Data analysis with arbitrary error measures approximated by piece-wise quadratic PQSQ functions

1st Alexander N. Gorban
Department of Mathematics
University of Leicester
Leiceter, UK
Lobachevsky University
Nizhni Novgorod, Russia
a.n.gorban@leicester.ac.uk

2nd Evgeny M. Mirkes
Department of Mathematics
University of Leicester
Leiceter, UK
Lobachevsky University
Nizhni Novgorod, Russia
em322@le.ac.uk

3rd Andrei Zinovyev
Computational Systems Biology of Cancer
Institut Curie, PSL Research University,
Mines ParisTech, INSERM, U900
Paris, France
Lobachevsky University
Nizhni Novgorod, Russia
andrei.zinovyev@curie.fr

***Abstract*—Defining an error function (a measure of deviation of a model prediction from the data) is a critical step in any optimization-based data analysis method, including regression, clustering and dimension reduction. Usual quadratic error function in case of real-life high-dimensional and noisy data suffers from non-robustness to presence of outliers. Therefore, using non-quadratic error functions in data analysis and machine learning (such as L1 norm-based) is an active field of modern research but the majority of methods suggested are either slow or imprecise (use arbitrary heuristics). We suggest a flexible and highly performant approach to generalize most of existing data analysis methods to an arbitrary error function of subquadratic growth. For this purpose, we exploit PQSQ functions (piece-wise quadratic of subquadratic growth), which can be minimized by a simple and fast splitting-based iterative algorithm. The theoretical basis of the PQSQ approach is an application of min-plus (idempotent) algebra to data approximation. We introduce the general idea of the approach and illustrate it on four standard tools of machine learning: simple regression, regularized regression, k-mean clustering and principal component analysis. In all cases, PQSQ-based methods achieve better robustness with respect to the presence of strong noise in the data compared to the standard methods.**

***Index Terms*—piece-wise quadratic error, k-means, principal component analysis, regression**

I. Introduction

Artificial intelligence methods based on machine learning are able to deal with large amounts of data converting it to predictive statistical models. One of the most essential ingredient of any machine learning method is the definition of *error function*, i.e., the measure of deviation of the model from the data. Most of the existing machine learning algorithms are based on minimizing the mean squared error, which can be explained by tractable properties of normal distribution and existence of computationally efficient methods for quadratic optimization. However, most of the real-life datasets are characterized by strong noise, long-tailed distributions, presence of contaminating factors, large dimensions. Using quadratic error can be compromised by all these circumstances: therefore, a lot of practical and theoretical efforts have been made in order to exploit the properties of non-quadratic error functions which can be more appropriate in certain contexts. For example, methods of regularized regression such as lasso and elastic net based on the properties of L1-metric [1], [2] found numerous applications in bioinformatics [3], and L1 norm-based methods of dimension reduction are used in automated image analysis [4]. Not surprisingly, these approaches comes with drastically increased computational cost, for example, connected with applying linear programming optimization techniques which are substantially more expensive compared to mean squared error-based methods.

In practical applications of machine learning, it would be very attractive to be able to deal with *arbitrary error functions*, including those based on L1 or fractional norm, in a computationally efficient and scalable way.

Recently, we have suggested a universal framework able to deal with a large family of error functions [5]. We exploited the fact that finding a minimum of a piece-wise quadratic function, or, in other words, a function which is the *minorant of a set of quadratic functions*, is not much more computationally costly as minimizing the standard quadratic error. Therefore, if a given arbitrary error (such as L1-based or fractional norm-based) can be approximated by a piece-wise quadratic function, this should lead to efficient and simple optimization algorithms. We introduced a rich family of piece-wise quadratic error functions of subquadratic growth (PQSQ-functions), and proved convergence of a simple iterative algorithm in the most general case.

In this paper we report on further development of the PQSQ-based approach to machine learning. We start with reviewing the basic notions of PQSQ framework. We extend the previously described coordinate-wise framework with its rotation-invariant version, which might be advantageous in practical applications (so called PQSQR error functions). We

The project is supported by the Ministry of education and science of Russia (Project No. 14.Y26.31.0022) and by Big Data PSL Research University project "PSL Institute for Data Science" (2016–2018).

show how the PQSQ formalism is applied in case of four classical data analysis methods: regularized regression and Principal Component Analysis (PCA) (as it was described before in [5]), multivariate regression and k-means clustering (new in this paper). We give few synthetic examples of how introducing PQSQ-based error function leads to improving method robustness to the presence of noise and artefacts.

II. Basic notions of PQSQ formalism

A. PQSQ function: minorant of a set of quadratic functions

Let us consider a finite set of functions $Q = \{q_k(x)\}, k = 1...s$ and a function $f(x)$ which grows not faster than any of them. Let us approximate $f(x)$ by the minorant function of Q

$$f(x) \approx u_{q_1,q_2,...,q_s}(x) = \min\{q_1(x), q_2(x), ..., q_s(x)\}. \quad (1)$$

This is a definition of the minorant function $u_{q_1,q_2,...,q_s}(x)$ in the general case. A special case important for applications is when Q is a set of centered parabolas $Q = \{q_k(x) = a_k x^2 + b_k\}, k = 1...s$. In this case we will call $u(x)$ "PQSQ function" (piece-wise quadratic function of subquadratic growth). In this case, in order to achieve $u(x) \approx f(x)$ in the interval $[0; R]$, we need to define parameters $a_k, b_k, k = 1...s$. 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 $r_0 = 0, r_{p+1} = \infty$. Then $u(x)$ is a piece-wise quadratic function (see Figure 1A):

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

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

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

where $f(x)$ is a function to be approximated (imitated) by $u(x)$ from below. 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 $L1$ norm. Note that accordingly to (3,4), $b_0 = 0$, $b_p = f(r_p)$. Also, in the case of $a_p = 0$, the choice of r_p can naturally create a "trimmed" version of the function $u(x)$ such that it becomes a constant at infinity.

