MM algorithm for Quantile regression for censored data with missing observations

Samidha Shetty

Pennsylvania State University

4^{th} December 2018

Quantile regression

A linear quantile regression model is given as below :

$$Y_i = Z_i^T \theta_q + r_i, \quad i = 1, \ldots, n$$

such that $P(r_i \leq 0 | Z_i) = q$ or $E\{q - I(r_i \leq 0) | Z_i)\} = 0$
Here Y_i : response, Z_i : vector of covariates, θ_q : unknown coefficient vector (which depends on q), r_i : error term

What are the advantages of quantile regression over mean regression ?

Koenker and Bassett(1978) defined $\hat{\theta}$ as the minimizer of

$$L(\theta) = \sum_{i=1}^{n} \rho_q[y_i - z_i^T \theta] = \sum_{i=1}^{n} \rho_q(r_i(\theta))$$

where $\rho_q(r) = |r|[q - I(r \leq 0)]$.

But this is not easily optimized as $\rho_q(r)$ is non-differentiable at $r = 0$.

MM algorithm

Hunter and Lange (2000) introduced an MM algorithm to optimize $L(\theta)$.

First, $L(\theta)$ is approximated by $L_\varepsilon(\theta) = \sum_{i=1}^{n} \rho_q^\varepsilon(r_i)$, where,

$$\rho_q^\varepsilon(r) = \rho_q(r) - \frac{\varepsilon}{2} ln(\varepsilon + |r|)$$

Second, the approximated function is minimized using an MM algorithm.
At k^{th} iteration $\rho_q^\varepsilon(r)$ is majorized by

$$\zeta_q^\varepsilon(r|r^k) = \frac{1}{4}\left[\frac{(r)^2}{\varepsilon + |r^k|} + (4q-2)r + c\right]$$

where is c is such that $\zeta_q^\varepsilon(r^k|r^k) = \rho_q^\varepsilon(r^k)$.

MM algorithm

Thus the majorizer for $L_\varepsilon(\theta)$ is given as

$$Q_\varepsilon(\theta|\theta^k) = \sum_{i=1}^{n} \zeta_q^\varepsilon(r_i|r_i^k)$$

In the linear case, one can solve explicitly for θ^{k+1}, but otherwise, just reducing the value of $Q_\varepsilon(\theta|\theta^k)$ at each iteration suffices.

MM algorithm for Quantile regression

1. Initialize θ^0 and small constant ε such that $\varepsilon n|\ln\varepsilon| = \tau$. Set $k = 0$.
2. At every k^{th} iteration $\theta^{k+1} = \theta^k + \alpha^k\phi_\varepsilon^k$ where α^k is step size and ϕ_ε^k is step direction.
3. Replace $k = k + 1$. Until $\frac{Q_\varepsilon(\theta^{k+1}|\theta^k) - Q_\varepsilon(\theta^k|\theta^k)}{Q_\varepsilon(\theta^k|\theta^k)} < \tau$.

Censored data

Censoring, roughly speaking, is when the value of a observation is only partially known.
Right censoring : We don't have the actual value of the observation, but instead know that it is above a certain value i.e. we observe $Y_i = min(T_i, c_i)$ and $\Delta_i : I(T_i \leq c_i)$ is an indicator of censoring.
Xie et al. (2015) used an inverse probability weighted estimating function of the form

$$\sum_{i=1}^{n} \frac{\Delta_i}{G(y_i|Z_i)} \rho_q[y_i - z_i^T \theta]$$

where $G(Y_i|Z_i)$ is the survival function which is estimated using the Kaplan-Meier estimator. When c_i is independent of covariates,

$$\widehat{G}(t|Z_i) = \widehat{G}(t) = \prod_{s \leq t} \left\{ 1 - \frac{\#\text{of deaths before time s}}{\#\text{of surviving people at time s}} \right\}$$

Missing values

To deal with missing values in quantile estimation (Chen et al. (2014) devised an inverse probability weighting method that estimates the probability weights non-parametrically.
The objective function is modifed as follows :

$$\sum_{i=1}^{n} \frac{\delta_i}{\widehat{p}(X_i)} \rho_q[y_i - z_i^T \theta]$$

where X_i is the matrix of response and subset of covariates which have complete data, δ_i is indicator of completeness of data for i^{th} observation.
$\widehat{p}(X_i) = \frac{\sum_{i=1}^{n} K_h(X_i - X_j)\delta_i}{\sum_{i=1}^{n} K_h(X_i - X_j)}$ where $K_h(u) = K(u/h)/h^d$. Here d is the dimension of X_i and $K(\cdot)$ is a d-variate probability density function.
I have combined these two methods, and get the following objective function :

$$\sum_{i=1}^{n} \frac{\Delta_i}{\widehat{G}(y_i|Z_i)} \frac{\delta_i}{\widehat{p}(X_i)} \rho_q[y_i - z_i^T \theta]$$

Simulation Study

Model : $Y_i = 4.5Z_{1i} - 2Z_{2i} + r_i$
where $Z_{1i} \sim Normal(0, 1)$, $Z_{2i} \sim Uniform(-3, 3)$ and $r_i \sim Normal(0, 1)$. I have used quantiles of Y to censor the data to attain the particular amount of censoring. Only covariate Z_2 has missing values.

