A Variational Bayes Approach to Robust Principal Component Analysis

Christopher Aicher
christopher.aicher@colorado.edu
Applied Math, University of Colorado Boulder
SFI REU 2013 Report, Mentor: Cristopher Moore

Abstract

We solve the *Robust Principal Component Analysis* problem: decomposing an observed matrix into a low-rank matrix plus a sparse matrix. Unlike alternative methods that approximate this ℓ_0 objective with an ℓ_1 objective and solve a *convex optimization problem*, we develop a corresponding generative model and solve a *statistical inference problem*. The main advantages of this approach is its ability to incorporate additional prior information when it exists and cope with missing data where it does not. Using a variational Bayes approach, we develop an algorithm the low-rank and sparse matrices. Finally, we test and compare our Bayesian model with alternative approaches on both synthetic and real-world examples.

1 Introduction

The increase in the size and dimension of datasets has lead to the development of data-compression and dimension-reduction techniques. One common and well-studied method is Principal Component Analysis (PCA), which attempts dimension-reduction by projecting each data point onto its 'principal components'. Classic PCA finds the 'principal components' that minimize error with respect to independently identically distributed Gaussian noise. However in practice, a sparse number of observations may be corrupted outliers and should be ignored when inferring the 'principal components'. This variant of PCA is called Robust PCA and has recieved renewed focus in the last decade from compressed sensing.

We first review how to solve PCA and why Robust PCA, although similar, is trickier. Then we describe our Bayesian model and our variational algorihm for performing inference. Lastly we describe performance on both synthetic and real-world examples.

1.1 Principal Component Analysis

Suppose we have m data points M_j each with n attributes $M_j \in \mathbb{R}^n$. Treating the dataset as a matrix $M \in \mathbb{R}^{n \times m}$, PCA attempts to find an orthogonal transfomation $D^T \in \ \times \ltimes$ of the columns M_j, such that the resulting matrix columns $A_j \in \mathbb{R}^d$ are linearly independent while still maintaining maximal variance. Here D is called a dictionary of M and A are the 'principal components'. By selecting $d < \min(n, m)$, we reduce the dimension of the data, but can only approximately recover M with the product DA.

Note that PCA approximates our dataset M with a matrix DA of rank d. The motivation for this in practice is that although our observed data points each have n attributes, there are d underlying factors that really matter. Once we have the appropriate dictionary D, we can reduce the dimension of each data point from n to d.

To determine the dictionary and principal components for a fixed dimension d, we minimize the elementwise ℓ_2 distance between M and its approximation DA. This classic problem, is solved by finding the Singular Value Decomposition (SVD) of the matrix. Recall that the SVD of a matrix M is the decomposition

$$M = \tilde{U}\tilde{\Sigma}\tilde{V}^T \ ,$$

Where $\tilde{U}, \tilde{V}$ are orthogonal matrices and $\tilde{\Sigma}$ is a diagonal matrix of singular values. It is a well known property of that the closest rank d matrix to M is the matrix $U\Sigma_d V^T$ formed by truncating $\tilde{\Sigma}$ to the first d singular values. That is

$$U\Sigma_d V^T = \arg\min_L \|M - L\|_F \ \text{ such that } \ \operatorname{rank} L = d \ .$$

In this case our dictionary is $D = U$ and our principal components are $A = \Sigma_d V^T$. Therefore finding the 'principal components' of M only require calculating the SVD of M.

1.2 Robust PCA

Unfortunately the SVD does not extend to Robust PCA. Instead of approximating M with a rank d matrix L, we

decompose M into a low rank matrix L plus a sparse matrix S. The motivation for introducing S is that there may be some sparse noise contamination, which we should ignore. Therefore to find the principal components of M, we apply SVD to the clean low-rank matrix L.

To select L and S we minimize the objective function

$$\arg\min_{L,S} \|M - S - L\|_F + \lambda \|S\|_0 \text{ such that } \operatorname{rank} L = d \ ,$$

where $\lambda > 0$ is a tuning parameter. The first term favors L be weakly close to $M - S$ and the second term favors S is sparse. This objective function is difficult to minimize because both rank and the ℓ_0 norm are non-convex functions.

One common approach in practice is to replace rank and ℓ_0 norm with the $\|\cdot\|_*$ nuclear and ℓ_1 norms respectively, relaxing our task into a convex optimiziation problem. Examples of this are [Candes 2008, Zhou T. 2011, Zhou X. 2011, Bach 2011].

Instead of convexifying the problem, our approach is to recast our objective function as a stastical inference question and apply tools from Bayesian inference. Prior work on applying Bayesian method to this task are [Salakhutdinov 2008, Salakhutdinov 2008b, Ding 2011, Lakshminarayanan 2011], however our model is slightly different from prior work in how we incorporate and model sparse noise as well as using variational Bayes instead of MCMC.

An advantage of this Bayesian approach is that it allows us to solve real-world cases of extra or missing data without resorting to heurestical methods and hacks. Furthermore, although in most applications a drawback of Bayesian statistics is that we introduce subjectivity through the prior distribution, our application treats Bayesian statistics as a tool for getting a matrix factorization and therefore we do want to introduce information through the prior such as having S be sparse. Of course, if we do not have prior knowledge, then we can always use non-informative priors as well.

2 Bayesian Model

Our model assumes the observed matrix M is the sum of a low rank matrix L, a sparse noise matrix S, and some small background noise ϵ.

Let n, m, d be fixed integers. Our model assumes the matrix $M \in \mathbb{R}^{n \times m}$ is the observation of a random matrix that can be decomposed via

$$M = UV^T + Z^* \circ B + \epsilon \ , \tag{1}$$

where $\circ$ denotes element-wise multiplication (Haydamare Product).

Our low rank matrix is $L = UV^T$. We restrict U to be an n by d matrix and V to be an m by d matrix so that the rank of L less than or equal to d. In what follows let $i = 1, \ldots, n$ be the index of the columns of U and $j = 1, \ldots, m$ be the index of the columns of V. Note that U_i and V_j are d dimensional vectors.