B. Minimizing minorant function in the general case

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

$$I_{q_1,q_2,...,q_s}(x) = \{i | u_{q_1,q_2,...,q_s}(x) = q_i(x)\}. \quad (5)$$

Using this notation, it is easy to explain how a minorant function (1) can be rapidly minimized (see Figure 1B). For this purpose, it is sufficient to iterate, until the index $I_{q_1,q_2,...,q_s}(x)$ does not change, the following operations

1) For a given x, find $I_{q_1,q_2,...,q_s}(x)$,

2) For each $i \in I_{q_1,q_2,...,q_s}(x)$ identify (local) minimum of q_i, denoted $\min q_i$

3) Find $j \in I_{q_1,q_2,...,q_s}(x)$ such that $\min q_j = \min_{i \in I_{q_1,q_2,...,q_s}(x)} \min q_i$,

4) If there are several such js then select the smallest j (the first in the list),

5) $x \leftarrow \arg\min_x q_j(x)$.

If each $q_i(x)$ is a quadratic function, its (global) minimum can be found by an explicit linear formula. Finding $I(x)$ requires determining a set of minimum value(s) between s numbers. This simplicity makes minimization of a PQSQ function very fast.

III. Linear PQSQ regression

Let us consider a regression problem with a set of vectors $\{x_1, \ldots, x_n\}$ and the corresponding responses $y = (y_1, \ldots, y_n)$. Here the subscript index indicates the sample number. Standard linear regression equation for evaluating $\hat{y}$ of the response value for the given input data vector x is

$$\hat{y} = \beta_0 + x^T \beta, \quad (6)$$

where β is a vector of regression coefficients and β_0 is an intercept (a single value). In the classical Ordinary Least Squares (OLS) formulation, we minimize the sum of squared regression residuals:

$$F_{OLS} = \sum_{i=1}^{n} (y_i - \beta_0 - x_i^T \beta)^2 \rightarrow min. \quad (7)$$

In order to improve robustness to the presence of outliers, Least Absolute Deviation (LAD) [6] has been suggested

$$F_{LAD} = \sum_{i=1}^{n} |y_i - \beta_0 - x_i^T \beta| \rightarrow min. \quad (8)$$

However, LAD is computationally expensive and the algorithm can suffer from computational instabilities.

Here we introduce PQSQ regression, which is the method of regression coefficient estimation which uses PQSQ function instead of (7):

$$F_{PQSQ} = \sum_{i=1}^{n} u(y_i - \beta_0 - x_i^T \beta) \rightarrow min, \quad (9)$$

where $u(x)$ is a PQSQ function. Various functions $f(x)$ allow imitating different norms or pseudo-norms. For example, $f(x) = |x|$ can imitate LAD and usage of $f(x) = x^2$ is equivalent to OLS (with an option of trimming).

For the ith observation, let us determine to which interval from (2) the regression residuals belong, and denote them as s_i:

$$r_{s_i - 1} \leq \left| y_i - \beta_0 - x_i^T \beta \right| \leq r_{s_i}. \quad (10)$$

Then, the minimization problem (9) is reduced to solving a system of linear equations:

