Algorithms for L1 Principal Component Analysis

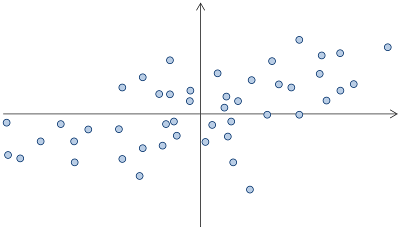
Diego Klabjan and Young Woong Park

Industrial Engineering and Management Sciences, Northwestern University

Sep 11, 2014

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.

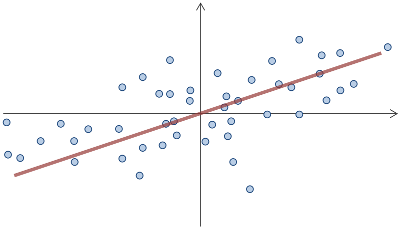


- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.

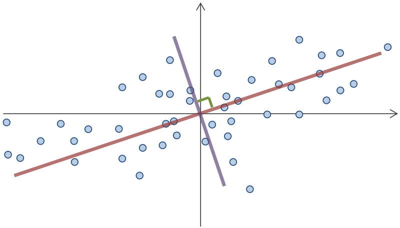


- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.

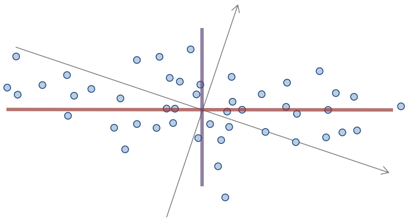


- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.

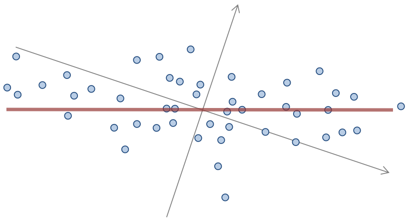


- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.

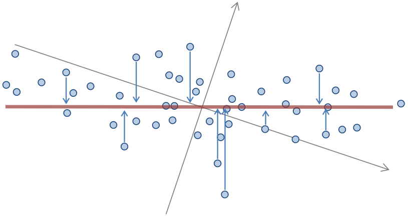


- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.

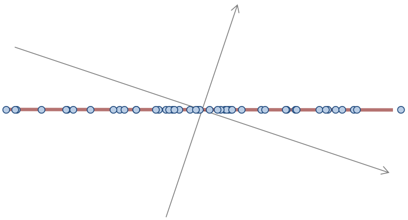


- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: Principal Component Analysis

- Principal Component Analysis (PCA) finds orthonormal vectors, which are a linear combination of the attributes of the data, that explain the variance structure of the data.



- Notations

  n: number of observations
  m: number of attributes
  p: number of principal components
  $I = \{1, \cdots, n\}$: index set of the observations
  $J = \{1, \cdots, m\}$: index set of the columns of the data set
  $P = \{1, \cdots, p\}$: index set of the principal components
  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}

Introduction: L2 PCA

- Minimization of reconstruction error

$$\min_{X \in \mathbb{R}^{m \times p}, X^\top X = I_p} \|A - AXX^\top\|_F^2. \tag{1}$$

- Maximization of deviation of projected data

$$\max_{X \in \mathbb{R}^{m \times p}, X^\top X = I_p} \|AX\|_F^2. \tag{2}$$

- (1) and (2) give the same solution
- An optimal solution can be obtained by
  - Eigenvalue decomposition (EVD) of $A^\top A$
  - Singular value decomposition (SVD) of A
- However, L_2 norms are known to be sensitive to outliers
  - L_1 PCA: use L_1 norm to build more robust PCs

Introduction: L1 PCA

- Minimization of reconstruction error

$$\min_{X \in \mathbb{R}^{m \times p}} \sum_{i \in I} \sum_{j \in J} |e_{ij}| \text{ s.t. } , X^\top X = I_p, E = A - AXX^\top. \tag{3}$$

- Maximization of deviation of projected data

$$\max_{X \in \mathbb{R}^{m \times p}} \sum_{i \in I} \sum_{k \in P} |y_{ik}| \text{ s.t. } , X^\top X = I_p, Y = AX. \tag{4}$$

- Unlike L_2 versions, solutions of (3) and (4) are different.
- There is no known exact approach to solving either (3) or (4).
- In this talk,
  1. We present the first Iteratively Reweighted Least Square (IRLS) algorithm for L1 PCA
  2. We present mathematical programming based algorithm together with use of SVD

Weighted Least Square Algorithm for the Minimization of Reconstruction Error

Weighted PCA for Error Minimization: Motivation (1)

- The only difference between L_1 and L_2 PCA is the objective function

$$\begin{array}{ll} \min & \sum_{i\in I}\sum_{j\in J}|e_{ij}| \\ \text{s.t.} & X^\top X = I_p \\ & E = A - AXX^\top \end{array} \qquad \begin{array}{ll} \min & \sum_{i\in I}\sum_{j\in J}e_{ij}^2 \\ \text{s.t.} & X^\top X = I_p \\ & E = A - AXX^\top \end{array}$$

- Optimal solution for L_2 PCA can be obtained by SVD of $A^\top A$.

- Weighted L_2 PCA (given weight $w_i > 0$ for each observation):

$$\begin{array}{ll} \min & \sum_{i\in I} w_i \sum_{j\in J}e_{ij}^2 \\ \text{s.t.} & X^\top X = I_p \\ & E = A - AXX^\top \end{array}$$

- Give smaller weight to observation with large error
  $\Rightarrow$ Reduce effect of outliers in the objective function of L1 PCA

- How can we solve the weighted L_2 PCA?