For θ_1 :

C %	M %	$q = 0.25$		$q = 0.5$		$q = 0.75$	
		Bias	MSE	Bias	MSE	Bias	MSE
25%	30%	−0.2486	0.0719	−0.2474	0.0693	−1.0631	1.1749
50%	30%	−0.2172	0.0586	−2.2552	5.1272	−2.2495	5.1068
25%	50%	−0.2387	0.0703	−0.6786	0.5033	−1.0252	1.1038

For θ_2 :

C %	M %	$q = 0.25$		$q = 0.5$		$q = 0.75$	
		Bias	MSE	Bias	MSE	Bias	MSE
25%	30%	0.1140	0.0155	0.1136	0.0153	0.4406	0.2009
50%	30%	0.1167	0.0163	0.9935	0.9949	0.9829	0.9738
25%	50%	0.1058	0.0145	0.2982	0.0973	0.4386	0.2009

Boxplot for θ_1 estimates

Situation 1 : 25% right censoring and 30% missing observations
Situation 2 : 50% right censoring and 30% missing observations
Situation 3 : 25% right censoring and 50% missing observations

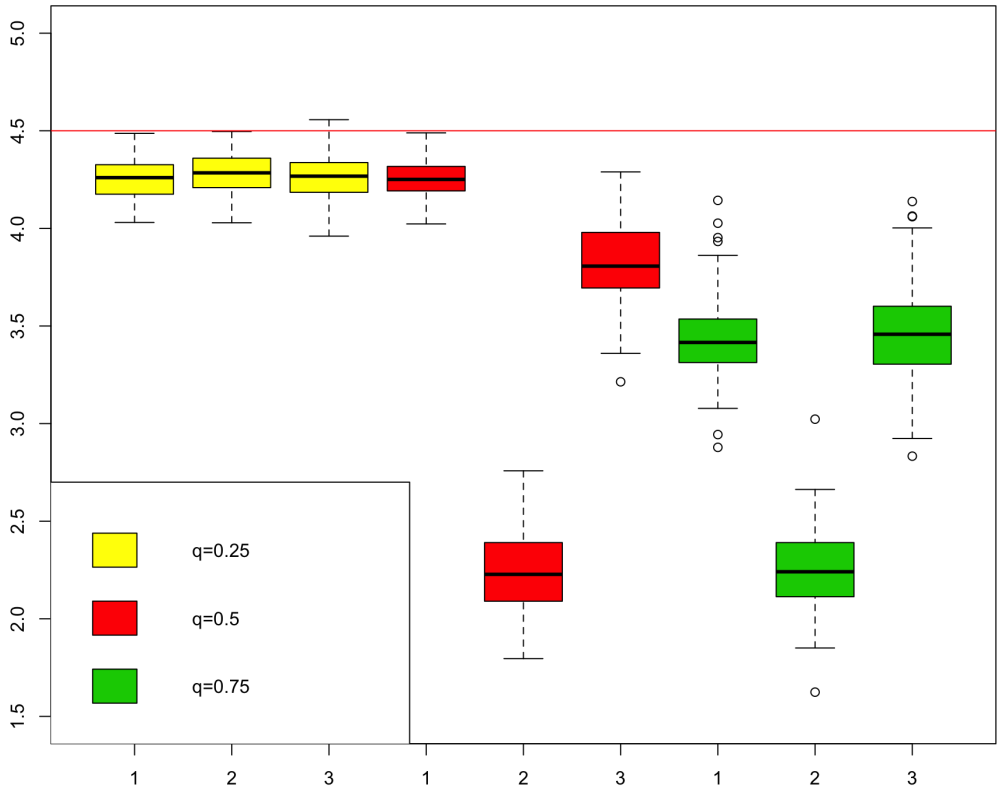


FIGURE – Box plot for θ_1 estimates

Observations and future work

Observations and challenges

- The method estimates better for smaller quantiles.
- Increase in censoring affects the estimates drastically.
- The estimates for θ_1 are slightly worse than those for θ_2.
- There seems to be a bias present in the estimation.
- Computation time : For n=500 it took 23.6 sec, n=1000 it took 130 sec and n=2000 it took 835 sec.

Future work

- Derive theoretical results for the combination of the two methods.
- Considering case where censoring is covariate dependent.
- Extending the methods to partially linear quantile regression.

Adam: A Method for Stochastic Optimization

Jun Tao

Penn State University

Dec. 2018

Motivation: SGD

Stochastic gradient descent (SGD) randomly selects a subset of the data to calculate the gradient. It is efficient comparing to naive gradient descent. However, it performs frequent updates with a high variance that causes the convergence path to fluctuate heavily.

The toy example on the right side shows how SGD behaves in simple linear regression, where the optimal function is quadratic. Within 600 iterations, SGD progresses slowly in the neighborhood of the optimum.

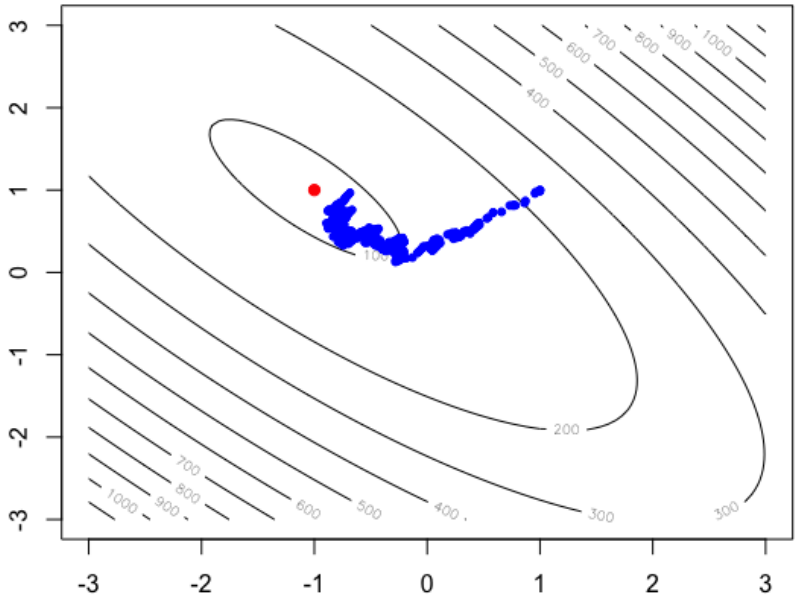


Figure 1: Toy example of SGD

Momentum Method

The momentum method (Polyak, 1964), or classical momentum (CM), is a technique for accelerating SGD that accumulates a velocity vector in directions of gradient across iterations. Polyak showed that CM can considerably accelerate convergence to a local minimum.

$$v_{t+1} = \beta v_t + (1-\beta)\nabla_\theta f(\theta_t);$$
$$\theta_{t+1} = \theta_t - \alpha v_{t+1},$$

where α is the learning rate and β is the momentum coefficient. They are both hyperparameters.