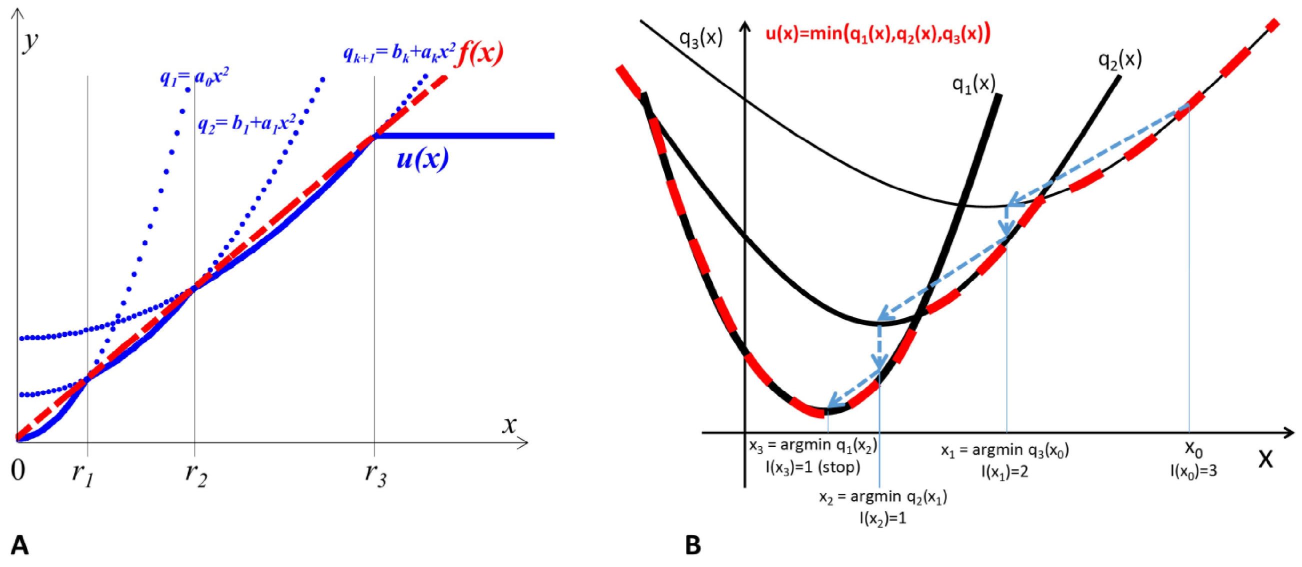


Fig. 1. A) Trimmed piecewise quadratic error of subquadratic growth $u(x)$ (solid blue line) defined for the function $f(x)$ (red dashed line) and several thresholds r_k. Dotted lines show the parabolas which fragments are used to construct $u(x)$. The last parabola is flat ($a_p = 0$) which corresponds to trimmed error function. B) 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 local minimum. Each optimization step consists in determining which $q_{I(x)}(x) = u(x)$ and making a step into the local minimum of $q_{I(x)}$.

$$\beta_0 \sum_{i=1}^{n} a_{s_i} + \sum_{j=1}^{m} \beta_j \sum_{i=1}^{n} a_{s_i} x_i^j = \sum_{i=1}^{n} a_{s_i} y_i \tag{11}$$

$$\beta_0 \sum_{i=1}^{n} a_{s_i} x_i^k + \sum_{j=1}^{m} \beta_j \sum_{i=1}^{n} a_{s_i} x_i^j x_i^k = \sum_{i=1}^{n} a_{s_i} y_i x_i^k \tag{12}$$

The systems of equations (11), (12) can be written in a matrix form as

$$Z^T Q Z \beta' = Z^T Q y, \tag{13}$$

where Z is an extended data matrix $Z = \{\vec{1}, x_1, \ldots, x_n\}$ (i.e., it contains an additional row of 1s), vector β' contains the intercept value β_0 as the first element and other elements of $\vec{\beta}$ starting from the second element. Q is diagonal matrix

$$\begin{matrix} a_{q_1} & 0 & \ldots & 0 \\ 0 & a_{q_2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{q_n} \end{matrix}$$

Formula (13) is equivalent to Weighted Least Square (WLS) with weights defined by PQSQ coefficients.

The problem (9) can be solved iteratively. One starts with some initial guess for β. Then one identifies the vector of indices $\{s_i\}$ and solve (13). This process is iterated until the vector of indices $\{s_i\}$ for two consecutive operations is the same. We stress here that this stopping criterion does not require to specify any parameter for defining the usual "tolerance level".

IV. Linear regression regularized by PQSQ function

One of the major application of non-Euclidean norm properties in machine learning is using non-quadratic terms for penalizing large absolute values of regression coefficients [1], [2]. Depending on the chosen penalization term, it is possible to achieve various effects such as sparsity or grouping coefficients for redundant variables. In a general form, regularized regression solves the following optimization problem

$$\frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{m} \beta^k x_i^k \right)^2 + \lambda f(\beta) \to \min, \tag{14}$$

where N is the number of observations, m is the number of independent variables in the matrix $\{x_i^k\}$, $\{y_i\}$ are dependent variables, λ is an internal parameter controlling the strength of regularization (penalty on the amplitude of regression coefficients β), and $f(z)$ is the regularizer function, which is $f(z) = \|z\|_{L2}^2$ for Tikhonov regularization also known as ridge regression, $f(z) = \|z\|_{L1}$ for lasso and $f(z) = (1-\alpha)\|z\|_{L2}^2 + \alpha\|z\|_{L1}$ for the elastic net methods correspondingly.

Here we suggest to imitate $f(x)$ with a $PQSQ$ function, i.e. instead of (14) to solve the following problem

$$\frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{m} \beta^k x_i^k \right)^2 + \lambda \sum_{k=1}^{m} u(\beta^k) \to \min, \tag{15}$$

where $u(x)$ is a PQSQ function imitating *arbitrary* subquadratic regression regularization penalty.