Weighted PCA for Error Minimization: Motivation (2)

- Weighted L_2 PCA (given weight $w_i > 0$ for each observation):

$$\begin{array}{ll} \min & \sum_{i\in I} w_i \sum_{j\in J} e_{ij}^2 \\ \text{s.t.} & X^\top X = I_p \\ & E = A - AXX^\top \end{array}$$

- Can solve the weighted L_2 PCA optimally by modifying data matrix A.
- Weighted matrix: $\bar{A}$, where $\bar{a}_{ij} = \sqrt{w_i} a_{ij}$
- Proposition Solving weighted L_2 PCA with A is equivalent to solving L_2 PCA with $\bar{A}$
- Use weighted L_2 PCA
  $\Rightarrow$ Optimize L_1 PCA
  Define proper (to reduce effect of observations with large error) weights
  $\Rightarrow$ Optimize L_1 PCA

Weighted Algorithm for Error Minimization: Weight Definition

- Given error matrix E^t obtained from iteration t, for iteration $t+1$,

$$w_i^{t+1} = \begin{cases} \frac{\sum_{j\in J} |e_{ij}^t|}{\sum_{j\in J} (e_{ij}^t)^2} & \text{if } \sum_{j\in J} (e_{ij}^t)^2 > 0 \\ M & \text{if } \sum_{j\in J} (e_{ij}^t)^2 = 0 \end{cases}$$

where M is a large constant.

- If we obtain optimal error matrix E^* for L_1 PCA in iteration t, then in iteration $t+1$, we have
  - $\sum_{i\in I} \sum_{j\in J} |e_{ij}^*| = \sum_{i\in I} w_i^{t+1} \sum_{j\in J} (e_{ij}^*)^2$
  - $\sum_{i\in I} w_i^{t+1} \sum_{j\in J} (e_{ij}^{t+1})^2 \leq \sum_{i\in I} \sum_{j\in J} |e_{ij}^*|$

- Want to decrease the effect of outliers $\Rightarrow$ smaller weights for outliers
  - Example 1: $e_1 = (3,3,3)$ and $e_2 = (5,1,1)$; $w_1 = \frac{9}{27}$ and $w_2 = \frac{7}{27}$; less weight for e_2
  - Example 2: $e_1 = (10,2,2)$ and $e_2 = (5,1,1)$; $w_1 = \frac{14}{108}$ and $w_2 = \frac{7}{27}$; less weight for e_1

Weighted Algorithm for Error Minimization: Overall Algorithm

- Notations

$W_t \in \mathbb{R}^{n \times n}$	diagonal matrix with $\sqrt{w_i^t}$'s on the diagonal
$A_t \in \mathbb{R}^{n \times m}$	adjusted data matrix, defined as $A_t = W_t A$
$X_t \in \mathbb{R}^{m \times p}$	the principal components matrix obtained by SVD of A_t

- Algorithm wPCA
  - Initialize parameters: $t \leftarrow 0$, $w_i^0 \leftarrow 2$, $w_i^1 \leftarrow 1$, $F^{best} \leftarrow \infty$, $X^{best} \leftarrow \emptyset$
  - **While** $\|w^t - w^{t-1}\| > \varepsilon$

    Set up A_t based on w_t
    $X_t \leftarrow PCA(A_t, p)$
    If $F(X_t) < F^{best}$ then $X^{best} \leftarrow X_t$, $F^{best} \leftarrow F(X_t)$
    Update weight for iteration $t+1$
    Increase t by 1

Weighted Algorithm for Error Minimization: Convergent weight and weighted matrix

- Convergent weights:
  - Previously defined weight is convergent in practice, but is not easy to analyze
  - Hence, we define

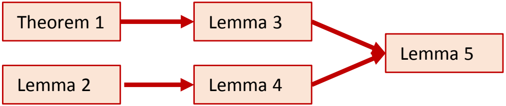
$$u_i^{t+1} = \begin{cases} \frac{\sum_{j\in J} |e_{ij}^t|}{\sum_{j \in J} (e_{ij}^t)^2}, & \text{if } \sum_{j\in J} (e_{ij}^t)^2 > 0, \\ M & \text{if } \sum_{j\in J} (e_{ij}^t)^2 = 0, \end{cases}$$

$$w_i^{t+1} = \begin{cases} w_i^t(1-\beta^t), & \text{if } u_i^{t+1} < w_i^t(1-\beta^t), \\ u_i^{t+1}, & \text{if } w_i^t(1-\beta^t) \le u_i^{t+1} \le w_i^t(1+\beta^t), \\ w_i^t(1+\beta^t), & \text{if } u_i^{t+1} > w_i^t(1+\beta^t), \end{cases}$$

  where M is a large number and $\beta \in (0,1)$
  - With this new weight, we can show w^t is convergent
- If weights w^t are convergent, then A_t is convergent
- $A_t^\top A_t$ is also convergent
- Recall: PCA can be solved by EVD of $A^\top A$

Weighted Algorithm for Error Minimization: Convergent eigenvalues

- Define $B_t = A_t^\top A_t$ (symmetric) and $\Omega_t^s = B_{t+s} - B_t$ (symmetric)
- Notation: $\lambda_i(M)$ (i^{th} eigenvalue of M)
- Theorem 1 [Wielandt-Hoffman] If M and $M + M_e$ are m by m symmetric matrices, then
$$\sum_{i=1}^{m}(\lambda_i(M + M_e) - \lambda_i(M))^2 \leq \|M_e\|_F^2$$
- Lemma 2 $\lim\limits_{\substack{t\to\infty \\ s\to\infty}} \lambda_i(\Omega_t^s) = 0$
- Lemma 3 $\lim\limits_{t\to\infty} [\lambda_i(B_t) - \lambda_i(B_{t+s})] = 0$ for any $s \geq 0$
- Lemma 4 $\lim\limits_{\substack{t\to\infty \\ s\to\infty}} [\lambda_i(B_t) - \lambda_i(B_{t+s})] = 0$
- Lemma 5 $\lambda_i(B_t)$ is convergent in t
- Flow of the proofs



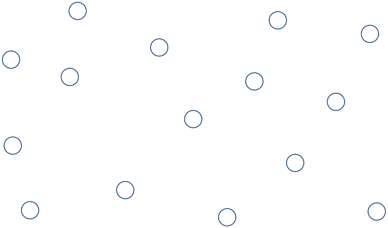
Convergent eigenvalues of Convergent Symmetric Matrices

- Lemma 5 $\lambda_i(B_t)$ is convergent in t
- Lemma 5 can be generalized for a series of convergent symmetric matrices
- Proposition For symmetric matrices $\left\{M_t\right\}_{t=0}^{\infty} \in \mathbb{R}^{m\times m}$ convergent in t, eigenvalues $\lambda_i(M_t)$ for each $i = 1, \cdots, m$ are convergent in t.