Our sparse matrix is $S = Z^* \circ B$. We set B to be a sparse binary matrix and let Z^* be free. For numerical reasons, we treat Z^* as a very diffuse Gaussian matrix. To induce sparsity in S, we select a prior on B such that it is sparse: most entries will be zero.

Finally $\epsilon(\sigma^2)$ is a small Gaussian noise term. Since we want only small noise, we make the prior on its variance small compared with the variance of Z^*.

Instead of solving (1), it is more numerical convenient to solve the identical problem

$$M = UV^T + Z \circ B + \epsilon \circ (1 - B) \ , \tag{2}$$

where we incorporate both Z^* and ϵ into Z.

The main feature of this model is the ability for different elements ij to have different variances depending on B_{ij}. Therefore some terms can be very far off, while still being constrained by most. The idea of using a Bernoulli-Gaussian distribution to model outliers is inspired from [Krzakala 2012, Ding 2011] and the contamination model in robust statistics.

Ultimately, we use the posterior of U, V to estimate L and apply SVD to obtain its principal components. We can then obtain a noisy estimate of S from $(M - L) \circ B$. Ideally, S contains only a sparse number of large terms and ϵ contains the small negligible terms.

3 Posterior Inference

To infer U, V, B, Z, ϵ, we approximate the posterior distribution π^* with a factorizable distribution q. This a variational approach selects the distribution q 'closest' to the posterior π^* in the sense of Kullback-Leibler (KL) Divergence. By parameterizing q, we convert this inference scheme back into an objective maximiziation problem. Finally after selecting a distribution to approximate the posterior, we then take expectations of U, V, B, Z, ϵ to estimate them.

Recall that the posterior distribution π^* is proportional to the product of the likelihood $\mathcal{L}$ and the prior π.

3.0.1 Prior

For the prior distribution, we model each column of U and V as an independent Gaussian with mean μ_0 or ν_0

and a large variance σ_0^2. We treat each element of B as an independent Bernoulli random variable with probability b_0. For the matrices Z and ϵ, we fix the mean to zero and estimate the variance by modeling the precision $\tau = 1/\sigma^2$ as Gamma random variables. All together, our prior distribution is

$$\begin{aligned}\pi =& \Pr(U, V, B, \tau_Z, \tau_\epsilon) \\ =& \prod_i \mathcal{N}(U_i|\mu_0, \sigma_0^2 \mathbb{I}_d) \times \prod_j \mathcal{N}(V_j|\nu_0, \sigma_0^2 \mathbb{I}_d) \\ & \times \prod_{ij} b_0^{B_{ij}} (1-b_0)^{1-B_{ij}} \times \Gamma(\tau_Z|\alpha_{Z0}, \beta_{Z0}) \\ & \times \Gamma(\tau_\epsilon|\alpha_{\epsilon 0}, \beta_{\epsilon 0})\end{aligned}$$

3.0.2 Likelihood

The likelihood function of U, V, B, Z, ϵ given M is

$$\begin{aligned}\mathcal{L} =& \Pr(M \mid U, V, B, \tau_Z, \tau_\epsilon) \\ =& \prod_{ij} \mathcal{N}\left(M_{ij} - U_i \cdot V_j \mid \tau_Z\right)^{B_{ij}} \\ & \times \prod_{ij} \mathcal{N}\left(M_{ij} - U_i \cdot V_j \mid \sigma_\epsilon^2\right)^{1-B_{ij}} \quad ,\end{aligned}$$

and the loglikelihood function is

$$\begin{aligned}\log \mathcal{L} =& -\sum_{ij} \frac{B_{ij}}{2} \left[\tau_Z \left(M_{ij} - U_i \cdot V_j\right)^2 + \log \sigma_Z^2\right] \\ & - \sum_{ij} \frac{1-B_{ij}}{2} \left[\tau_\epsilon \left(M_{ij} - U_i \cdot V_j\right)^2 + \log \sigma_\epsilon^2\right] \quad .\end{aligned}$$

3.1 Naive VB

Calculating the posterior distribution from the product of the prior and the likelihood function is difficult because integrating for the normalizing constant is intractable. Instead we approximate the posterior over $U, V, B, \tau_Z, \tau_\epsilon$ using a naive variational Bayes approach. We approximate the posterior distribution π^* with q, the factorized product of $q(U)q(V)q(B)q(\tau_Z)q(\tau_\epsilon)$. In the appendix, I summarize some of the effects of using slightly more accurate variational methods (i.e. $q = q(U, V)$ instead of assuming $q(U)q(V)$).

To find q 'closest' to π^* recall that

$$\log \Pr(M) = \mathcal{G}(q) + D_{KL}(q||\pi^*) \quad ,$$

thus minimizing $D_{KL}(q||\pi^*)$ is equivalent to maximizing $\mathcal{G}$

$$\mathcal{G}(q) = \mathbb{E}_q(\log \mathcal{L}) - D_{KL}(q||\pi) \quad .$$

The restrictions on our approximation q are as follows.

The distributions $q(U), q(V)$ are multivariate normals with parameters μ_i and ν_j for the mean of U_i and V_j respectively and Σ_i and Σ_j for their respective covariance matrices. For now, we assume that the covariance between U_i and V_j is zero.

Since B is a binary matrix, we approximate it with a matrix of Bernoullis each with parameter b_{ij}.

Finally since the Gamma distribution is a conjugate prior for precision our factorized distributions for τ_Z and τ_ϵ are Gamma as well with parameters α_Z, β_Z and α_ϵ and β_ϵ.