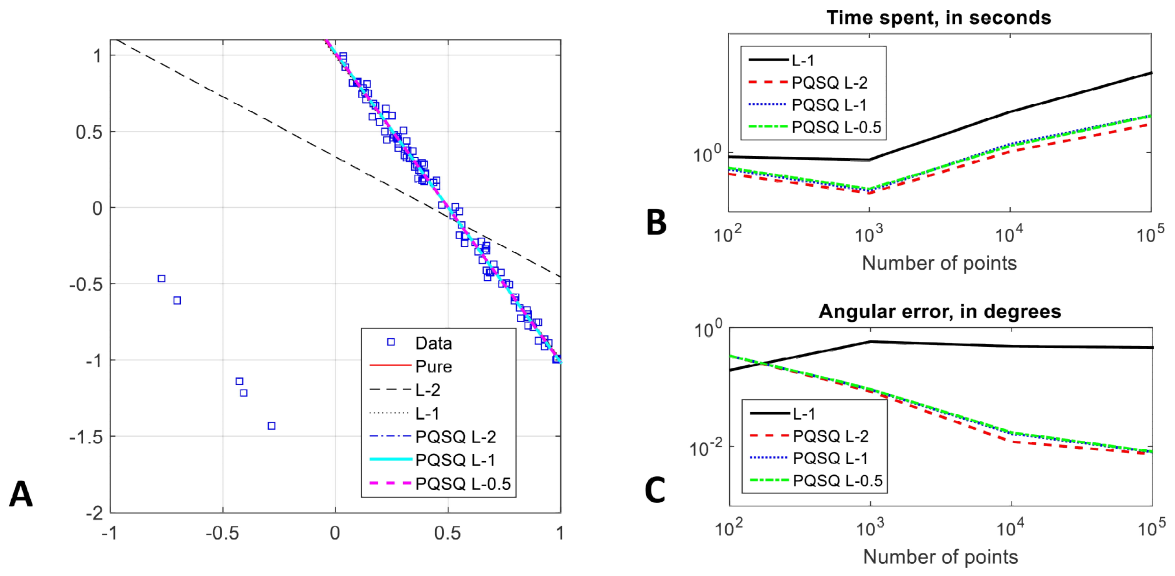


Fig. 2. Comparison of PQSQ-based regressions with the standard OLS (Ordinary Least Squares), denoted "L-2", and LAD (Least Absolute Deviation), denoted "L-1", regressions. A) The example contains a certain number of points located close to a regression line and accompanied by 5% of clear outliers. B) Estimating the computational performance of the algorithms. C) Estimating the angular accuracy of the regression estimation. Three PQSQ functions used to produce these examples (PQSQ L-2, PQSQ L-1 and PQSQ L-0.5) correspond to three choices of the error function approximated by the minorant function from below and give similar results. In all plots, 'Pure' means true regression line. For the details of how the example is constructed, read the 'Testing PQSQ regression' section of the paper.

Solving (15) is equivalent to iteratively solving a system of linear equations

$$\frac{1}{N}\sum_{k=1}^{m}\beta^k\sum_{i=1}^{N}x_i^k x_i^j+\lambda a_{I(\beta^j)}\beta^j = \sum_{i=1}^{N}y_i x_i^j,\ j=1,\ldots,m, \tag{16}$$

where $a_{I(\beta^j)}$ constant is computed accordingly to the definition of $u(x)$ function (see (3)) and I index is defined from $r_I \leq \beta^j < r_{I+1}$, given the estimation of β^k regression coefficients at the current iteration. In practice, iterating (16) converges in a few iterations, therefore, the algorithm can work very fast and outperform the widely used least angle regression and coordinate descend method for solving (14) in case of L_1 penalties.

Examples of application of PQSQ-regularized regression to real-life data was shown in our previous work [5].

V. K-means clustering with PQSQ function

Fréchet mean vector $\bar{x}_F$ for a set of vectors $X=\{x_i\}$, $i=1,\ldots,N$, in R^m and an error function $f(x)$ can be defined as a vector minimizing the sum of deviations from $\bar{x}_F$ to all points in X.

$$\sum_i\sum_k f(x_i^k-\bar{x}_F^k)\rightarrow\min. \tag{17}$$

For Euclidean metric L_2 ($f(x)=x^2$) it is the usual arithmetic mean. For L_1 metric ($f(x)=|x|$), (17) 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. This equation can have a non-unique solution in case of even number of points or when some data point coordinates coincide. More generally, any non-quadradic $f(x)$ (17) might lead to a non-unique definition of the Fréchet mean. For PQSQ functional (2), the mean value is defined through solving the optimization problem (17) by a simple iterative algorithm (Algorithm 1). In general case, the suggested algorithm converges to a local minimum which depends on the initial point.

Algorithm 1 Computing PQSQ mean value

```
1:  define intervals $r_s^k, s=0,\ldots,p, k=1,\ldots,m$
2:  compute coefficients $a_s^k$
3:  initialize $\bar{x}_{PQSQ}$
    eg., by arithmetic mean
4:  repeat till convergence of $\bar{x}_{PQSQ}$:
5:  for each coordinate $k$
6:  define sets of indices
        $\mathcal{R}_s^k=\{i: r_s^k\leq|x_i^k-\bar{x}_{PQSQ}^k|<r_{s+1}^k\}, s=0,\ldots,p$
7:  compute new approximation for $\bar{x}_{PQSQ}$:
8:  $\bar{x}_{PQSQ}^k \leftarrow \frac{\sum_{s=1,\ldots,p}a_s^k\sum_{i\in\mathcal{R}_s^k}x_i^k}{\sum_{s=1,\ldots,p}a_s^k|\mathcal{R}_s^k|}$
9:  end for
10: goto repeat till convergence
```

Based on the PQSQ approximation measure and the algorithm for computing the PQSQ mean value (Algorithm 1), we can construct the PQSQ-based k-means clustering procedure in the usual way (Algorithm 2).