LP-based Algorithm for the Maximization of Deviation of Projected Data

LP-based Algorithm for Deviation Maximization: Motivation

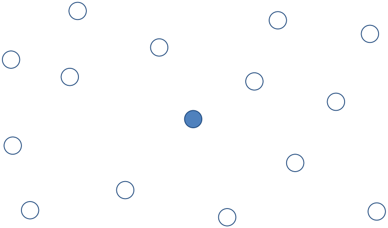
- Motivating concept



Orthogonal matrices

LP-based Algorithm for Deviation Maximization: Motivation

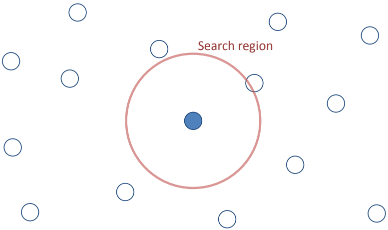
- Motivating concept



Current solution

LP-based Algorithm for Deviation Maximization: Motivation

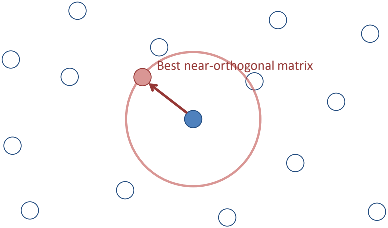
- Motivating concept



Consider nearby non-orthogonal matrices

LP-based Algorithm for Deviation Maximization: Motivation

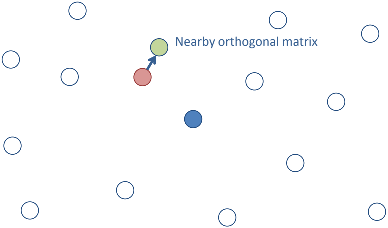
- Motivating concept



Find the best near-orthogonal matrix

LP-based Algorithm for Deviation Maximization: Motivation

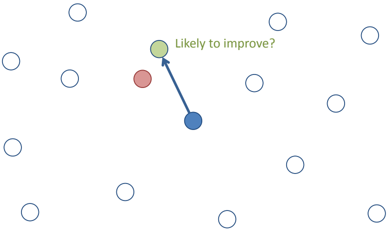
- Motivating concept



Recover orthogonality

LP-based Algorithm for Deviation Maximization: Motivation

- Motivating concept

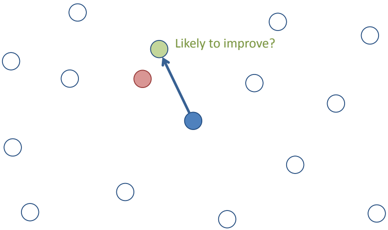


Improvement of objective function value

- Two steps are needed
  - Find the best near-orthogonal matrix in the search region
  - Recover orthogonality

LP-based Algorithm for Deviation Maximization: Motivation

- Motivating concept



Improvement of objective function value

- Two steps are needed
  - Find the best near-orthogonal matrix in the search region : Mathematical programming
  - Recover orthogonality : SVD

LP-based Algorithm for Deviation Maximization: Overall Algorithm (1)

- Notations

  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}
  $\bar{X}$: the current best PCs
  Y: projected data with elements y_{ik}, defined as $Y = AX$
  Λ: perturbation matrix with elements λ_{jk}
  $\hat{X} = \bar{X} + \Lambda^*$: near-orthogonal matrix perturbed from $\bar{X}$
  δ^{max}: maximum allowed change by the perturbation

- Finding the best direction: solve MIP$(\bar{X}, \delta^{max})$ to obtain Λ^*

$$
\begin{aligned}
\max \quad & \textstyle\sum_{i\in I}\sum_{k\in P} |y_{ik}| & \\
s.t. \quad & y_{ik} = \textstyle\sum_{j\in J} a_{ij}(\bar{x}_{jk} + \lambda_{jk}), & i \in I, k \in J, \\
& \sum_{j\in J}\sum_{k\in J} |\lambda_{jk}| \le \delta^{max}, & \\
& |\lambda_{jk}| \le \frac{\delta^{max}}{m\sqrt{m}}, & j \in J, k \in J, \\
& \lambda_{jk}, y_{ik} \text{ unconstrained} &
\end{aligned}
$$

LP-based Algorithm for Deviation Maximization: Overall Algorithm (1)

- Notations

  A: data matrix of n observations and m attributes with elements a_{ij}
  X: principal components matrix with elements x_{jk}
  $\bar{X}$: the current best PCs
  Y: projected data with elements y_{ik}, defined as $Y = AX$
  Λ: perturbation matrix with elements λ_{jk}
  $\hat{X} = \bar{X} + \Lambda^*$: near-orthogonal matrix perturbed from $\bar{X}$
  δ^{max}: maximum allowed change by the perturbation