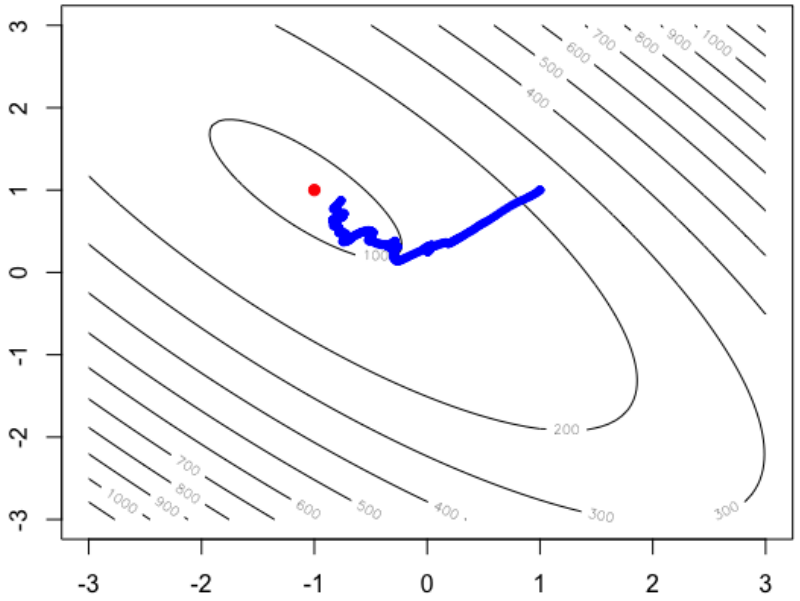


Figure 2: Toy example of CM

RMSProp

Root Mean Square Propogation (RMSprop) is an unpublished, adaptive learning rate method proposed by Geoff Hinton. It has been successfully employed in several practical applications but there's no theoretical guarantee of global convergence.

RMSprop divides the learning rate by an exponentially decaying average of squared gradients.

$$S_{t+1} = \beta S_t + (1-\beta)(\nabla_\theta f(\theta_t)).^2;$$
$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta f(\theta_t)/\sqrt{\epsilon + S_{t+1}},$$

where α is the learning rate and β is weight hyperparameter. ϵ is a small quantity to make sure the denominator is nonzero.

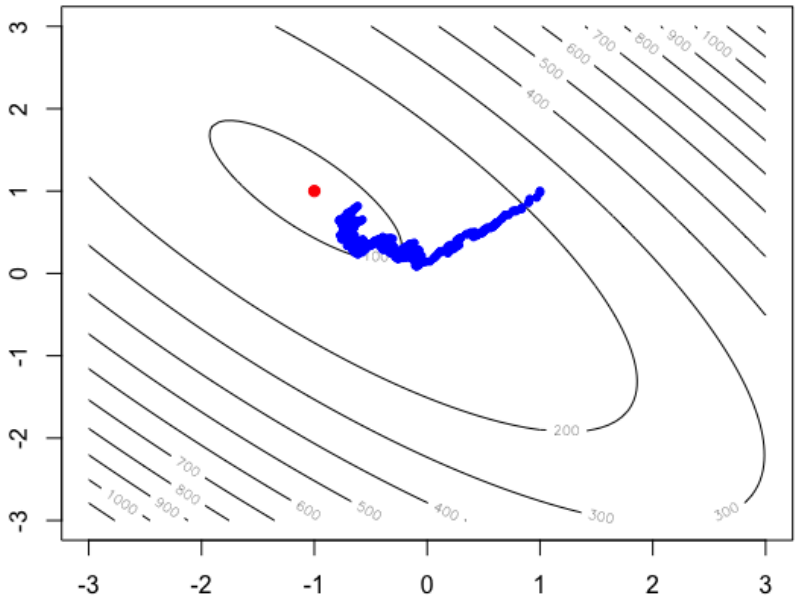


Figure 3: Toy example of RMSprop

Adam

Adam is a combination of CM and RMSprop with additional de-biasing process. The name Adam is derived from adaptive moment estimation.

Algorithm 1 Adam

1: Set proper initial values, $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.
2: Require $f(\theta)$ and θ_0, the stochastic objective function and the starting point.
3: Initialize variables: $m_0 \leftarrow 0$, $v_0 \leftarrow 0$ and $t \leftarrow 0$.
4: **while** θ_t does not converge **do**
5: [Step 1] $t \leftarrow t + 1$;
6: [Step 2] $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$;
7: [Step 3] $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$;
8: [Step 4] $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$;
9: [Step 5] $\alpha_t \leftarrow \alpha \cdot \sqrt{1 - \beta_2^t}/(1 - \beta_1^t)$;
10: [Step 6] $\theta_t \leftarrow \theta_{t-1} - \alpha_t \cdot m_t/(\sqrt{v_t} + \epsilon\sqrt{1 - \beta_2^t})$;
11: **end while**; **return** θ_t