Algorithm 2 PQSQ k-means clustering

```
1: define $k$
2: initialize centroids $c_i, i = 1...k$
3: repeat till convergence
4: partition the data into $k$ disjoint sets $K_i, i = 1...k$,
     such that if $\vec{x} \in K_i$ then $c_i = argmin_j \sum_l u(x^l - c_j^l)$
5: for each set $K_i, i = 1...k$ compute PQSQ mean value
     and assign $c_i = PQSQ\_Mean\_value(K_i)$
6: goto repeat till convergence
```

VI. Principal Component Analysis with PQSQ-based error function

Accordingly to the classical definition of the first principal component, it is a line best fit to the data set X [7]. Let us define a line in the parametric form $\vec{y} = \vec{V}\nu + \vec{\delta}$, where $\nu \in R^1$ is the parameter.

The first principal component is defined by vectors $\vec{V}, \vec{\delta}$ satisfying

$$\sum_i \sum_k u(x_i^k - V^k \nu_i - \delta^k) \to \min, \quad (18)$$

where

$$\nu_i = \arg\min_s \sum_k u(x_i^k - V^k s - \delta^k). \quad (19)$$

The usual 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 L_1-based principal components ($u(x) = |x|$) represents a much more challenging optimization problem [8]–[12].

Computing PCA based on PQSQ approximation error is only slightly more complicated than computing the standard L_2 PCA by SVD. A pseudo-code (Algorithm 3) has guaranteed convergence as it was proven in [5].

It is sufficient to define the first principal component: higher order components can be computed by the standard deflation approach. Figure 4 shows an example of how application of PQSQ-based error function improves the robustness properties of Principal Component Analysis.

VII. PQSQR error function: constructing universal rotationally invariant data approximators

The definition of the mean value (17) in the case of non-quadratic function $f(x)$ might be inconvenient in applications in two aspects: it can be non-rotationally invariant and it can be non-unique. From the other hand, using non-quadratic functions might drastically improve robustness of the clustering (as it can be seen in examples shown in this article).

Algorithm 3 Computing PQSQ PCA

```
1: define intervals $r_s^k, s = 0, \dots, p, k = 1, \dots, m$
2: compute coefficients $a_s^k$
3: $\vec{\delta} \leftarrow \bar{X}_{PQSQ}$
4: initialize $\vec{V}$ : eg., by $L_2$-based PC1
5: initialize $\{\nu_i\}$ : eg., by
     $\nu_i = \frac{\sum_k V^k (x_i^k - \delta^k)}{\sum_k (V^k)^2}$
6: repeat till convergence of $\vec{V}$:
7: normalize $\vec{V}$ : $\vec{V} \leftarrow \frac{\vec{V}}{\|\vec{V}\|}$
8: for each coordinate $k$
9: define sets of indices
     $\mathcal{R}_s^k = \{i : r_s^k \leq |x_i^k - V^k \nu_i - \delta^k| < r_{s+1}^k\}, s = 0, \dots, p$
10: end for
11: for each data point $i$ and coordinate $k$
12: find all $s_{i,k}$ such that $i \in \mathcal{R}_{s_{i,k}}^k$
13: if all $a_{s_{i,k}}^k = 0$ then $\nu_i' \leftarrow 0$ else
14: $\nu_i' \leftarrow \frac{\sum_k a_{s_{i,k}}^k V^k (x_i^k - \delta^k)}{\sum_k a_{s_{i,k}}^k (V^k)^2}$
15: end for
16: for each coordinate $k$
     $V^k \leftarrow \frac{\sum_s a_s^k \sum_{i \in \mathcal{R}_s^k} (x_i^k - \delta^k) \nu_i}{\sum_s a_s^k \sum_{i \in \mathcal{R}_s^k} (\nu_i)^2}$
17: end for
18: for each $i$ :
19: $\nu_i \leftarrow \nu_i'$
20: end for
21: goto repeat till convergence
```

In order to enjoy the good robustness properties but reducing the impact of inconvenient properties, any PQSQ-based data approximator can be reformulated such that the PQSQ error function would be defined not coordinate-wise, but to the usual Euclidean distance $||\vec{x} - \vec{y}||_2 = \sqrt{\sum_l (x^l - y^l)^2}$ (which is rotationally invariant). We will systematically denote such method formulations using "PQSQR" prefix.

Let us define the PQSQR error function (pseudo-distance between two data vectors $\vec{x}$ and $\vec{y}$) as

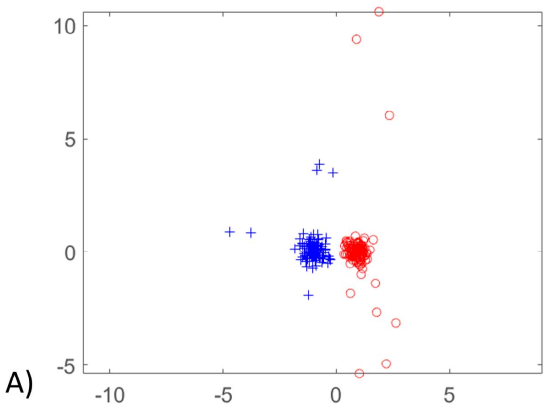
$$d_{PQSQR}(\vec{x}, \vec{y}) = u(||\vec{x} - \vec{y}||_2), \quad (20)$$

where $u(x)$ is a one-dimensional PQSQ function (1).