- Finding a good direction: **fix the sign of y_{ik} to match $\bar{y}_{ik}$**, solve LP$(\bar{X}, \delta^{max})$ to obtain Λ^*

$$
\begin{array}{lll}
\max & \sum_{i \in I} \sum_{k \in P} |y_{ik}| & \\
s.t. & y_{ik} = \sum_{j \in J} a_{ij}(\bar{x}_{jk} + \lambda_{jk}), & i \in I, k \in J, \\
& \sum_{j \in J} \sum_{k \in J} |\lambda_{jk}| \leq \delta^{max}, & \\
& |\lambda_{jk}| \leq \frac{\delta^{max}}{m\sqrt{m}}, & j \in J, k \in J, \\
& y_{ik} \geq 0, & \text{if } \bar{y}_{ik} \geq 0, \\
& y_{ik} \leq 0, & \text{if } \bar{y}_{ik} \leq 0, \\
& \lambda_{jk} \text{ unconstrained}, &
\end{array}
$$

where LP$(\bar{X}, \delta^{max})$ can be converted into an LP

LP-based Algorithm for Deviation Maximization: Overall Algorithm (2)

- Orthogonality Recovery: given near-orthogonal matrix $\hat{X}$,
  - Do SVD of $\hat{X}$ and obtain $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^\top$
  - Replace $\hat{\Sigma}$ with I_m, m by m identity matrix
  - $\tilde{X} = \hat{U} I_m \hat{V}^\top$ is orthogonal matrix

- Notation

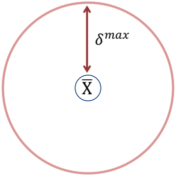
  ρ^u: scaling factor for δ^{max}, $\rho^u > 1$
  ρ^d: scaling factor for δ^{max}, $\rho^d \in (0, 1)$
  Δ^{min}: lower bound for δ^{max}
  Δ^{max}: upper bound for δ^{max}
  ε: precision tolerance, $\varepsilon > 0$,

- Algorithm
  - Initialize $\bar{X}$, δ^{max}
  - **While** $\delta^{max} > \Delta^{min}$

    $\hat{X} \leftarrow$ solve LP$(\bar{X}, \delta^{max})$
    $\tilde{X} \leftarrow \textit{SVD-Heur}(\hat{X})$
    If $G(\tilde{X}) \geq G(\bar{X}) + \varepsilon$ **then** $\bar{X} \leftarrow \tilde{X}$, $\delta^{max} \leftarrow \min\{\rho^u delta^{max}, \Delta^{max}\}$
    Else $\delta^{max} \leftarrow \rho^d delta^{max}$

LP-based Algorithm for Deviation Maximization: Convergence

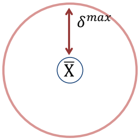
- Concept: shrinking feasible region for LP($\bar{X}, \delta^{max}$) if there is no update of $\bar{X}$



Current best matrix $\bar{X}$ and search region

LP-based Algorithm for Deviation Maximization: Convergence

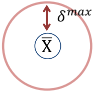
- Concept: shrinking feasible region for LP($\bar{X}, \delta^{max}$) if there is no update of $\bar{X}$



Current best matrix $\bar{X}$ and shrinking search region

LP-based Algorithm for Deviation Maximization: Convergence

- Concept: shrinking feasible region for LP($\bar{X}, \delta^{max}$) if there is no update of $\bar{X}$



Current best matrix $\bar{X}$ and shrinking search region

LP-based Algorithm for Deviation Maximization: Convergence

- Concept: shrinking feasible region for LP($\bar{X}, \delta^{max}$) if there is no update of $\bar{X}$

Current best matrix $\bar{X}$ fitting into search region

- Proposition $\displaystyle \lim_{\delta^{max} \to 0^+} \hat{X} = \lim_{\delta^{max} \to 0^+} \tilde{X} = \bar{X}$

  * The proof is based on singular values of $\hat{X}$ and inequalities between several matrix norms

Computational Experiment for wPCA and lpPCA

Computational Experiment: Instances

- Synthetic Instances are generated
  - $(m, n) = \{(20, 100), (20, 300), (50, 100), (50, 300)\}$
  - have approximate rank $q = 10$
  - $r \in \{0, 10, 20, 30\}$ % of observations with a higher variance
  - Each (m, n, r) tuple has 5 difference instances
  - Total 80 instances
  - Run the algorithms with various number of PCs: $p \in \{8, 9, 10, 11, 12\}$ (around q)

- Instances extracted from classification dataset from the UCI Machine Learning Repository
  - Small instances ($mn \leq 15000$): cancer_2, cancer_4, ilpd_1, ilpd_2, cardio_1, cardio_2, iono_b, iono_g, sonar_g, sonar_r, landsat_1, landsat_3
  - Large instances: spam_0, spam_1, magic_g, magic_h, blocks_1, hand_0, hand_1

Computational Experiment: Benchmark and Comparison

- Benchmark algorithms in the literature
  - Ke and Kanade(2005), Brooks *et al*(2013) for the minimization of reconstruction error
  - Kwak(2008) and Nie *et al*(2011) for the maximization of projected deviation
  - Ke and Kanade(2005), Brooks *et al*(2013), Kwak(2008) are implemented in R by Brooks and Jot(2012)
  - We implement Nie *et al*(2011) in R script
- Performance measures
  - Improvement by our algorithm from the benchmark

$$\Delta_{\text{wPCA}} = 1 - \frac{F_{\text{wPCA}}}{\min\{F_{\text{Ke}}, F_{\text{Brooks}}, F_{\text{Kwak}}, F_{\text{Nie}}\}},$$
$$\Delta_{\text{mipPCA}} = \frac{G_{\text{mipPCA}}}{\max\{G_{\text{Ke}}, G_{\text{Brooks}}, G_{\text{Kwak}}, G_{\text{Nie}}\}} - 1,$$
$$\Delta_{\text{lpPCA}} = \frac{G_{\text{lpPCA}}}{\max\{G_{\text{Ke}}, G_{\text{Brooks}}, G_{\text{Kwak}}, G_{\text{Nie}}\}} - 1.$$

  - Average ranking
  - Average execution time
- All parameters are tuned
- lpPCA is executed for maximum of 5 seconds and 5 iterations