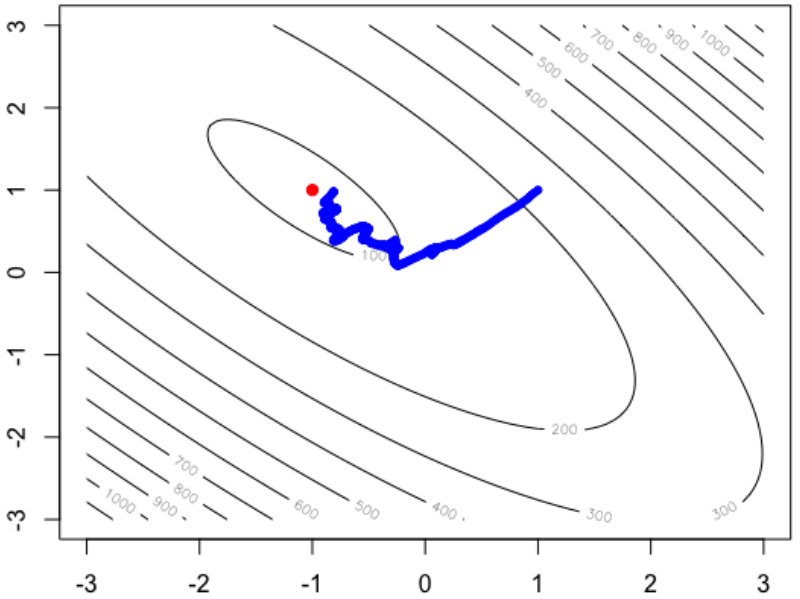


Figure 4: Toy example of Adam

Example: Logistic Regression

- In logistic regression setting, the goal is to maximize loglikelihood function $\ell(\boldsymbol{\beta}) = \sum_i \ell_i(\boldsymbol{\beta}) = \sum_i [y_i \eta_i - \log(1 + \exp(\eta_i))]$.
- Newton-Raphson method requires the inverse of observed Fisher information for $\boldsymbol{\beta}$, which can be difficult to obtain when p is large.
- For simulation, choose $p = 19$, $x_i \sim N(0, .7I_p + .3\mathbf{1}_p\mathbf{1}_p^T)$, $\beta \sim U(-1, 1)^{p+1}$. Training data size $= 1000$ and test size $= 250$.

	SGD	CM	Adam	N-R
time (s)	0.23	0.35	0.08	0.01
training error (%)	15.83	15.83	22.71	16.04
test error (%)	30.83	30.83	27.50	30.83
$\left\|\hat{\beta} - \beta\right\|$	0.66	0.77	2.11	0.72

Example: Logistic Regression

- For simulation, choose $p = 99$, $x_i \sim N(0, .7I_p + .3\mathbf{1}_p\mathbf{1}_p^T)$, $\beta \sim U(-1, 1)^{p+1}$. Training data size $= 1000$ and test size $= 250$.

	SGD	CM	Adam	N-R
time (s)	1.26	1.27	0.57	0.15
training error (%)	1.04	1.25	4.17	0*
test error (%)	13.33	12.50	11.70	16.70
$\left\|\hat{\beta} - \beta\right\|$	3.18	3.26	3.04	151.50

*: I used "glm" function in R. Actually N-R fails to work in this case. The warning message said: the algorithm does not converge.

Real Data

- MNIST is database of handwritten digits. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.
- I applied Adam to distinguishing 0 from 6.

	SGD	CM	Adam	N-R
time (s)	1.58	1.05	0.30	203.90
training error (%)	2.34	2.31	3.25	NA*
test error (%)	2.37	2.79	3.25	NA

*: For MNIST data, identification of 0 and 6 has $n = 11841$ and $p = 28^2 = 784$. "glm" function can not return a proper answer.

Potential Problem of Adam

- Adam may miss the global optimum.
- Possible solution (Keskar, 2017): first use Adam to locate the rough area of the optimum, and then switch to SGD.

A Sparse Singular Value Decomposition Method for Spiked-mean Model

Qing Sun

Penn State

December 2, 2018

Introduction

Basic Problem: Given a high-dimension observation matrix data X with noise:

$$X = Y + Z, \qquad Z \text{ is the noise}$$

Want to reconstruct or approximate the true structure Y from X.

Spiked Mean Model: Given a large, noisy matrix data X, assume the variables $X_{i,j}$ can be modeled as

$$X = UDV' + Z, \qquad Z_{i,j} \sim \text{ iid }, E(Z_{i,j}) = 0, \ \text{var}(Z_{i,j}) = \sigma^2$$

$U_{n\times r}$, $V_{p\times r}$ are sparse singular vectors, with $r \ll \min(n, p)$.

Example: two-way functional data, $Y_{i,j} = Y(s_i, t_j)$. As smooth function of (s, t), if expand Y in suitable basis, the coefficient should be sparse.

Challenge and a Solution

Difficulties:

1. X is high-dimension, the accumulation of the noise $Z_{i,j}$ results in poor estimate when apply classical SVD on X;
2. The computation involves many structureless cells $Z_{i,j}$, thus computation expensive.

A possible Solution:

Fast Iterative Thresholding-SparseSVD (FIT-SSVD):
Combine classical SVD with thresholding step in each iteration.

Methods

For the **Classical SVD Iteration**, given a right starting frame $V^{(0)}$, a $p \times r$ orthonormal columns, repeat

(1) Right-to-Left: $\tilde{U}^{(k)} = XV^{(k-1)}$
(2) Left QR: $U^{(k)} R_u^{(k)} = \tilde{U}^{(k)}$
(3) Left-to-Right: $\tilde{V}^{(k)} = X'U^{(k)}$
(4) Right QR: $V^{(k)} R_v^{(k)} = \tilde{V}^{(k)}$

FIT-SSVD:

(1) Right-to-Left: $\tilde{U}^{(k)} = XV^{(k-1)}$
(2) Left Thresholding: $\tilde{U}^{(k),thr} = \eta(\tilde{U}^{(k)}, \gamma_u^{(k)})$
(3) Left QR: $U^{(k)} R_u^{(k)} = \tilde{U}^{(k),thr}$
(4) Left-to-Right: $\tilde{V}^{(k)} = X'U^{(k)}$
(5) Right Thresholding: $\tilde{V}^{(k),thr} = \eta(\tilde{V}^{(k)}, \gamma_v^{(k)})$
(6) Right QR: $V^{(k)} R_v^{(k)} = \tilde{V}^{(k),thr}$

FIT-SSVD: threshold level

Choose suitable threshold level γ is tricky:

- If γ too small, only kick out few structureless elements, make little benifit.
- If γ too large, shave off too many elements, give results with high bias.