Under this model, the objective function can be written as

$$\begin{aligned}\mathcal{G} =& \sum_{ij} \Big[- \frac{b_{ij}\mathbb{E}(\tau_Z) + (1-b_{ij})\mathbb{E}(\tau_\epsilon)}{2} \times \\ & \left(M_{ij}^2 - 2M_{ij}(\mu_i \cdot \nu_j) + \left(\Sigma^i + \mu_i \mu_i^T\right) \circ \left(\Sigma^j + \nu_j \nu_j^T\right)\right) \Big] \\ & + \sum_{ij} \left[\frac{b_{ij}}{2} \mathbb{E}(\log \tau_Z) + \frac{1-b_{ij}}{2} \mathbb{E}(\log \tau_\epsilon) \right] - D_{KL}(q||\pi) \quad .\end{aligned}$$

Note that selecting our optimal approximation q is equivalent to maximizing $\mathcal{G}$ with respect to the parameters of q. We maximize $\mathcal{G}$ by finding the critical point from setting all derivatives equal to zero an iteratively updating the parameters, discussed in the sections below.

3.2 Updating U, V

To find the updates for U, V we maximize $\mathcal{G}$ with respect to the parameters μ, ν, Σ of q. For notational convenience let

$$T_{ij} = \begin{cases} \mathbb{E}(\tau_Z) b_{ij} + \mathbb{E}(\tau_\epsilon)(1-b_{ij}) & \text{if Observed} \\ 0 & \text{if Missing} \end{cases} \quad .$$

be the expected precision for ij-th element of M. Then the updates simplfy to

$$\begin{aligned}\mu_I &= \Sigma^I \left[(T_{I\cdot} \circ M_{I\cdot}) \cdot \nu + \frac{\mu_0}{\sigma_0^2} \right] \\ \nu_J &= \Sigma^J \left[(T_{\cdot J} \circ M_{\cdot J}) \cdot \mu + \frac{\nu_0}{\sigma_0^2} \right] \\ \Sigma^I &= \left[\sum_j \left(\Sigma^j + \nu_j \nu_j^T\right) T_{Ij} + \mathbb{I}_d/\sigma_0^2 \right]^{-1} \\ \Sigma^J &= \left[\sum_i \left(\Sigma^i + \mu_i \mu_i^T\right) T_{iJ} + \mathbb{I}_d/\sigma_0^2 \right]^{-1} \quad .\end{aligned}$$

The complete derivation is in the appendix.

Note that these updates are a bit computationally intensive because we need to invert $n + m$ d by d matrices to compute Σ^I and Σ^J for all I and J.

3.3 Updating B

Since B is a binary matrix, each element B_{ij} has an associated parameter b_{ij}, the probability of observing a 1. Assuming that our prior belief about observing a 1 is b_0, then our objective function $\mathcal{G}$ becomes

$$\begin{aligned}\mathcal{G} = &- \sum_{ij} T_{ij} \frac{\langle S_{ij} \rangle}{2} \\ &+ \frac{b_{ij}}{2} \mathbb{E} \log \sigma_Z^2 + \frac{1 - b_{ij}}{2} \mathbb{E} \log \sigma_\epsilon^2 \\ &- b_{ij} \log \frac{b_{ij}}{b_0} - (1 - b_{ij}) \log \frac{1 - b_{ij}}{1 - b_0} \quad ,\end{aligned}$$

where $\langle S_{ij} \rangle$ is the square deviation

$$\langle S_{ij} \rangle = \mathbb{E}_q \left[(M_{ij} - U_i \cdot V_j)^2 \right] \quad .$$

Taking a derivative with respect to b_{IJ} and solving for b_{ij} gives us

$$\begin{aligned}\text{Logit}(b_{ij}) = \log \frac{b_{ij}}{1 - b_{ij}} &= \log \frac{b_0}{1 - b_0} \\ &+ \frac{1}{2} \mathbb{E} \left(\log \frac{\tau_Z}{\tau_\epsilon} \right) + \mathbb{E} \left(\tau_\epsilon - \tau_Z \right) \frac{\langle S_{ij} \rangle}{2} \quad ,\end{aligned}$$

which we can easily solve for b_{ij} using the Expit (inverse Logit) function. Recall that negative Logit values correspond to $b_{ij} < 1/2$ and positive Logit values implies $b_{ij} > 1/2$.

The first term corresponds our prior belief about b_{ij}.

The second term favors b_{ij} close to zero as Z is more spread out compared with ϵ thus $\tau_Z < \tau_\epsilon$ and the second term is negative.

The third term corresponds to how far our observation M_{ij} is from our expected value $\mathbb{E}(U_i \cdot V_j)$. This will typically favor b_{ij} close to one as $\tau_Z < \tau_\epsilon$ implies the third term is positive.

Therefore the tradeoff between the second and third term keeps B from degenerating into all 0 or all 1.

3.3.1 Calculating $\langle S_{ij} \rangle$

Above, we skipped over the calculation of $\langle S_{ij} \rangle$. It turns out calculating this value in our model is a bit complicated.

The simple, computationally-nice thing to do is to use the approximation

$$\begin{aligned}\langle S_{ij} \rangle &= \mathbb{E}_q \left[(M_{ij} - U_i \cdot V_j)^2 \right] \\ &\approx \mathbb{E}_q \left[(M_{ij} - U_i \cdot V_j) \right]^2 = (M_{ij} - \mu_i \cdot \nu_j)^2 \quad ,\end{aligned}$$

which is off by the variance of $M_{ij} - U_i \cdot V_j$

The mathematically correct equation is more complicated.

$$\begin{aligned}\langle S_{ij} \rangle &= \mathbb{E} \left((M_{ij} - U_i \cdot V_j)^2 \right) \\ &= M_{ij}^2 - 2 M_{ij} \mathbb{E}(U_i \cdot V_j) + \mathbb{E} \left((U_i \cdot V_j)^2 \right) \quad .\end{aligned}$$

Then recall expected value of $U_i \cdot V_j$ is

$$\mathbb{E} \left(U_i \cdot V_j \right) = \mu_i \cdot \nu_j \quad ,$$

and the expected value of $(U_i \cdot V_j)^2$ is

$$\begin{aligned}\mathbb{E} \left((U_i \cdot V_j)^2 \right) = (\mu_i \cdot \nu_j)^2 &+ \text{Tr} \left(\Sigma^i \Sigma^j \right) \\ &+ \text{Tr} \left(\Sigma^i \cdot \nu_j \nu_j^T \right) + \text{Tr} \left(\Sigma^j \cdot \mu_i \mu_i^T \right) \quad ,\end{aligned}$$

The derivation involves calculating the fourth-moments of $q(U, V)$ outlined in the appendix. In practice, we have found they give similar results and the approximation method is faster.