Computational Experiment: wPCA for Synthetic Instances

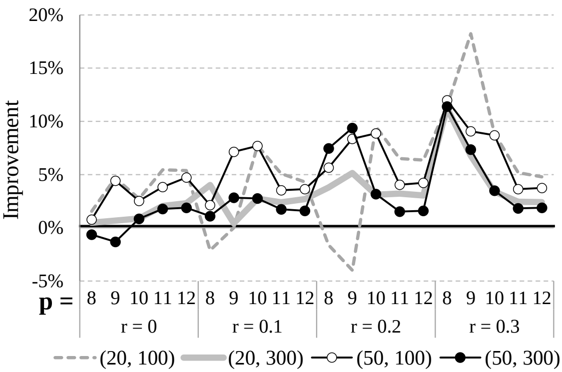


Figure : Average Δ_{wPCA}: Above 0 for most cases

Computational Experiment: wPCA for Synthetic Instances

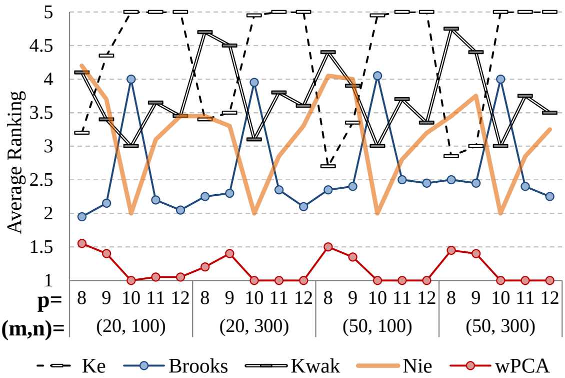


Figure : Average ranking: constantly near 1

Computational Experiment: wPCA for Synthetic Instances

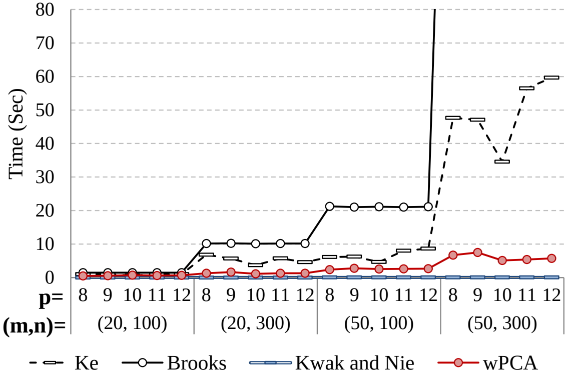


Figure : Average execution time: competitive

Computational Experiment: wPCA for Small UCI instances

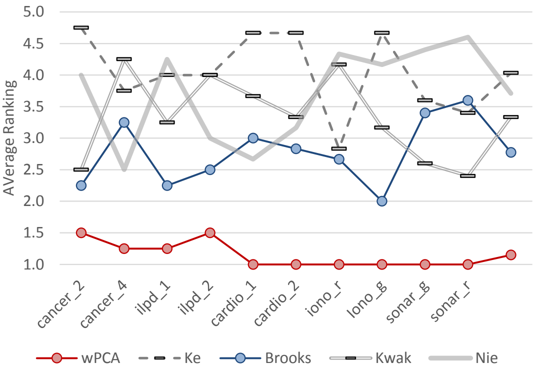


Figure : Average ranking over p values: near rank 1

Computational Experiment: wPCA for Large UCI instances

- Due to scalability issue, only Kwak, Nie are executed among the benchmark algorithms.