One can define the PQSQR-mean value $\bar{X}$ as a solution to the following optimization problem:

$$\sum_i u(||\vec{x}_i - \bar{X}||_2) \to \min. \quad (21)$$

Quite similarly, PQSQR projection on a line $\vec{y} = \vec{V}\nu + \vec{\delta}$, where $\nu \in R^1$, is a solution of the following problem:



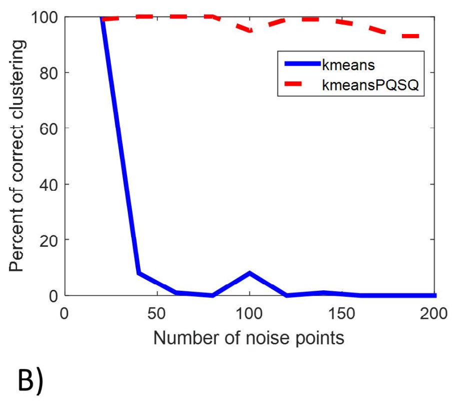


Fig. 3. Testing the PQSQ k-means against the standard k-means MATLAB implementation. A) Example with two (known) clusters accompanied by noisy points. An example of correct clustering produced by 2-means algorithm is shown by color. B) Ability of the algorithm to produce correct clustering *vs* number of noisy points. For the details of how the example is constructed, read the 'Testing PQSQ k-means' section of the paper.

$$\nu_i = \arg\min_s \sum_k u(||x_i^k - V^k s - \delta^k||_2), \qquad (22)$$

and the first PQSQR principal component is defined by vectors $\vec{V}, \vec{\delta}$ satisfying

$$\sum_i \sum_k u(||x_i^k - V^k \nu_i - \delta^k||_2) \to \min. \qquad (23)$$

Therefore, the generalizations of Algorithm 1, Algorithm 2 and Algorithm 3 for the PQSQR error function are straightforward. The exact pseudo-code will be published elsewhere. Quite similarly, it is straightforward to implement the linear regression with PQSQR-based regularization.

The definitions (21) and (22) do not guarantee uniqueness of the mean point by themselves. However, under certain assumptions (convexity of the $f(x)$ function and no trimming) the number of local minima should be smaller and their distribution should form more compact sets compared to coordinate-wise PQSQ function. The later statement requires more careful study.

It is also expected that PQSQR implementations of data approximation algorithms are faster than the corresponding PQSQ implementations, due to the smaller number of individual PQSQ-based function minimizations.

VIII. Synthetic examples showing robust properties of PQSQ-based error function

In order to benchmark the advantage of using PQSQ error function for the classical data analysis methods (regression, k-means, PCA), we created several simulated numerical examples.

A. Testing PQSQ regression

We considered how well PQSQ approach performs for estimating a well-defined regression line, in the situation when there exists 5% of clear outliers (Figure 2A). We compared the PQSQ-based regression with the function $f(x) = |x|$ against the standard OLS regression and an existing implementation of LAD regression from http://matlabdatamining.blogspot.fr/2007/10/l-1-linear-regression.html (L1-based, applying re-weightening [6]). The computational time shown in Figure 2B is measured using an ordinary laptop. Angular error (Figure 2C) means angular distance between the true regression line and the line estimated by an algorithm.

B. Testing PQSQ kmeans

We demonstrate the advantage of using PQSQ-based k-means clustering algorithm by constructing a simple example of data distribution consisting of a dense two-cluster component superimposed with a sparse contaminating component with relatively large variance whose co-variance does not coincide with the dense signal. We modeled two clusters as two 100-point samples S1 and S2 from normal distribution centered in points $[-1; 0]$ and $[1; 0]$ with isotropic variance with the standard deviation 0.1. The sparse noise distribution was modeled as a k-point sample from the product of two Laplace distributions of zero means with the standard deviations 2 along abscissa and 4 along ordinate (Figure 3A).

We study the ability of PQSQ k-means to produce correct separation of two clusters from the dense component, without knowing the cluster labels. For both standard k-means and PQSQ k-means (L1-like PQSQ error function), the clustering is applied 5 times from random centroid initializations, and then the optimal clustering is selected. In order to select the optimal clustering in the case of PQSQ k-means, we did the following test. For each point x_i we select the nearest centroid $c(x_i)$ as centroid with minimal PQSQ norm of vector $x_i - c(x_i)$. We select the clustering with minimal sum of PQSQ norm of vectors $x_i - c(x_i)$. In other words, for selecting the best clustering we use the PQSQ-based approximation error. In the case of standard k-means, we used the internal to MATLAB implementation clustering optimality criterion.

