i)^2}$$

  end for
  for each i :
  $u_i \leftarrow u_i'$
  end for
  goto *repeat till convergence*

data vectors $\vec{x}_i$ many times and the deepest in PQSQ approximation error (1) minimum should be selected.

2.5. PQSQ-based Principal Graphs and Manifolds

2.6. Convergence of the piece-wise quadratic potentials and the cone of minorant functions

In order to prove the general convergence of the data approximation algorithms based on using PQSQ potentials, let us consider even more general case where a potential can be constructed from a set of functions $\{q_i(x)\}$ with the only two requirements: 1) that each $q_i(x)$ has a (local) minimum; 2) that the whole set of all possible $q_i(x)$s forms a cone. In this case, instead of the operational definition (2) it is convenient to define the potential $u(x)$ as a minorant function for a set of functions as follows. For convenience, in this section, x will notify a vector $\vec{x} \in R^m$.

Let us consider *a generating cone of functions* Q. We remind that the definition of a cone implies that for any $q(x) \in Q, p(x) \in Q$, we have $\alpha q(x) + \beta p(x) \in Q$, where $\alpha \geq 0, \beta \geq 0$.

For any finite set of functions $q_1(x) \in Q, q_2(x) \in Q, ..., q_s(x) \in Q$, we define a minorant function (Figure 2):

$$u_{q_1,q_2,...,q_s}(x) = \min(q_1(x), q_2(x), ..., q_s(x)). \tag{10}$$

It is convinient to introduce a multiindex $I_{q_1,q_2,...,q_s}(x)$ indicating which particular function(s) q_i corresponds to the value of $u(x)$, i.e.

$$I_{q_1,q_2,...,q_s}(x) = \{i | u_{q_1,q_2,...,q_s}(x) = q_i(x)\}. \tag{11}$$

For a cone Q let us define a set of all possible minorant functions $\mathbb{M}(Q)$

$$\mathbb{M}(Q) = \{u_{q_{i_1},q_{i_2},...,q_{i_n}} | q_{i_1} \in Q, q_{i_2} \in Q, q_{i_n} \in Q, n = 1, 2, 3, \dots\}. \tag{12}$$

Proposition 1. $\mathbb{M}(Q)$ *is a cone.*

Proof For any two minorant functions $u_{q_{i_1},q_{i_2},...,q_{i_k}}, u_{q_{j_1},q_{j_2},...,q_{j_s}} \in \mathbb{M}(Q)$ we have

$$\alpha u_{q_{i_1},q_{i_2},...,q_{i_k}} + \beta u_{q_{j_1},q_{j_2},...,q_{j_s}} = u_{\{\alpha q_{i_p} + \beta q_{j_r}\}} \in \mathbb{M}(Q), p = 1, \dots, k, r = 1, \dots, s,$$

where $\{\alpha q_{i_p} + \beta q_{j_r}\}$ is a set of all possible linear combinations of functions from $\{q_{i_1}, q_{i_2}, ..., q_{i_k}\}$ and $\{q_{j_1}, q_{j_2}, ..., q_{j_s}\}$.

Proposition 2. *Any restriction of* $\mathbb{M}(Q)$ *onto a linear manifold* L *is a cone.*

Proof Let us denote $q(x)|_L$ a restriction of $q(x)$ function onto L, i.e. $q(x)|_L = \{q(x)|x \in L\}$. $q(x)|_L$ is a part of Q. Set of all $q(x)|_L$ forms a restriction $Q|_L$ of Q onto L. $Q|_L$ is a cone, hence, $\mathbb{M}(Q)|_L = \mathbb{M}(Q|_L)$ is a cone (Proposition 1).

Definition *Splitting algorithm* minimizing $u_{q_1,q_2,...,q_n}(x)$ is defined as follows.

Algorithm 3 Finding local minimum of minorant function $u_{q_1,q_2,...,q_n}(x)$

procedure MINIMIZING MINORANT FUNCTION
    initialize $x \leftarrow x_0$
repeat utill stopping criterion has been met:
    compute multiindex $I_{q_1,q_2,...,q_s}(x)$
    for all $i \in I_{q_1,q_2,...,q_s}(x)$
    $x_i = \arg\min\, q_i(x)$
    end for
    select optimal $x_i, x_{opt} \leftarrow \arg\min_{x_i} u(x_i)$
    $x \leftarrow x_{opt}$
    stopping criteria: check if the multiindex $I_{q_1,q_2,...,q_s}(x)$ does not change compared to the previous iteration
end repeat:

Theorem 2.1. *Splitting algorithm (Algorithm 3) for minimizing* $u_{q_1,q_2,...,q_n}(x)$ *converges in a finite number of steps.*