3.4 Updating τ_Z, τ_ϵ

The factorized distributions q_z, q_ϵ for precision are the Gamma distributions. Let the parameters for τ_Z be α_Z, β_Z and the parameters for τ_ϵ be $\alpha_\epsilon, \beta_\epsilon$. Recall that for the Gamma distribution the expected values

$$\mathbb{E}_q(\tau) = \frac{\alpha}{\beta}$$

and

$$\mathbb{E}_q(\log \tau) = \psi(\alpha) - \log(\beta) \quad .$$

For a refresher see the appendix.

Setting the derivative $\frac{\partial \mathcal{G}}{\partial \alpha}$ and $\frac{\partial \mathcal{G}}{\partial \beta}$ to zero gives the following update equations,

$$\alpha_Z = \alpha_{Z0} + \sum_{ij} \frac{b_{ij}}{2}$$

$$\beta_Z = \beta_{Z0} + \sum_{ij} \frac{b_{ij} \langle S_{ij} \rangle}{2}$$

$$\alpha_\epsilon = \alpha_{\epsilon 0} + \sum_{ij} \frac{1 - b_{ij}}{2}$$

$$\beta_\epsilon = \beta_{\epsilon 0} + \sum_{ij} \frac{(1 - b_{ij}) \langle S_{ij} \rangle}{2} \quad ,$$

where α_{Z0}, β_{Z0} are the prior parameters of τ_Z selected such that α_{Z0}/β_{Z0} is large, and $\alpha_{\epsilon 0}, \beta_{\epsilon 0}$ are the prior parameters of τ_ϵ selected such that $\alpha_{\epsilon 0}/\beta_{\epsilon 0}$ is small.

One can think of β as the pseudo-"sum of square errors" and α as the pseudo-"number of observations".

3.5 The Complete Algorithm

Putting all the updates together forms the following iterative RPCA algorithm Algorithm 1.

Algorithm 1 Robust PCA via Factorized VB

Input: Observed Data M, Parameters d, b_0 σ^2
Initialize μ, ν randomly.
Set Σ^i, Σ^j to $\mathbb{I}_d$ for all i, j.
Set B to $b_0 \cdot 1$.
repeat
  Main Loop
  repeat
    Calculate the Expected Precision T,

$$T = \tau_z B + \tau_\epsilon (1 - B)$$

    U, V Loop:
    for $i = 1$ to n **do**
      Update Σ^i:

$$\Sigma^i = \left[\sum_j T_{ij} \left(\Sigma^j + \nu_j \nu_j^T \right) + \mathbb{I}_d / \sigma_0^2 \right]^{-1}$$

      Update μ_i:

$$\mu_i = \Sigma^i \left(T_{i\cdot} \circ M_{i\cdot} \cdot \nu + \frac{\mu_0}{\sigma_0^2} \right)$$

    end for
    for $j = 1$ to m **do**
      Update Σ^j:

$$\Sigma^j = \left[\sum_i T_{ij} \left(\Sigma^i + \mu_i \mu_i^T \right) + \mathbb{I}_d / \sigma_0^2 \right]^{-1}$$

      Update ν_j:

$$\nu_j = \Sigma^j \left(T_{\cdot j} \circ M_{\cdot j} \cdot \mu + \frac{\nu_0}{\sigma_0^2} \right)$$

    end for
  until $\mu_i, \nu_i, \Sigma^i, \Sigma^j$ converge
  for $i = 1$ to n, $j = 1$ to m **do**
    Calculate $\langle S_{ij} \rangle$

$$\langle S \rangle_{ij} \approx (M_{ij} - \mu_i \cdot \nu_j)^2$$

  end for

Algorithm 1 Robust PCA (Cont.)

  for $i = 1$ to n, $j = 1$ to m **do**
    Update B_{ij}

$$B_{ij} = \text{Logit}^{-1} \Big(\text{Logit}(b_0) + \log \left(\tau_Z / \tau_\epsilon \right) / 2 + \left(\tau_\epsilon - \tau_Z \right) \langle S_{ij} \rangle / 2 \Big)$$

  end for
  Update τ_Z

$$\alpha_Z = \alpha_0 + \sum_{ij} B_{ij} / 2$$
$$\beta_Z = \beta_0 + \sum_{ij} B_{ij} \langle S_{ij} \rangle / 2$$
$$\tau_Z = \alpha_Z / \beta_Z$$

  Update τ_ϵ

$$\alpha_\epsilon = \alpha_0 + \sum_{ij} (1 - B_{ij}) / 2$$
$$\beta_\epsilon = \beta_0 + \sum_{ij} (1 - B_{ij}) \langle S_{ij} \rangle / 2$$
$$\tau_\epsilon = \alpha_\epsilon / \beta_\epsilon$$

until All parameters converge
return $\mu, \nu, \Sigma^i, \Sigma^j, B, \tau_Z, \tau_\epsilon$

4 Results

To get a better understanding of our algorithm's performance we test it on a wide varity of synthetic data. Our synthetic experiments suggest that our algorithm is quite robust to misspecified parameters and missing data.

Finally, to show some applications of our algorithm to real data we apply it to foreground detection and predicting movie and joke ratings.

4.1 Synthetic Data

For our sythetic data experiments we tested a collection of RPCA algorithms for estimating the true low rank matrix L against various combinations of parameters (for both the inference and the noise).

The algorithms we compared against were:

- Noise, we predict the observed matrix as our guess for the truth ignoring rank information.
- Singular Value Decomposition (SVD) a classic baseline algorithm.