Recall $X = UDV' + Z$, Given the present estimate $V^{(k-1)}$

$$\tilde{U}^{(k)} = XV^{(k-1)} = UDV'V^{(k-1)} + ZV^{(k-1)}$$

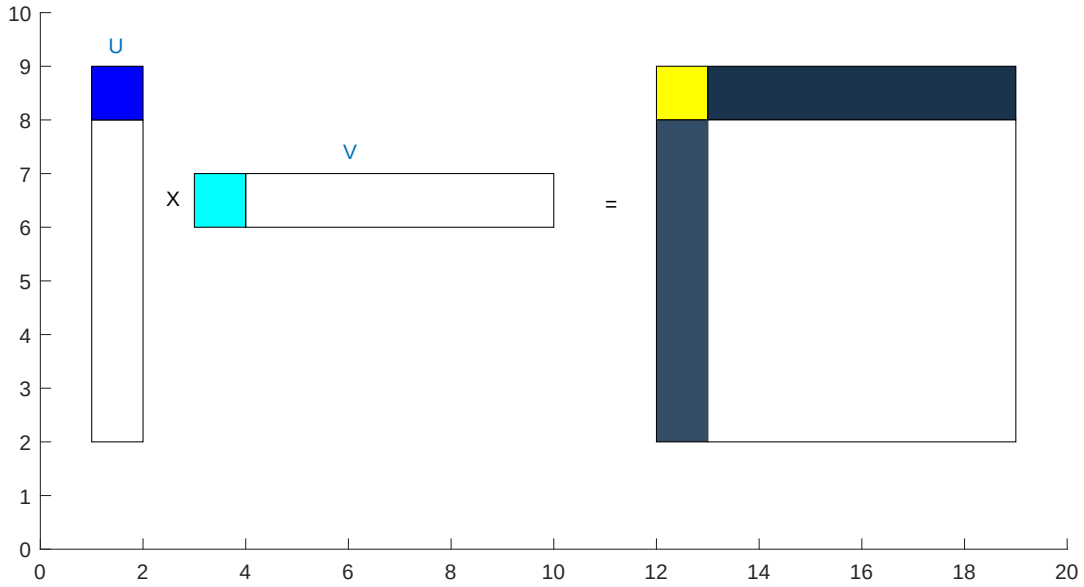
Theoretical threshold level:

$$\gamma = E(\|ZV^{(k-1)}\|_\infty)$$

The element in $\tilde{U}^{(k)}$ with absolute value less than γ can be regarded low signal since it is weaker than the expected noise level.

FIT-SSVD: threshold level

Question: $\gamma = E(\|ZV^{(k-1)}\|_\infty)$ requires the information of Z, we only has the observation data X.
Given the present estimates, $U^{(k-1)}, V^{(k-1)}$



Regard elements in white area as the samples from noise, nonparametric bootstrap Z^*, let $\gamma = \text{median}\{\|Z_i^* V^{(k-1)}\|_\infty\}$

real data study

Lung Cancer data: $X = [\ \ ]_{12625 \times 56}$

- gene expression levels of 12,625 genes
- 56 cases (56 patients)
- 4 types of lung cancer

Only a part of the genes regulate the type of the cancer, thus the singular vector should be sparse.