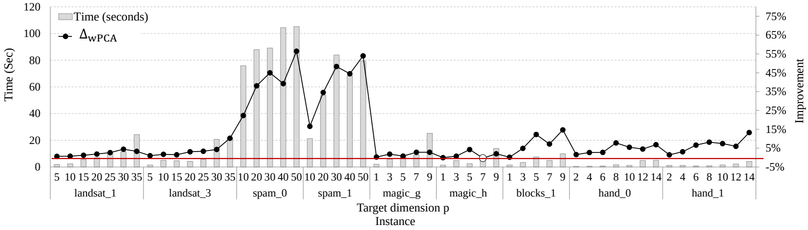


Figure : Δ_{wPCA}: all positive except for one

Computational Experiment: lpPCA with Time Limit for Synthetic Instances

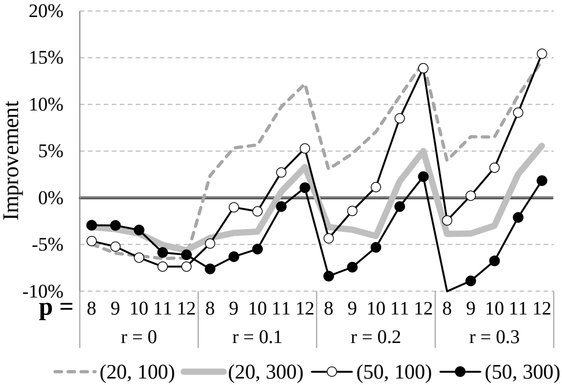


Figure : Average Δ_{wPCA}: better as p increases

Computational Experiment: lpPCA with Time Limit for Synthetic Instances

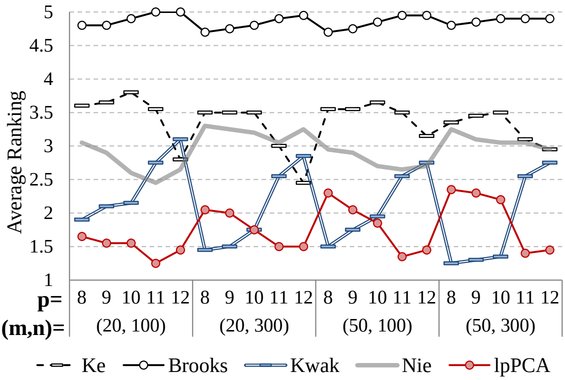


Figure : Average ranking: better when $p > 10$

Computational Experiment: lpPCA with Time Limit for Synthetic Instances

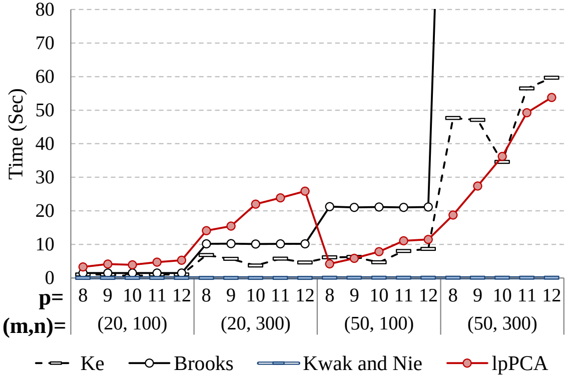


Figure : Average execution time: competitive

Computational Experiment: lpPCA with Time Limit for Small UCI Instances

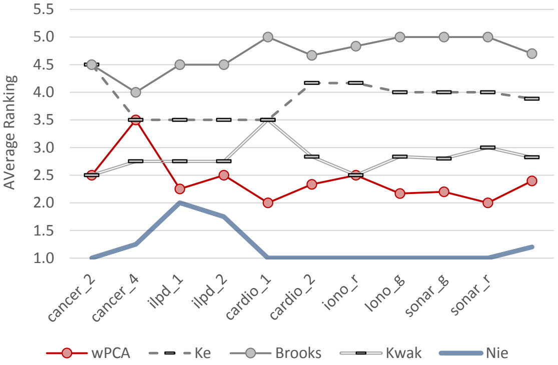


Figure : Average ranking: constantly near 2

Computational Experiment: lpPCA with Time Limit for Large UCI Instances

- Due to scalability issue, only Kwak, Nie are executed among the benchmark algorithms.

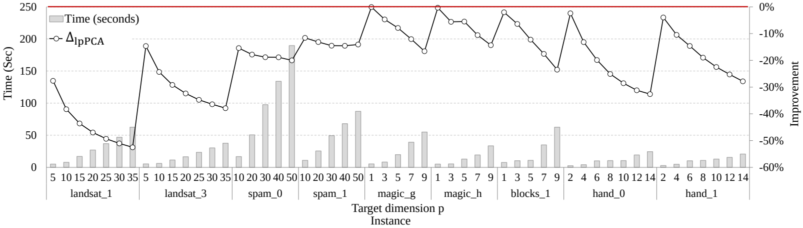


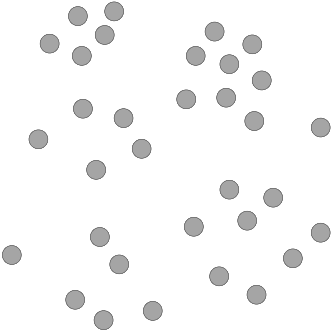
Figure : Δ_{lpPCA}: all negative

PCA: Contributions and Major Findings

1. For the minimization problem with the L_1 norm, we propose the first IRLS algorithm.
2. We show that the algorithm gives convergent eigenvalues
3. The result is generalized to eigenvalues of symmetric convergent matrices
4. For the maximization with the L_1 norm, we propose the mathematical programming based algorithm together with SVD to overcome the difficulty caused by the orthogonal constraint. We also show that the algorithm is convergent.
5. The results of the computational experiment show that both of the proposed algorithms outperform the benchmark algorithms in most cases in the presence of significant outliers.

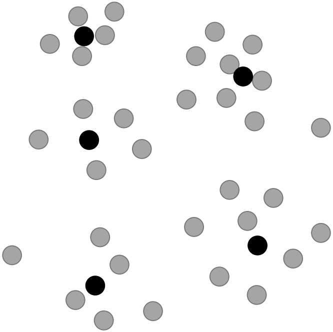
Disaggregate and Bound (DAB): Clustering and Machine Learning with Centroids

Motivation



When data size is large, solving an optimization problem is intractable

Motivation



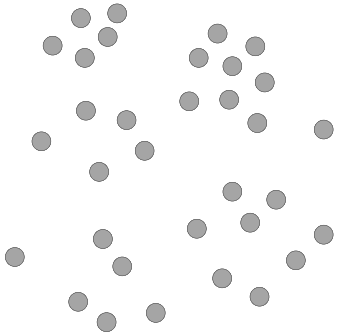
Can we optimize with centroids?

Contribution

- A clustering-based iterative algorithm (Disaggregate and Bound, DAB) to solve certain optimization problems in machine learning is proposed
- For least absolute deviation regression (LAD) and support vector machine (SVM), the algorithm monotonically converges to global optimum, while providing the optimality gap in each iteration
- Computational experiment shows that DAB outperforms when the data size is large

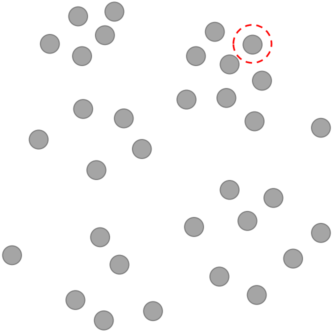
Algorithm Disaggregate and Bound (DAB)