- Bayesian PCA (VBPCA) our algorithm with $B = 0$ for all entries (no sparse noise).
- Bayesian RPCA (RPCA) our algortihm with non-informative priors unless otherwise specified.
- Bayesian RPCA (VBLR) an alternative Bayesian algorithm with a slightly different sparse noise model [Babacan 2011]. Unfortunately, we were not able to get this algorithm to work appropriately. With the default setting, the algorithm returned random noise in most cases except $\sigma_\epsilon^2 = 0$.
- Go Decomposition (GoDec) an RPCA algorithm that maximizes the convex objective function by alternating approximating L and S [Zhou T. 2011].
- Augmented Lagrange Method (ALM) a convex-optimization method for RPCA [Lin 2010]. Unfortunately, we were not able to get this algorithm to work appropriately. With the default settings, the algorithm appeared to return the observed matrix in the majority of cases.

Therefore we are really comparing our RPCA algorithm with GoDec and SVD.

As for the parameters in our datasets, we explored the parameter space by varying one parameter about a fixed point. An example of a dataset and inference about this fixed point is shown in Figure 2.

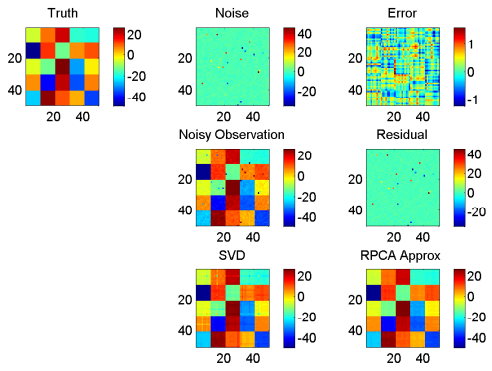


Figure 2: Synthetic Data Example

Here the rank is $k = 5$, the sparse noise is Gaussian with variance $\sigma_Z^2 = 1000$ affecting the fraction $p_{sparse} = 0.01$ of the entries, the universal noise is Gaussian with variance $\sigma_\epsilon^2 = 1$, the guess of rank is $k_{infer} = 5$, the guess of the sparse fraction is $b_0 = 0.01$.

As far as sensitivity is concerned, it appears that our algorithm does not need strong knowledge of $\tau_Z, \tau_\epsilon, b_0$. This is because our method learns the appropriate value of τ_Z, τ_ϵ from the data.

On the other hand, our algorithm suffers when the rank k is under-specified or grossly over-specified. However when the inferred rank is only slightly over-specified, our algorithm performs nicely. This is to be expected as when k gets large enough to capture all the sparse noise, we overfit and when k is too small, it is impossible to accurately recover L.

4.2 Foreground Detection

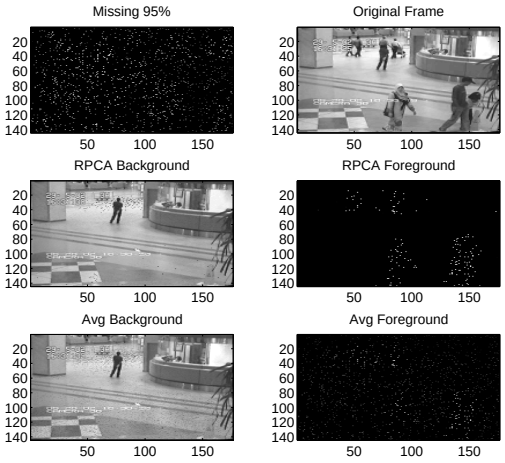


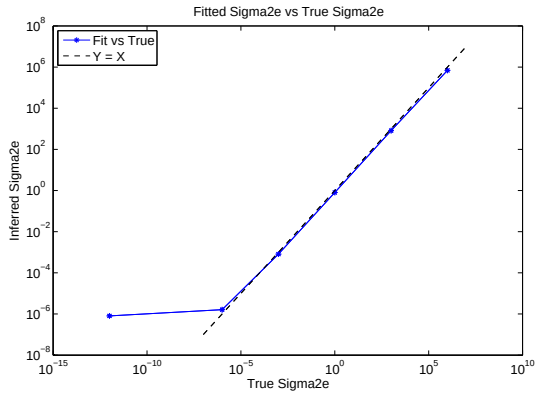
Figure 3: Foreground Detection Example: Frame 20

One application of our method to real data is detecting the foreground from the background in camera footage. Here the sparse noise S is the foreground and the low-rank matrix L is the background and our matrix is the collection of 200 image frames. An advantage our algorithm has is it can perform in the presense of missing data. Its clear that our algorithm recovers a more sharp and clear representation of the background than the averaging heuristic, which suffers because it does not distinguish the foreground noise.

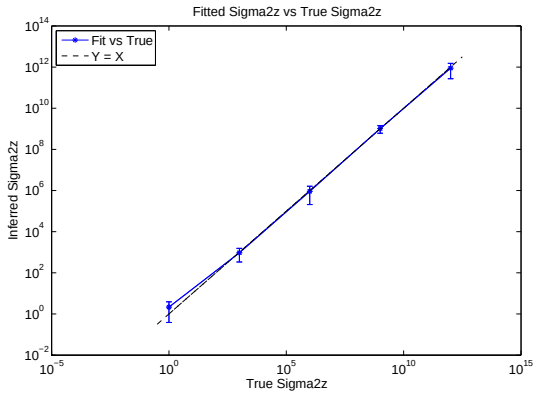
4.3 Rating Prediction

Another application of our method is to the matrix completion problem in prediction ratings of movies and jokes. Here we are given only a fraction of the entries in a matrix and our goal is to predict the unobserved entries. Our method exploits the belief that the matrix is low rank like Bayesian PCA, but in addition we allow for outlier. For example, most people have a genre preference, but there may be a few outlier ratings that are due to recommendations or other mechanisms.