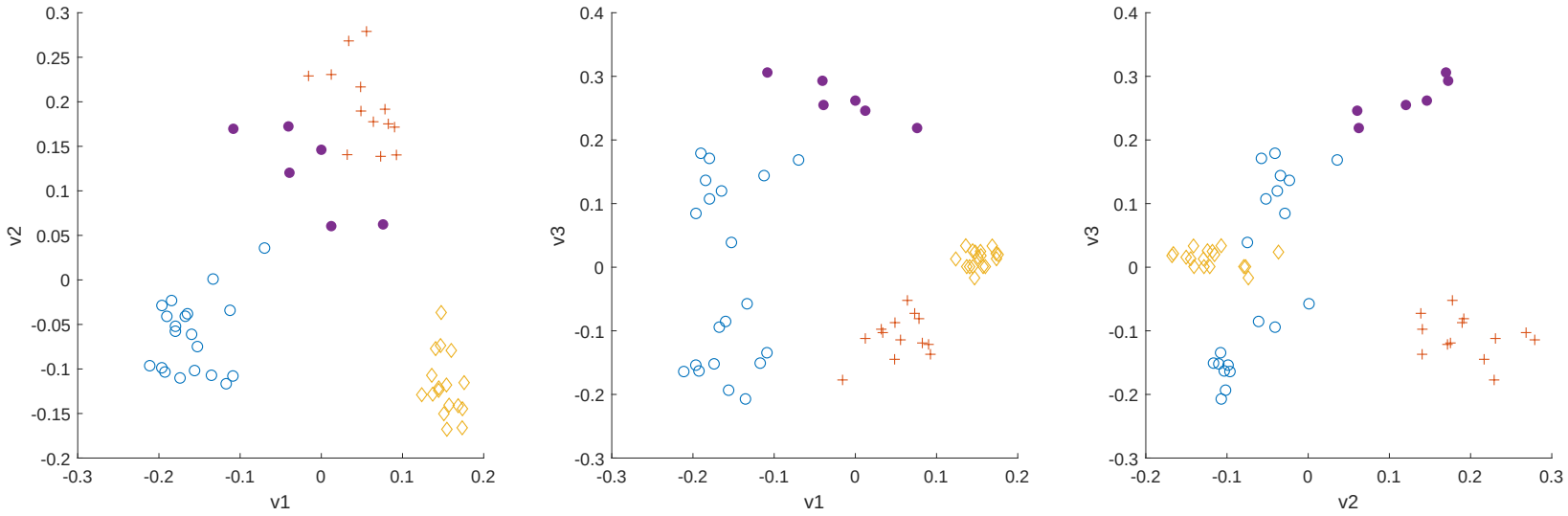


Figure 1: scatter plot of the right eigenvectors

real data study

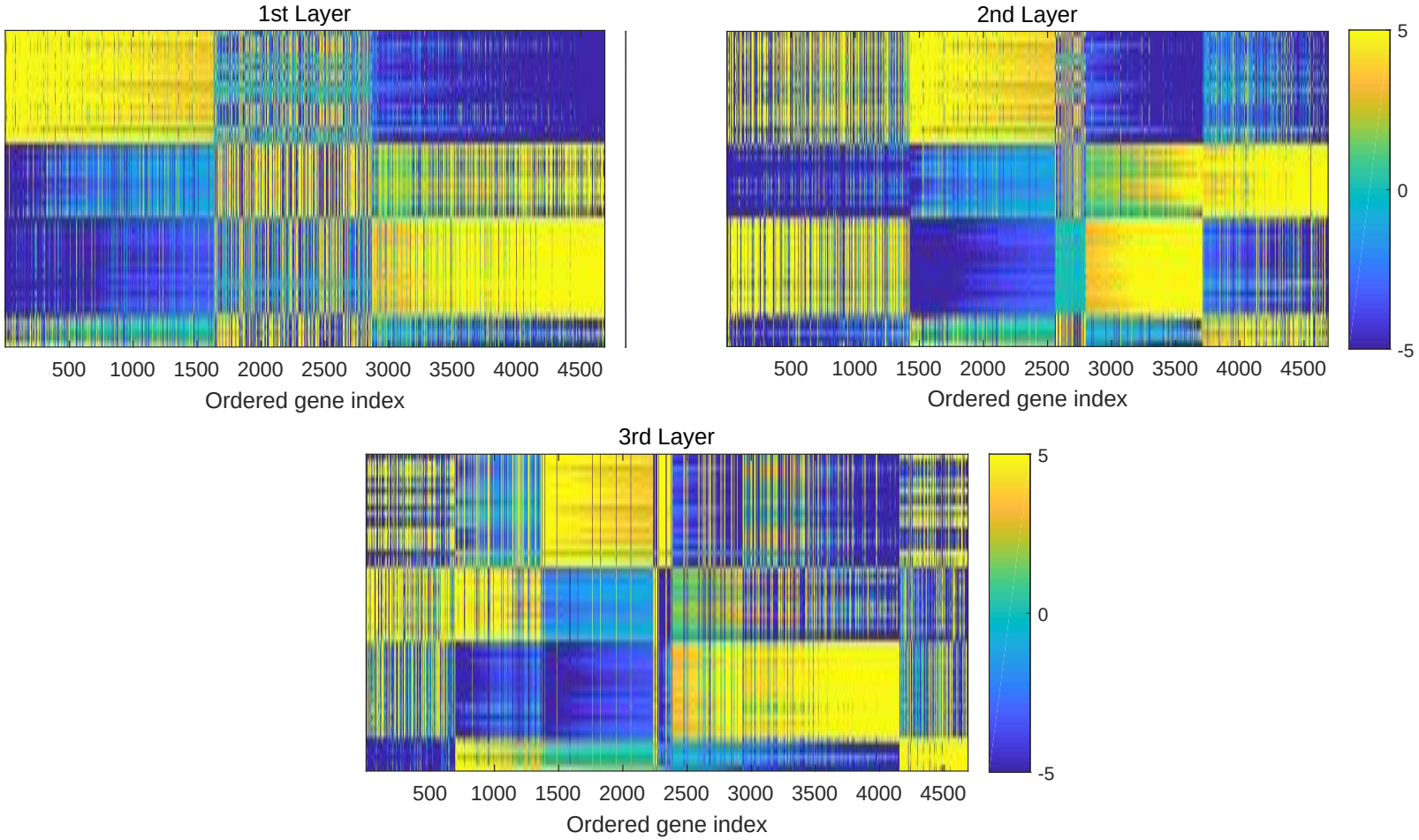


Figure 2: reconstructed rank-three approximation by FIT-SSVD

LSHM: more than 1 hour

FIT-SSVD: 3.7 seconds

Future work: In the real data study I use the hard threshold. In the literature, FIT-SSVD can use many other threshold precedure, e.g. soft threshold. is there an optimal one? produce the good estimate with fastest speed.

Reference

[1] Dan Yang, Zongming Ma, and Andreas Buja, A Spares Singular Value Decomposition Method for High-Dimensional Data, *Journal of Computational and Graphical Statistics* **23**(2014), 923–942.

[2] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron. Biclustering via sparse singular value decomposition, *Biometrics*, **66**(2010),1087-–1095.

Block Coordinate Descent Method for Cross-Modal Retrieval

Zihao Wang[1]

[1] College of Information Sciences and Technology
Penn State University

4 Dec 2018

Cross-modal Retrieval

Modal : Datatype

View : Each type of data is treated as a single view

Cross Modal : Returns relevant results of one modality in response to a query of another modality