Definition



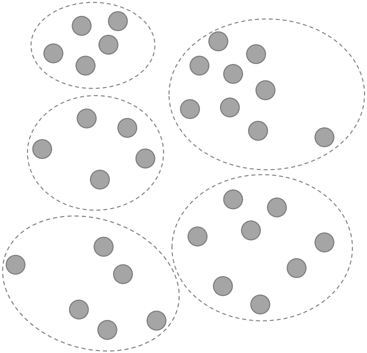
Original Data

Definition



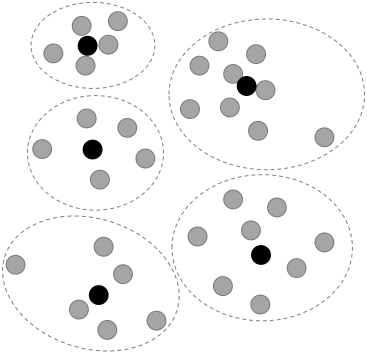
Original Observation

Definition



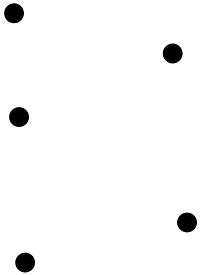
Clusters

Definition



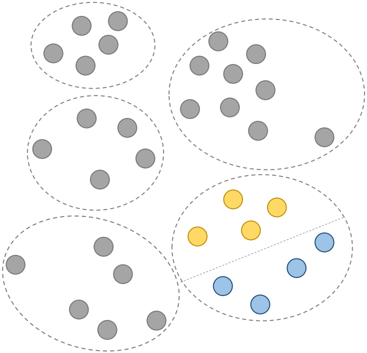
Aggregated Observations

Definition



Aggregated Data

Definition



Declustering

Algorithm: Disaggregate and Bound (DAB)

- Components of the algorithm: needs to be tailored for a particular machine learning problem, optimality is achieved
  - Definition of clusters and the aggregated data
  - Declustering procedure: how to partition the current clusters?
  - Optimality criteria: under what condition, optimality is achieved?
- DAB: algorithmic framework

  Initialization: Define clusters and the aggregated data

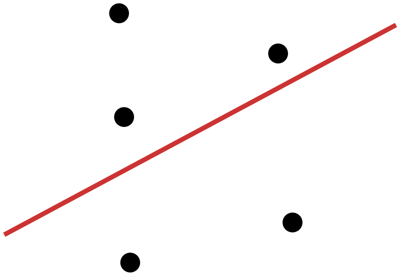
  While Optimality condition is not satisfied