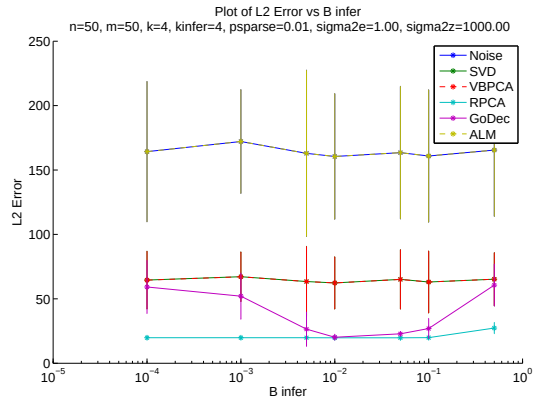
The two datasets we used were the Movie Lens 100k dataset from [GroupLens] and the Jester dataset from [Goldberg 2001]. To test our algorithm's predictive power



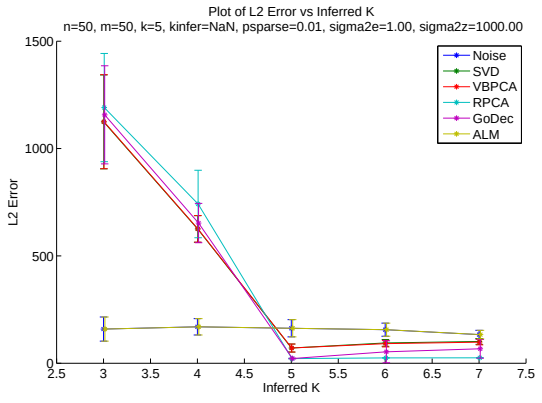
(a) True vs Inferred σ_ϵ^2



(b) True vs Inferred σ_z^2



(c) Error vs Prior on B



(d) Error vs Fitted Rank k

Figure 1: Synthetic Results (a,b) RPCA's inferred belief about σ_ϵ^2 and σ_Z^2 versus the actual value. (c) RPCA's sensitivity to misspecified B. (d) RPCA's sensitivity to misspecified k.

we used cross-validation: hiding 20% of our original data as a test set and training our method on the other 80%. We compared our method to simple heuristics such as guessing the average of all ratings(global averaging), assigning each movie an average rating (row averaging) and assigning each user an average rating (column averaging).

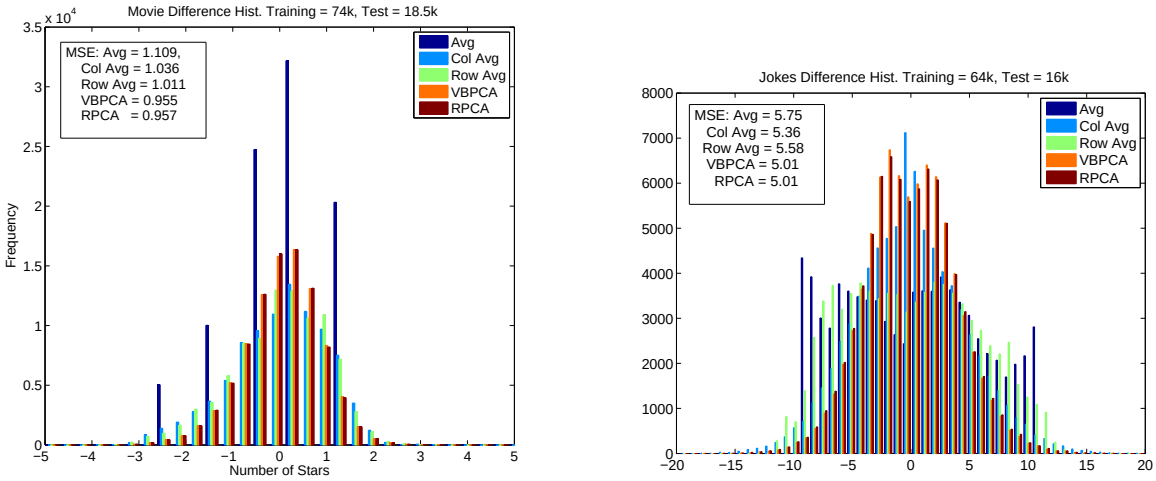
An example of our performance on both datasets is shown in Figure 4. We outperform the simple heuristics, but not by a practically significant amount. In addition, Bayesian RPCA performs on a comparable level to Bayesian PCA suggesting that perhaps the outliers are either not sparse or the ϵ noise is too large for RPCA to work.

5 Discussion

We have developed a variational Bayes algorithm that estimates the low-rank approximation of a matrix for Robust PCA. We note that our method is fairly robust to misspecification of the noise level τ_Z, τ_ϵ, the fraction of sparse noise b_0, and missing data, because of its Bayesian roots. In addition, our method is slightly robust for small overestimation of k. However more work should be done in comparing our method to alternatives and exploring the phase transition boundary of the parameter space where we can no longer recover L. This especially true for comparing the method to matrix completion problems.

Another application area to look into is Robust Multivariate Regression. This may also include modifying the $M-L-S$ distance function to weight different attributes by their relative importance. Instead of each attribute of M_j having equal importance, we may prioritize some over others.

Finally although the higher order approximation methods we tried did not work out, there are many more tricks and approximations still to try. Its possible that these higher order approximations may have a better phase transition boundary, but finding algorithms that are efficient will also be a challenge.



(a) Movie Lens

(b) Jester

Figure 4: Rating Prediction Results. The table in the upper right corners are the MSE values over multiple trials.

Acknowledgements

I thank the Santa Fe Institute's Summer REU program for sponsoring this summer research project, especially my advisor Cristopher Moore.

References

[Salakhutdinov 2008] Salakhutdinov, R. & Mnih, A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. ICML (2008).

[Salakhutdinov 2008b] Salakhutdinov, R. & Mnih, A. Probabilistic matrix factorization. Advances in neural information processing systems 20, 12571264 (2008).

[Ding 2011] Xinghao Ding, Lihan He & Carin, L. Bayesian Robust Principal Component Analysis. IEEE Transactions on Image Processing 20, 34193430 (2011).