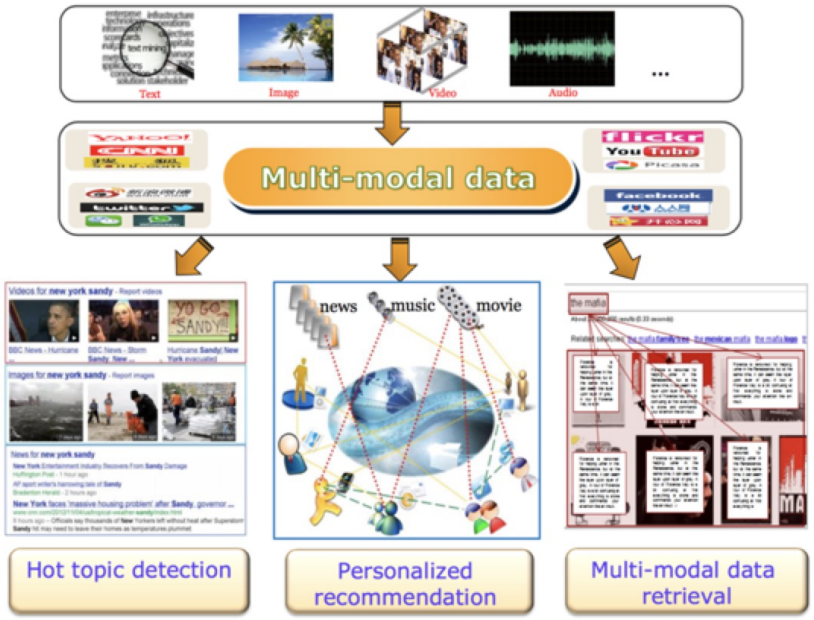


FIGURE – Multi-Modal data

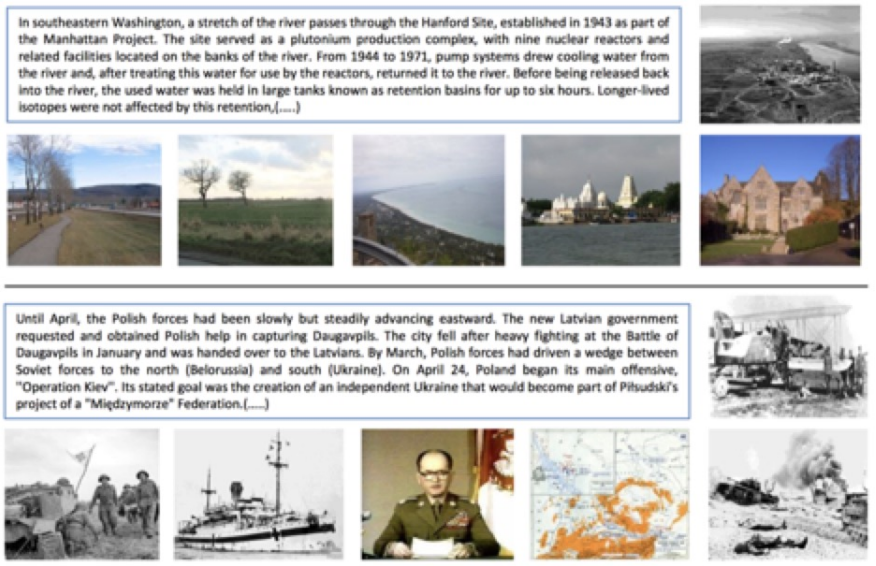


FIGURE – Cross-modal Retrieval

Definition and Problem Description

Training Data : $\mathcal{D} = \{((\mathbf{I}_1, \mathbf{T}_1), y_1), \ldots, ((\mathbf{I}_N, \mathbf{T}_N), y_N)\}$, imagine and text pair

y_i : label

N : sample size

Image Data : $\mathbf{I}_i = (\mathbf{I}_i^{(1)}, \ldots, \mathbf{I}_i^{(V_1)}), \mathbf{I}_i^{(\nu_1)} \in \mathbb{R}^{d_{\nu_1}}$

$\mathbf{I}_i^{(\nu_1)}$: the image feature vector in the view ν_1

d_{ν_1} : the dimensionality of the view ν_1

Text Data : $\mathbf{T}_i = (\mathbf{T}_i^{(1)}, \ldots, \mathbf{T}_i^{(V_2)}), \mathbf{T}_i^{(\nu_2)} \in \mathbb{R}^{d_{\nu_2}}$

$\mathbf{T}_i^{(\nu_2)}$: the text feature vector in the view ν_2

d_{ν_2} : the dimensionality of the view ν_2

Purpose : Build a function $f(\{(\mathbf{I}_i, \mathbf{T}_i)\}) : \mathcal{I} \to \mathcal{T}$ or $f(\{(\mathbf{I}_i, \mathbf{T}_i)\}) : \mathcal{T} \to \mathcal{I}$

Approach : Learn a discriminant function $F : \mathcal{I} \times \mathcal{T} \to \mathbb{R}$ to predict the optimal output $\mathbf{T}^*$

$$\mathbf{T}^* = f(\mathbf{I}; \mathbf{w}) = \arg\max_{\mathbf{T} \in \mathcal{T}} F(\mathbf{I}, \mathbf{T}; \mathbf{w}) \tag{1}$$

$\mathbf{w}$: the parameter needed to be learned

Empirical Risk

With the training data coming from P topics $\{\mathcal{D}_{p=1}^{P}\}$, we can write the regularized **empirical risk** $\mathcal{R}$ for cross-modal retrieval problem as :

$$\mathcal{R}(\{F_p\}_{p=1}^{P}) = \sum_{p=1}^{P}(\mathcal{L}_p(F_p(\mathbf{I}_p.\mathbf{T}_p; \mathbf{w}_p), y) + \lambda\Omega(\mathbf{w}_p)) \tag{2}$$

where $\Omega(\cdot)$ is a regularization term, and $\lambda > 0$ is the regularization hyper-parameter. The empirical loss of the training data from each topic $\mathcal{L}_p$ is

$$\mathcal{L}_p\left(F_p\left(\mathbf{I}_p, \mathbf{T}_p; \mathbf{w}_p\right), \mathbf{y}\right) = \sum_{i=1}^{N_p} \frac{1}{N_p} \ell\left(F_p(\mathbf{I}_{p,i}, \mathbf{T}_{p,i}; \mathbf{w}_p), y_{p,i}\right) \tag{3}$$

where ℓ is the prescribed loss function, here I use squared loss, and N_p is the sample number of the topic p

Multi-view Data Fusion through Tensor Modeling

Multi-view Data Fusion :

$$f\left(\left\{x^{(v)}\right\}_{v=1}^{V}\right)=\sum_{S=1}^{P}\sum_{i_1=0}^{d_1}\cdots\sum_{i_V=0}^{d_1}w_{i_1,\cdots,i_V,s}\left(\Pi_{v=1}^{V}\mathbf{z}_{i_\nu}^{(\nu)}\right) \tag{4}$$