Proof Since the set of functions $\{q_1, q_2, ..., q_n\}$ is finite then we only have to show that at each step the value of the function $u_{q_1,q_2,...,q_n}(x)$ can not increase. For any x and the value $x' = \arg\min q_i(x)$ for $i \in I_{q_1,q_2,...,q_s}(x)$ we can have only two cases:

(1) Either $I_{q_1,q_2,...,q_s}(x) = I_{q_1,q_2,...,q_s}(x')$ (convergence, and in this case $q_{i'}(x') = q_i(x)$ for any $i' \in I_{q_1,q_2,...,q_s}(x')$);

(2) Or $u_{q_1,q_2,...,q_n}(x) < u_{q_1,q_2,...,q_n}(x')$ since, accordingly to the definition (10), $q_{i'}(x') < q_i(x)$, for any $i' \in I_{q_1,q_2,...,q_s}(x'), i \in I_{q_1,q_2,...,q_s}(x)$ (see Figure 2).

Note that in Algorithm 3 we do not specify exactly the way to find the local minimum of $q_i(x)$. To be practical, the cone Q should contain only functions for which finding a local minimum is fast and explicit. Evident candidates for this role are positively defined quadratic functionals $q(x) = q_0 + (\vec{q_1}, x) + (x, \mathbb{Q}_2 x)$, where $\mathbb{Q}_2$ is a positively defined symmetric matrix. Any minorant function (10) constructed from positively defined quadratic functions will automatically provide subquadratic growth, since the minorant can not grow faster than any of the quadratic forms by which it is defined.

Operational definition of PQSQ given above (2), corresponds to a particular form of the quadratic functional, with $\mathbb{Q}_2$ being diagonal matrix. This choice corresponds to coordinate-wise definition of data approximation error function (1) that are particularly simple to minimize. This circumstance is used in Algorithms 1,2.

Let us finally clarify how the Algorithm 3 serves a more abstract version of the Algorithms 1,2. For example, the "variance" function $m(\vec{x}) = \frac{1}{N}\sum_j u(\vec{x}_j - \vec{x})$ to be minimized in Algorithm 1 uses the generating functions in the form $Q = \{b^k_{ji} + \sum_k a^k_{ji}(x^k - x^k_j)^2\}$, where i is the index of the interval in (2). Hence, $m(x)$ is a minorant function, belonging to the cone $\mathbb{M}(Q)$, and must converge (to a local minimum) in a finite number of steps accordingly to Theorem 2.1.

3. Numerical examples

3.1. Practical choices of parameters

The main parameters of PQSQ are (a) majorating function $f(x)$ and (b) decomposition of each coordinate range into $p+1$ non-overlapping intervals. Depending on these parameters, various approximation error properties can be exploited, including robustness to outlier data points.

When defining the intervals $r_j, j = 1 \dots p$, it is desirable to achieve a small difference between $u(\Delta x) - u(\Delta x)$ for expected argument values Δx (differences between an estimator and the data points), and choose the suitable value of the potential trimming threshold r_p in order to achieve the desired robustness properties. If no trimming is needed, then r_p should be made larger than the maximum expected difference between coordinate values.

In our numerical experiments we used the following definition of intervals. For any data coordinate k, we define a characteristic difference D^k, for example

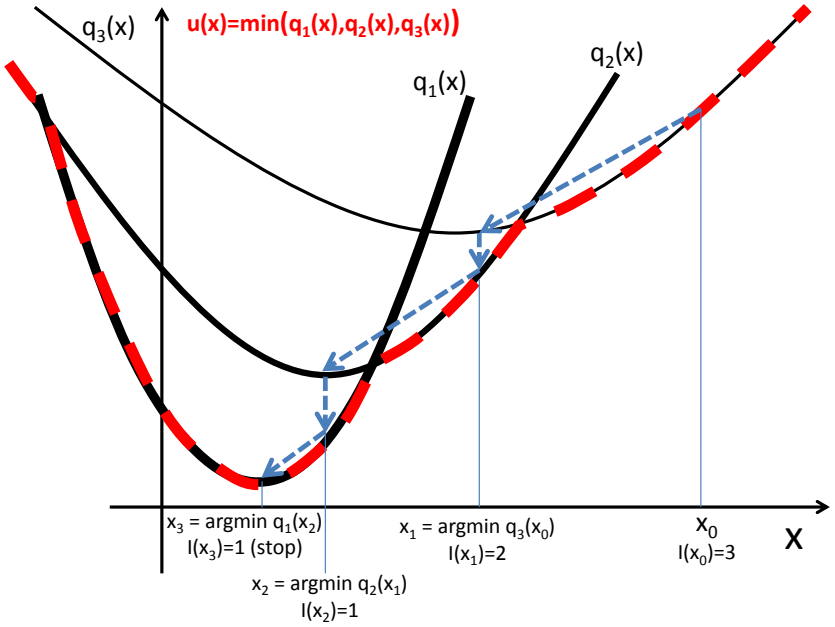

Figure 2: Optimization of a one-dimensional minorant function $u(x)$, defined by three functions $q_1(x), q_2(x), q_3(x)$ each of which has a global minimum. Each optimization step consists in determining which $q_{I(x)}(x) = u(x)$ and making a step into the local minimum of $q_{I(x)}$.