[Plan 2011] Plan, Y. Compressed sensing, sparse approximation, and low-rank matrix estimation. (2011).

[Goldberg 2001] Goldberg, K., Roeder, T., Gupta, D. & Perkins, C. Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4, 133151 (2001).

[Candes 2008] Candes, E. J. & Recht, B. Exact Matrix Completion via Convex Optimization. (2008).

[Seeger 2005] Seeger, M. Expectation propagation for exponential families. Technical report, University of California at Berkeley, 2005.

[Zhou T. 2011] Zhou, T., UTS, S., Tao, D. & EDU, U. GoDec: Randomized Low-rank & Sparse Matrix Decomposition in Noisy Case. 2011

[Zhou X. 2011] Zhou, X., Yang, C. & Yu, W. Moving Object Detection by Detecting Contiguous Outliers in the Low-Rank Representation. (2011).

[Krzakala 2012] Krzakala, F., Mézard, M., Sausset, F., Sun, Y. & Zdeborová, L. Probabilistic Reconstruction in Compressed Sensing: Algorithms, Phase Diagrams, and Threshold Achieving Matrices. (2012).

[Lin 2010] Lin, Z., Chen, M. & Ma, Y. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. (2010).

[Lakshminarayanan 2011] Lakshminarayanan, B., Bouchard, G. & Archambeau, C. Robust bayesian matrix factorisation. in International Conference on Artificial Intelligence and Statistics 425433 (2011).

[Babacan 2011] Babacan, S. D., Luessi, M., Molina, R. & Katsaggelos, A. K. Sparse Bayesian Methods for Low-Rank Matrix Estimation. (2011).

[Bach 2011] Bach, F. Structured Sparse Methods for Matrix Factorization. (2011). at

[GroupLens] Movie Lens Recommender System Dataset. GroupLens Research. http://www.grouplens.org/node/12

[Yedidia 2003] Yedidia, J. S., Freeman, W. T. & Weiss, Y. Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium 8, 236239 (2003).

Appendix

A Distribution Notation

Gamma Distribution

A random variable X in $[0, \infty)$ is a Gamma distribution with parameters α, β in $(0, \infty)$, if its probability density function has the form

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x) \quad ,$$

for all x in $\mathbb{R}$. The expected value of x is α/β and the variance is α/β^2.

Multivariate Gaussian or Normal Distribution

A random vector X in $\mathbb{R}^d$ is a d-dimensional Multivariate Gaussian with mean μ in $\mathbb{R}^d$ and covariance matrix Σ, a positive definite d by d matrix, if its probability density function takes the form

$$\mathcal{N}(x|\mu, \Sigma) = \left(\frac{1}{2\pi}\right)^{d/2} |\Sigma|^{-d/2} \times \exp\left[(x-\mu)^T \Sigma^{-1} (x-\mu)/2\right] \quad ,$$

for all x in $\mathbb{R}^d$. The expected value of x is μ and the covariance matrix is Σ.

Fourth Moment of Non-Central Multivariate Normal

Isserlis' Theorem states that for a Multivariate Normal r.v. with mean zero we have

$$\begin{aligned}&\mathbb{E}[X_1X_2X_3X_4] = \\ &\mathbb{E}[X_1X_2]\mathbb{E}[X_3X_4] + \mathbb{E}[X_1X_3]\mathbb{E}[X_2X_4] + \mathbb{E}[X_1X_4]\mathbb{E}[X_2X_3] \\ &= \Sigma_{12}\Sigma_{34} + \Sigma_{13}\Sigma_{24} + \Sigma_{14}\Sigma_{23} \quad .\end{aligned}$$

For Y_1, Y_2, Y_3, Y_4 with means $\mu_1, \mu_2, \mu_3, \mu_4$ and covariance matrix Σ, we calculate the expected product by expanding

$$\mathbb{E}[Y_1Y_2Y_3Y_4] = \mathbb{E}[(X_1+\mu_1)(X_2+\mu_2)(X_3+\mu_3)(X_4+\mu_4)] \quad .$$

Recall that odd-moments are zero $\mathbb{E}(X_iX_jX_k) = 0$. The rest is straightforward algebra.

B Derivations

The following derivations are for Bayesian PCA. For Robust PCA replace τ with the expected precision T and the update for τ is adjusted by B for τ_Z and $1-B$ for τ_ϵ.

Our goal is to find the closest factorized distribution q to the posterior distribution π^*. Recall that the loglikelihood of our model is

$$\begin{aligned}\log \mathcal{L} = &-\sum_{ij} \frac{B_{ij}}{2}\left[\tau_Z (M_{ij} - U_i \cdot V_j)^2 + \log \sigma_Z^2\right] \\ &-\sum_{ij} \frac{1-B_{ij}}{2}\left[\tau_\epsilon (M_{ij} - U_i \cdot V_j)^2 + \log \sigma_\epsilon^2\right] \quad ,\end{aligned}$$

and that minimizing the KL-Divergence

$$D_{KL}(q||\pi^*) = \int q \log \frac{q}{\pi^*} \quad ,$$

is equivalent to maximizing $\mathcal{G}(q)$

$$\mathcal{G}(q) = \mathbb{E}_q(\log \mathcal{L}) - D_{KL}(q||\pi) \quad .$$

B.1 The U, V Update

Recall that $q(U)$ is the product of n k-dimensional multivariate normals and $q(V)$ is the product of m k-dimensional multivariate normals. To minimize the KL-divergence we solve for the critical points of $\mathcal{G}$ with respect to q parameters μ, ν, Σ. Recall that

$$T_{ij} = \begin{cases}\mathbb{E}(\tau_Z) b_{ij} + \mathbb{E}(\tau_\epsilon)(1-b_{ij}) & \text{if Observed} \\ 0 & \text{if Missing}\end{cases} \quad .$$

Then (with respect to μ, ν, Σ) our objective function is,

$$\begin{aligned}\mathcal{G}(\mu, \nu, \Sigma) \propto &\sum_{ij} T_{ij} M_{ij} (\mu_i \cdot \nu_j) \\ &-\sum_{ij} T_{ij} \left(\Sigma^i + \mu_i \mu_i^T\right) \circ \left(\Sigma_j + \nu_j \nu_j^T\right)/2 \\ &- D_{KL}(q_U||\pi_U) - D_{KL}(q_V||\pi_V) \quad ,\end{aligned}$$

where the KL-Divergence for two multivariate normal distributions is

$$D_{KL}(q||\pi) = \sum_i \frac{\mathrm{Tr}(\Sigma^i) + (\mu_i - \mu_0)^2 + \sigma_0^2 \log|\Sigma^i|}{2\sigma_0^2} \quad .$$

Recall that $\partial \log|A|/\partial x = \mathrm{Tr}(A^{-1}\partial A/\partial x)$.