    Solve the problem with the aggregated data

    Decluster

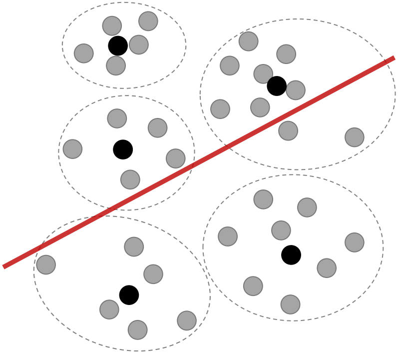
  End While

DAB for LAD Regression



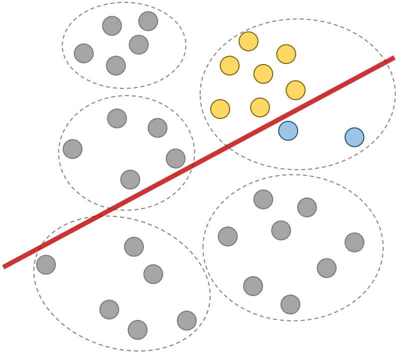
Solve with the aggregated data

DAB for LAD Regression



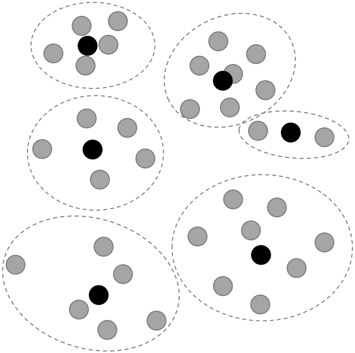
Check optimality criteria

DAB for LAD Regression



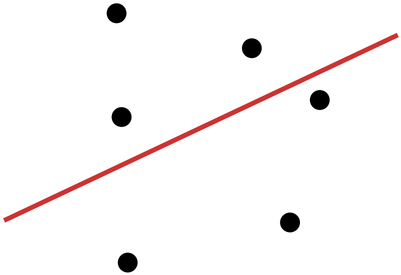
Decluster

DAB for LAD Regression



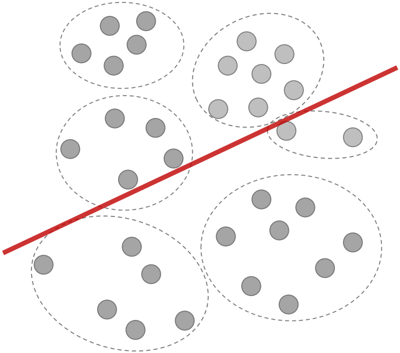
Create new aggregated data

DAB for LAD Regression



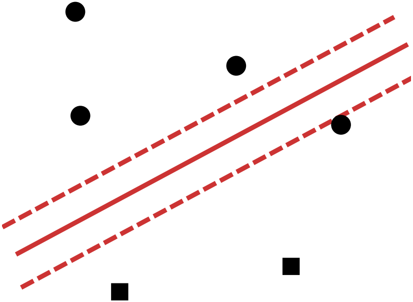
Solve with the aggregated data

DAB for LAD Regression



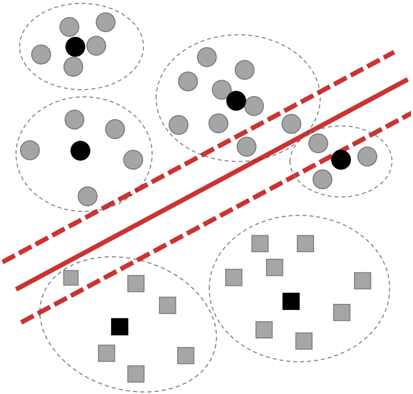
Check optimality criteria

DAB for SVM



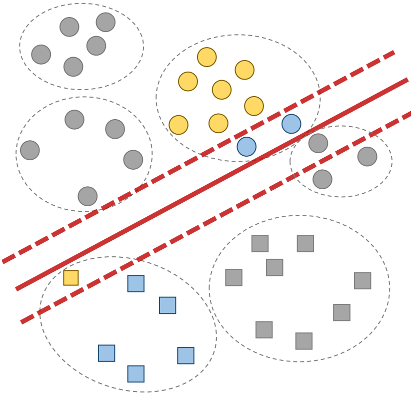
Solve with the aggregated data

DAB for SVM



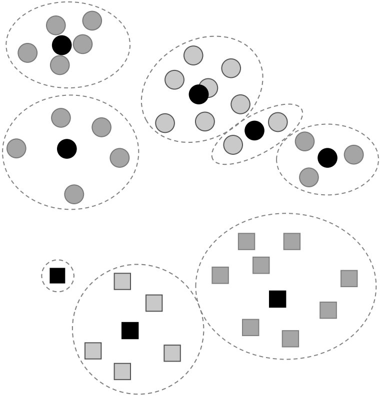
Check optimality criteria

DAB for SVM



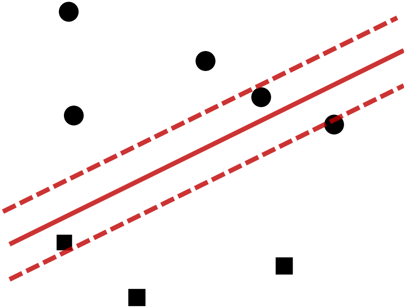
Decluster

DAB for SVM



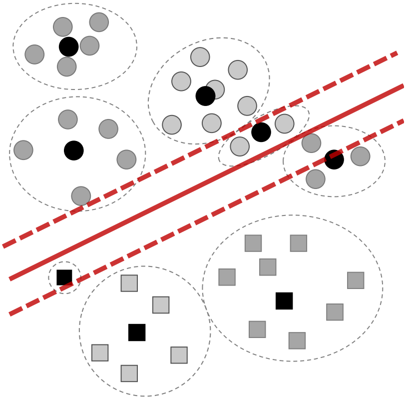
Create new aggregated data

DAB for SVM



Solve with the aggregated data

DAB for SVM



Check optimality criteria

Convergence Result

For both LAD and SVM, DAB has the following properties

Proposition 1 If the optimality condition is satisfied, then the optimal solution for the aggregated problem is an optimal solution to the problem with the original data

Proposition 2 Objective function value of the aggregated problem is non-decreasing in iteration.

$\Rightarrow$ By Propositions 1 and 2, DAB is monotonically convergent to the global optimum of the original problem.

Computational Experiment

- Implementation

  LAD: R script with package **quantreg**

  SVM: Python script with package **scikit learn**

- Notation

  r^0: initial aggregation rate for DAB

  r^T: final aggregation rate for DAB

  T: number of iterations for DAB

  $\mathcal{T}^{\text{init}}$: initialization time of DAB

  $\mathcal{T}^{\text{loop}}$: loop time of DAB

  $\mathcal{T}^{\text{DAB}}$: total execution time of DAB

  $\mathcal{T}^{\text{fn}}$, $\mathcal{T}^{\text{libsvm}}$: time of benchmark algorithms

  $\rho = \frac{\mathcal{T}^{\text{DAB}}}{\mathcal{T}^{\text{fn}}}$: if $\rho < 1$, then DAB is faster

Computational Experiment: LAD

Instance		DAB						fn	
n	m	r^0	r^T	T	$\mathcal{T}^{\text{init}}$	$\mathcal{T}^{\text{loop}}$	$\mathcal{T}^{\text{DAB}}$	$\mathcal{T}^{\text{fn}}$	ρ
200,000	10	0.05%	2.4%	8	1	49	50	1	34.59
200,000	100	0.10%	10.0%	9	2	83	84	22	3.83
200,000	500	0.50%	16.5%	7	9	443	451	393	1.15
200,000	800	0.80%	22.4%	7	17	1331	1347	1062	1.27
400,000	10	0.05%	2.2%	8	2	97	99	3	29.84
400,000	100	0.05%	5.8%	9	3	160	163	44	3.68
400,000	500	0.25%	13.3%	8	17	887	904	904	1.00
400,000	800	0.40%	21.1%	8	33	2656	2689	2125	1.27
800,000	10	0.05%	0.9%	5	5	134	139	9	15.46
800,000	100	0.05%	5.0%	9	7	330	336	96	3.51
800,000	500	0.13%	10.3%	9	38	1750	1788	1851	0.97
800,000	800	0.30%	10.3%	7	73	2920	2992	15215	0.20
1,600,000	10	0.05%	0.4%	4	18	216	235	18	13.29
1,600,000	100	0.05%	3.2%	8	18	595	612	196	3.12
1,600,000	500	0.09%	5.35%	8	82	2378	2460	12164	0.20

Table : LAD result

Computational Experiment: SVM

Instance		DAB						libsvm	
n	m	r^0	r^T	T	$\mathcal{T}^{\text{init}}$	$\mathcal{T}^{\text{loop}}$	$\mathcal{T}^{\text{DAB}}$	$\mathcal{T}^{\text{libsvm}}$	ρ
30,000	10	1.0%	2.1%	3.5	1	6	7	50	0.14
30,000	30	1.0%	8.2%	6.5	2	11	13	20	0.68
30,000	50	1.0%	11.8%	7	3	20	23	35	0.65
50,000	10	1.0%	1.5%	3	4	9	13	127	0.10
50,000	30	1.0%	5.2%	5.5	7	26	33	133	0.25
50,000	50	1.0%	8.8%	6.7	11	30	41	90	0.45
100,000	10	1.0%	1.4%	1.8	18	20	38	431	0.09
100,000	30	1.0%	3.7%	5.4	34	30	64	165	0.39
100,000	50	1.0%	5.6%	6	50	45	95	283	0.33
150,000	10	1.0%	1.1%	0.8	55	31	87	1086	0.08
150,000	30	1.0%	2.4%	4.3	93	49	142	1601	0.09
150,000	50	1.0%	4.8%	6	134	68	202	599	0.34

Table : SVM result