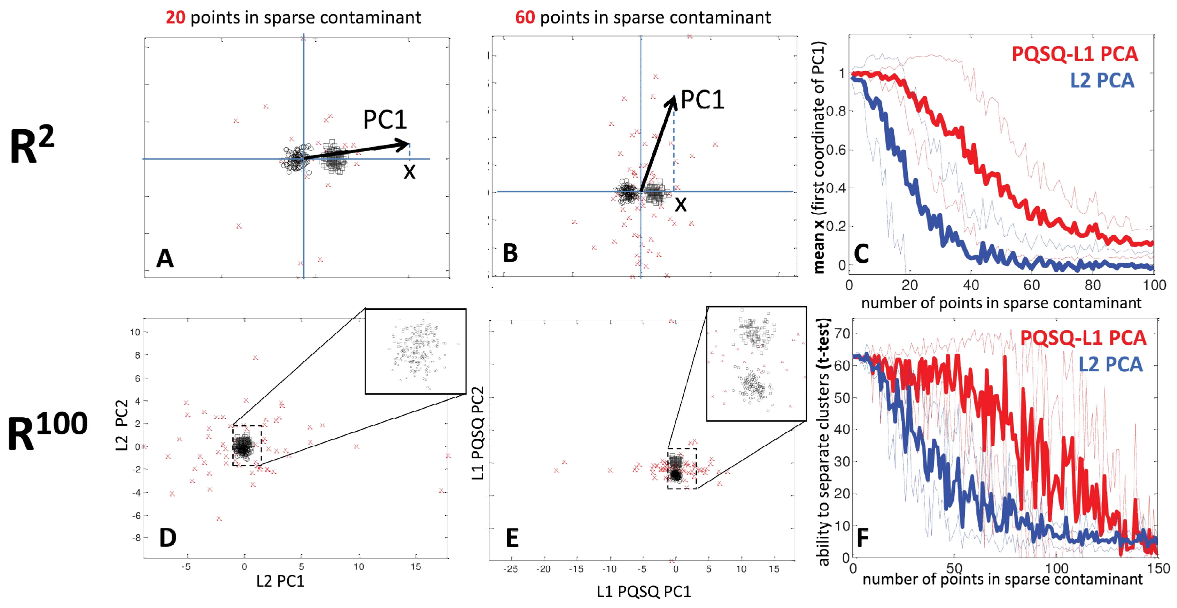


Fig. 4. Comparing L_2- and PQSQ PCA using example of two-cluster distribution (100 black circles and 100 squares) contaminated by sparse noise (red crosses). A) Dense two cluster distribution contaminated by sparse distribution (20 points) of large variance. In the presence of noise, the abscissa coordinate x of PC1 vector is slightly less than 1. B) Same as A) but in the case of strong contamination (60 points). The value of x is much smaller in this case. C) Average absolute value of the abscissa coordinate of PC1 $|x|$ (thick lines) shown with standard interval (thin lines) for 100 samples of k contaminating points. D) Projection of the data distribution on the first two principal components computed with the standard L_2 PCA algorithm. The number of contaminating points is 40. The cluster structure of the dense part of the distribution is completely hidden as shown in the zoom window. E) Same as in D) but computed with PQSQ L_1-based algorithm. The cluster structure is perfectly separable. F) The value of t-test computed based on the known cluster labels of the dense part of the distribution, in the projections onto the first two principal components of the global distribution. As in C), the mean values of 100 contamination samples together with confidence intervals are shown. For the details of how the example is constructed, read the 'Testing PQSQ PCA' section of the paper.

This procedure was then repeated 100 times, for which we recorded how many times the correct clustering was produced. For estimating 'correct' clustering, we took into account only the points from the dense data component. Clustering is considered as correct if at least 95% of points of the set S1 belong to one discovered cluster and at least 95% of points of set S2 belong to another discovered cluster (Figure 3B). Example shown in Figure 3 can be reproduced by launching benchmarkPQSQKmeans.m script in https://github.com/auranic/PQSQ-DataApproximators/tree/master/test_data/kmeans_test folder.

C. Testing PQSQ PCA

Using the same example data distribution as in the previous section, we study the ability of PQSQ PCA to withstand certain level of sparse contamination and compare it with the standard L_2-based PCA. In this example, without noise the first principal component coincides with the vector connecting the two cluster centers: hence, it perfectly separates them in the projected distribution. Noise interferes with the ability of the first principal component to separate the clusters to the degree when the first principal component starts to match the principal variance direction of the contaminating distribution (Figure 4A,B). In higher dimensions, not only the first but also the first two principal components are not able to distinguish two clusters, which can hide an important data structure when applying the standard data visualization tools.

The intervals for computing the PQSQ functional were defined by thresholds $R = \{0, 0.01, 0.1, 0.5, 1\}$ for each coordinate. Increasing the number of points in the contaminating distribution diminishes the average value of the abscissa coordinate of PC1, because the PC1 starts to be attracted by the contaminating distribution (Figure 4C). However, it is clear that on average PQSQ L_1-based PCA is able to withstand much larger amplitude of the contaminating signal (very robust up to 20-30 points, i.e. 10-20% of strong noise contamination) compared to the standard L_2-based PCA (which is robust to 2-3% of contamination).

In the second test we study the ability of the first two principal components to separate two clusters, in R^{100} (Figure 4D-F). As in the first test, we modeled two clusters as two 100-point samples from normal distribution centered in points $[-1, 0, \ldots, 0]$ and $[1, 0, \ldots, 0]$ with isotropic variance with the standard deviation 0.1 in all 100 dimensions. The sparse noise distribution is modeled as a k-point sample from the product of 100 Laplace distributions of zero means with the standard deviations 1 along each coordinate besides the third coordinate (standard deviation of noise is 2) and the forth coordinate (standard deviation of noise is 4). Therefore, the first two principal component in the absence of noise are attracted by the dimensions 1 and 2, while in the presence of strong noise they are be attracted by dimensions 3 and 4, hiding the cluster structure of the dense part of the distribution.