Taking the derivative with respect to Σ^I gives

$$\begin{aligned}\frac{\partial \mathcal{G}}{\partial \Sigma^I_{kk'}} = &-\frac{1}{2}\sum_j T_{ij}\left(\Sigma^j_{kk'} + (\nu_j \nu_j^T)_{kk'}\right) \\ &-\frac{1}{2\sigma_0^2}\delta_{kk'} + 1/2(\Sigma^I)^{-1} \quad .\end{aligned}$$

Setting this equal to zero and solving for Σ^I gives

$$\Sigma^I = \left[\sum_j \left(\Sigma^j + \nu_j \nu_j^T \right) T_{Ij} + \mathbb{I}_d / \sigma_0^2 \right]^{-1}$$

Taking the derivative with respect to μ^I gives

$$\begin{aligned} \frac{\partial \mathcal{G}}{\partial \mu_I} =& (M_{I\cdot} \circ T_{I\cdot}) \cdot \nu - \left(\sum_j T_{Ij} \Sigma^j + \nu_j \nu_j^T \right) \mu_I \\ & - \frac{1}{\sigma_0^2} (\mu_I - \mu_0) \quad . \end{aligned}$$

Setting this equal to zero and solving for μ_I gives

$$\mu_I = \Sigma^I \left[(T_{I\cdot} \circ M_{I\cdot}) \cdot \nu + \frac{\mu_0}{\sigma_0^2} \right]$$

The updates for V are similar by symmetry.

B.2 The τ_Z, τ_ϵ Update

Recall that $q(\tau_Z)$ and $q(\tau_\epsilon)$ are Gamma distributions. To minimize the KL-divergence we solve for the critical points of $\mathcal{G}$ with respect to q's parameters $\alpha_Z, \beta_Z, \alpha_\epsilon, \beta_\epsilon$. Recall that

$$\langle S_{ij} \rangle = \mathbb{E} \left[(M_{ij} - U_i \cdot V_j)^2 \right] \quad .$$

Then, our objective function (with respect to α, β) is

$$\begin{aligned} \mathcal{G} =& \sum_{ij} \frac{-T_{ij}}{2} \langle S_{ij} \rangle + \frac{b_{ij}}{2} \mathbb{E}_q(\log \tau_Z) + \frac{1 - b_{ij}}{2} \mathbb{E}_q(\log \tau_\epsilon) \\ & - D_{KL}(q_Z || \pi_Z) - D_{KL}(q_\epsilon || \pi_\epsilon) \quad , \end{aligned}$$

where the KL-Divergence between two Gamma distributions is

$$\begin{aligned} D_{KL}\left(q||\pi\right) =& (\alpha - \alpha_0)\psi(\alpha) - \log \Gamma(\alpha) + \log \Gamma(\alpha_0) \\ & + \alpha_0(\log \beta - \log \beta_0) + \alpha(\beta_0 - \beta)/\beta \quad , \end{aligned}$$

and the expected log of a Gamma random variable is $\mathbb{E}(\log \tau) = \psi(\alpha) - \log(\beta)$ where $\psi(\cdot)$ is the Digamma function.

Taking the derivative with respect to α_Z, β_Z gives

$$\begin{aligned} \frac{\partial \mathcal{G}}{\partial \alpha_Z} =& \sum_{ij} \frac{-b_{ij}}{2\beta_Z} \langle S_{ij} \rangle - \frac{\beta_{Z0} - \beta_Z}{\beta_Z} \\ & + \left(\sum_{ij} \frac{b_{ij}}{2} + \alpha_{Z0} - \alpha_Z \right) \frac{\partial \psi}{\partial \alpha_Z}(\alpha_Z) \quad , \end{aligned}$$

and

$$\frac{\partial \mathcal{G}}{\partial \beta_Z} \propto \alpha_Z \left(\sum_{ij} \frac{b_{ij} \langle S_{ij} \rangle}{2} + \beta_{Z0} \right) - \beta_Z \left(\sum_{ij} \frac{b_{ij}}{2} + \alpha_{Z0} \right)$$

Setting the equations to zero and solving for α_Z and β_Z gives

$$\begin{aligned} \alpha_Z &= \alpha_{Z0} + \sum_{ij} \frac{b_{ij}}{2} \\ \beta_Z &= \beta_{Z0} + \sum_{ij} \frac{b_{ij} \langle S_{ij} \rangle}{2} \quad . \end{aligned}$$

Repeating this proceedure for α_ϵ and β_ϵ results in

$$\begin{aligned} \alpha_\epsilon &= \alpha_{\epsilon 0} + \sum_{ij} \frac{1 - b_{ij}}{2} \\ \beta_\epsilon &= \beta_{\epsilon 0} + \sum_{ij} \frac{(1 - b_{ij}) \langle S_{ij} \rangle}{2} \quad . \end{aligned}$$

C Other Variational Methods

This is future work. I have my notes, but they aren't clean yet. Email me if you are interested.

In short, Belief Propagation [Yedidia 2003] does not have a nice conjugate prior and therefore the continuous valued messages are ugly. Expectation Propagation [Seeger 2005] suffers a similar fate and reduces to naive VB when 2nd-order Taylor series approximations are made.