$$D^k = \alpha_{scale}(max_i(x_i^k) - min_i(x_i^k)), \tag{13}$$

where α_{scale} is a scaling parameter, which can be put at 1 (in this case, the approximating potential will not be trimmed). In case of existence of outliers, for defining D^k, instead of amplitude one can use other measures such as the median absolute deviation (MAD):

$$D^k = \alpha_{scale} median_i(x_i^k - median(\{x_i^k\})); \tag{14}$$

in this case, the scaling parameter should be made larger, i.e. $\alpha_{scale} = 10$, if no trimming is needed.

After defining D^k we use the following definition of intervals:

$$r_j^k = D^k \frac{j^2}{p^2}, j = 0 \dots p. \tag{15}$$

More sophisticated approaches are also possible to apply such as, given

the number of intervals p and the majorating function $f(x)$, choose $r_j, j = 1 \dots p$ in order to minimize the integral difference

$$\int_0^{r_p} (f(x) - u(x))dx \to \min .$$

In further examples, we use (13) and (15) to define intervals in (2).

3.2. Implementation

At https://github.com/auranic/PQSQ-DataApproximators we provide implementation of PQSQ approximators (in particular, PCA) in Matlab. We also provide Java implementation of PQSQ-based approximators (PCA, principal graphs) as a part of *vdaoengine* library at https://github.com/auranic/VDAOEngine. The code is accompanied by examples of application.

3.3. Computation performance

Comparison PQSQ-based PCA with standard quadratic metrics algorithm for computing SVD.

3.4. Robust principal components, approximating $L1$-based PCA

Comparison of computation time and precision PQSQ-based PCA for $u(x) = |x|$ with $L1$-based PCA (pcaL1 R implementation by Brooks et al.). See http://www.optimization-online.org/DB_FILE/2012/04/3436.pdf for possible benchmarking.

4. Conclusion

In this paper we propose a method of constructing the standard data approximators (mean value, k-means clustering, principal components, principal graphs) for arbitrary non-quadratic approximation error with subquadratic growth by using a piecewise-quadratic energy functional (PQSQ potential). These approximators can be computed by applying quasi-quadratic optimization procedures, which are simple adaptations of the previously described standard and computationally efficient algorithms.

The suggested methodology have several advantages:

(a) *Scalability*: the algorithms are computationally efficient and can be applied to large data sets containing millions of numerical values.

(b) *Flexibility*: the algorithms can be adapted to any type of data metrics with subquadratic growth, even if the metrics can not be expressed in explicit form. Idea of adaptive metrics [6, 7].

(c) *Built-in robustness*: choice of intervals in PQSQ can be done in the way to achieve a trimmed version of the standard data approximators, when points distant from the approximator do not affect to the energy minimization during the current optimization step.

(d) *Guaranteed convergence*: the suggested algorithms converge to local or global minimum just as the corresponding predecessor algorithms based on quadratic optimization and expectation/minimization-based splitting approach.

One of the application of the suggested methodology is approximating the popular in data mining $L1$ metrics. Does it provide (more) sparsity also compared to L2? We show by numerical simulations, that PQSQ-based approximators...

PSQS potential can be evidently applied in the task of regression, replacing the classical Least Squares or L1-based Least Absolute Deviation methods.

References

[1] A. Gorban, B. Kegl, D. Wunsch, A. Zinovyev (Eds.), Principal Manifolds for Data Visualisation and Dimension Reduction, LNCSE 58, Springer, 2008.

[2] A. N. Gorban, A. Zinovyev, Principal graphs and manifolds, In Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, eds. Olivas E.S., Guererro J.D.M., Sober M.M., Benedito J.R.M., Lopes A.J.S. (2009).

[3] C. Lu, Z. Lin, S. Yan, Smoothed low rank and sparse matrix recovery by iteratively reweighted least squares minimization., IEEE Trans Image Process 24 (2015) 646–654.

[4] J. Brooks, J. Dulá, E. Boone, A pure l1-norm principal component analysis., Comput Stat Data Anal 61 (2013) 83–98.

[5] K. Pearson, On lines and planes of closest fit to systems of points in space, Philos. Mag. 2 (1901) 559–572.

[6] L. Yang, R. Jin, Distance metric learning: A comprehensive survey, Michigan State Universiy 2 (2006).

[7] L. Wu, R. Jin, S. C. Hoi, J. Zhu, N. Yu, Learning bregman distance functions and its application for semi-supervised clustering, in: Advances in neural information processing systems, pp. 2089–2097.