The definitions of the intervals were taken as in the first test. We measured the ability of the first two principal components to separate clusters by computing the t-test between the two clusters projected in the 2D-space spanned by the first principal components of the global distribution (Figure 4D-F). As one can see, the average ability of the first principal components to separate clusters is significantly stronger in the case of PQSQ L_1-based PCA which is able to separate robustly the clusters even in the presence of strong noise contamination (up to 80 noise points, i.e. 40% contamination).

IX. Implementation

We provide highly efficient MATLAB implementations of PQSQ-based data analysis methods. PQSQ regression is available at https://github.com/Mirkes/PQSQ-regression. PQSQ-regularized regression can be found at https://github.com/Mirkes/PQSQ-regularized-regression. The data approximation methods, including PQSQ and PQSQR k-means, PQSQ and PQSQR Principal Component Analysis can be found at https://github.com/auranic/PQSQ-DataApproximators. The usage of the PQSQ-based MATLAB functions is analogous to the usage of the corresponding standard functions in MATLAB. All the numerical tests in this article were made using MATLAB R2013a. The code of the tests is provided as a part of the package distributions.

X. Conclusion

In this paper we developed a new machine learning framework allowing one to deal with arbitrary error function of not-faster than quadratic growth, imitated by piece-wise quadratic function of subquadratic growth (PQSQ function).

We developed methods for constructing the standard data approximators (mean value, k-means clustering, regression, principal components) for arbitrary non-quadratic approximation error with subquadratic growth, PQSQ-based regression and regularized linear regression with arbitrary subquadratic penalty by using a piecewise-quadratic error functional (PQSQ-based error function). These problems could be solved by applying quasi-quadratic optimization procedures, which are organized as solutions of sequences of linear problems by standard and computationally efficient algorithms.

In addition, we suggested two formulations of the PQSQ-based optimization problems.

First one is coordinate-wise, where each coordinate is treated by an independent PQSQ function, which can be different for each coordinate. Coordinate-wise formulation is not invariant with respect to rotating the dataset in multi-dimensional space (orthonormal transformation), just as L1-based PCA. Second formulation is rotational-invariant which is achieved by applying the PQSQ-based error function to the standard Euclidean distance function (PQSQR). In the second case, the optimization result remains the same even if the dataset is orthonormally transformed.

The rotational invariant approach to PCA suggested in this article is conceptually close to [13]: however, PQSQR error functions can be applied to much more general case than only imitating L1-metric and they enjoy very fast algorithms for optimization. PQSQR approach to k-means can be considered as a significant generalization of both k-medoids clustering and trimmed k-means methods [14]. In case of PQSQR k-means it is possible to avoid introducing a hard trimming threshold: instead, one can gradually reduce the impact of distant points onto the definition of the k-means centroids.

Interestingly, the theory of PQSQ functions is based on the notion of the cone of minorant functions, and represents a natural approximation formalism based on the application of min-plus algebra [5].

PQSQ-based error function can be easily introduced literally in any machine learning method based on quadratic error optimization. This is expected to lead to the improvement in the computational cost/accuracy trade-off, since PQSQ-based machine learning methods achieve orders of magnitude faster computational performance than the corresponding state-of-the-art methods, having similar or better approximation accuracy.

References

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

[2] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[3] E. Barillot, L. Calzone, P. Hupe, J.-P. Vert, and A. Zinovyev, *Computational Systems Biology of Cancer*. Chapman & Hall, CRC Mathemtical and Computational Biology, 2012.

[4] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[5] A. N. Gorban, E. M. Mirkes, and A. Zinovyev, "Piece-wise quadratic approximations of arbitrary error functions for fast and robust machine learning," *Neural Networks*, vol. 84, pp. 28–38, 2016. [Online]. Available: https://doi.org/10.1016/j.neunet.2016.08.007

[6] D. P. O'Leary, "Robust regression computation using iteratively reweighted least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 466–480, 1990.

[7] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, vol. 2, no. 6, pp. 559–572, 1901.

[8] J. Brooks, J. Dulá, and E. Boone, "A pure l1-norm principal component analysis." *Comput Stat Data Anal*, vol. 61, pp. 83–98, May 2013. [Online]. Available: http://dx.doi.org/10.1016/j.csda.2012.11.007

[9] J. Brooks and S. Jot, "pcal1: An implementation in r of three methods for l1-norm principal component analysis," *Optimization Online preprint*, 2012.

[10] Y. W. Park and D. Klabjan, "Algorithms for l1-norm principal component analysis," 2014.

[11] Q. Ke and T. Kanade, "Robust l 1 norm factorization in the presence of outliers and missing data by alternative convex programming," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 739–746.

[12] N. Kwak, "Principal component analysis based on l1-norm maximization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 9, pp. 1672–1680, 2008.

[13] C. Ding, D. Zhou, X. He, and H. Zha, "R1-PCA: rotational invariant L1-norm principal component analysis for robust subspace factorization," *ICML*, pp. 281–288, 2006.

[14] J. Cuesta-Albertos, A. Gordaliza, C. Matrán *et al.*, "Trimmed k-means: An attempt to robustify quantizers," *The Annals of Statistics*, vol. 25, no. 2, pp. 553–576, 1997.