$\mathbf{x}^{(v)}$ $\in \mathbb{R}^{d_v}$, the input multi-view data

$\mathbf{z}^{(\nu)}$ $= [1; \mathbf{x}^{(\nu)}]$

$\left\{w_{i_1}, \cdots, i_V, s\right\}$ the weight tensor to be learned, can be factorized into R factors as $[\![\Theta^{(1)}, \ldots, \Theta^{(V)}]\!]$

$\Theta^{(\nu)}$ $\in \mathbb{R}^{(d_v+1)\times R}$, the shared structure matrix for the v-th view

After CP factorization :

$$f\left(\left\{x^{(\nu)}\right\}_{\nu=1}^{V}\right)=\Pi_{\nu=1}^{V}\odot\left(z^{(\nu)T}\Theta^{(\nu)}\right)^{T} \tag{5}$$

Joint Optimization Problem

The joint optimization problem following the regularization formulation :

$$\min \mathcal{R}(\{\Theta^{(v)}\}) = \mathcal{L}_p(f(\{\mathbf{x}_I^{(v_1)}\}, \{\mathbf{x}_T^{(v_2)}\}), \mathbf{y}) + \lambda\Omega_\lambda(\{\Theta_I^{(v)}\}, \{\Theta_T^{(v)}\}) \tag{6}$$

- y the label
- $\mathcal{L}$ the loss function
- $\{\Theta_I^{(v)}\}, \{\Theta_T^{(v)}\}$ can be obtained by solving the problem
- Ω_λ the regularization terms, I use Frobenius norm

Though all parameters are convex, together Eqn(6) is non-convex with all the parameters.

Framework of Block Coordinate Descent

Algorithm 1 Block coordinate descent

Initialization: choose $(\mathbf{x}_1^0, \cdots, \mathbf{x}_s^0)$
for $k = 1, 2, \cdots$ **do**
  for $i = 1, 2, \cdots, s$ **do**
    update $\mathbf{x}_i^k$ with all other blocks fixed
  end for
  if stopping criterion is satisfied **then**
    return $(\mathbf{x}_1^k, \cdots, \mathbf{x}_s^k)$.
  end if
end for

Throughout iterations, each block x_i is updated by one of the three update schemes :

1. **Block minimization**
2. Block proximal descent
3. Block proximal linear

Alternating Block Coordinate Descent

STEP 1 : Fix α, and $\mathbf{\Theta}_T^{(v_2)}$, minimize $\mathbf{\Theta}_I^{(v_1)}$

$$\frac{\partial \mathcal{L}_p}{\partial f_p}\frac{\partial f_p}{\partial \mathbf{\Theta}_I^{(v_1)}} = \alpha_p \mathbf{Z}_{p,I}^{(v_1)}((\frac{\partial \mathcal{L}_p}{\partial f_p}) * \mathbf{\Pi}_{p,I}^{(-v_1)}) \tag{7}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}_p}{\partial f_p} &= \frac{1}{N_p}[\frac{\partial \ell_{p,1}}{f_{p,1}}, \cdots, \frac{\partial \ell_{p,N_p}}{f_{p,N_p}}]^T \in \mathbb{R}^{N_p} \\
\mathbf{\Pi}_{p,I}^{(-v_1)} &= [\boldsymbol{\pi}_{I,1}^{(-v_1)}, \cdots, \boldsymbol{\pi}_{I,N_p}^{(-v_1)}]^T \\
\boldsymbol{\pi}_I^{(-v_1)} &= \Pi_{v_1'=1, v_1' \neq v_1}^{V_1} * (\mathbf{z}_I^{(v_1')T} \mathbf{\Theta}_I^{(v_1')})^T \in \mathbb{R}^R
\end{aligned}$$

Similarly, for $\mathbf{\Theta}_T^{(v_2)}$

$$\frac{\partial \mathcal{L}_p}{\partial f_p}\frac{\partial f_p}{\partial \mathbf{\Theta}_T^{(v_2)}} = (1-\alpha_p)\mathbf{Z}_{p,T}^{(v_2)}((\frac{\partial \mathcal{L}_p}{\partial f_p}) * \mathbf{\Pi}_{p,T}^{(-v_2)}) \tag{8}$$

$$\mathbf{\Pi}_{p,T}^{(-v_2)} = [\boldsymbol{\pi}_{T,1}^{(-v_2)}, \cdots, \boldsymbol{\pi}_{T,N_p}^{(-v_2)}]^T$$

STEP 2 : Update α

$$\frac{\partial \mathcal{R}}{\partial \boldsymbol{\alpha}} = [(\frac{\partial \mathcal{L}_1}{\partial f_1})^T \mathbf{\Delta}_1, \cdots, (\frac{\partial \mathcal{L}}{\partial f})^T \mathbf{\Delta})] \tag{9}$$

$$\mathbf{\Delta}_p = \mathbf{f}_{p,I} - \mathbf{f}_{p,T}, \forall p \in [1:P], \mathbf{\Delta}_p \in \mathbb{R}^{N_p}$$

Experiments

Dataset : The NUS-WIDE dataset is a real-world image dataset created by Lab for Media Search in National University of Singapore. This dataset contains 81 topics.

TABLE – Imagine to Text Retrieval

Modle	mAP	Precision
JRL	0.5432	0.5010
SMFH	0.5974	0.4658
TM	0.7011	0.7467

TABLE – Text to Imagine Retrieval

Modle	mAP	Precision
JRL	0.5199	0.5218
SMFH	0.5596	0.4973
TM	0.7693	0.7748

mAP : the mean of the average precision scores for each query

$$mAP = \frac{\sum_{q=1}^{Q} \text{AveP}(q)}{Q} \tag{10}$$

Q : the number of queries