University of Washington
Department of Computer Science and Engineering / Department of Statistics

CSE 599 / Stat 592 Machine Learning (Statistics) for Big Data

Homework 4
Winter 2013

Issued: Sunday, March 3, 2013 **Due:** Tuesday,March 12, 2013

Suggested Reading: Assigned Readings in Case Study III and IV (see website).
Instructions: The homework consists of two parts: (i) Problems 4.1 covers theoretical and analytical questions and (ii) Problem 4.2 covers data analysis questions. Please submit each portion as *separate* sets of pages with your name and userid (UW student number) on each set. For Part II, please print out your code and graphs and attach them to the written part of your homework. Alternatively, you may submit via Catalyst as described in the Google Group.

Problem 4.1

Coordinate Descent for the Graphical Lasso

The graphical lasso is a method for structure learning in an undirected Gaussian graphical model, using ℓ_1 regularization to encourage sparsity in the precision matrix Ω (the inverse of covariance matrix Σ^{-1}). If the ij-th component of $\Omega = \Sigma^{-1}$ is zero, then variables i and j are conditionally independent given the other variables. Thus, zeros in the precision matrix encode conditional independence statements.

Let $Y = [y^1, ..., y^N]' \in R^{N\times d}$, where $y^i \in R^d$, N is the number of samples, and d is the number of dimensions. Let S denote the sample covariance matrix. As show in lecture, the graphical lasso is the solution to minimizing

$$-\log|\Omega| + \mathrm{tr}(S\Omega) + \lambda||\Omega||_1$$

over positive definite matrices Ω. The first two terms correspond to the negative log likelihood and $||\Omega||_1 = \sum_{i,j} |\omega_{ij}|$, where ω_{ij} are the elements of Ω.

One approach to solving the graphical lasso is based on a block coordinate descent algorithm, similar to the shooting algorithm for lasso. This approach operates by iteratively updating an estimate of the covariance matrix $\hat{\Sigma}$, from which an estimate of the precision $\hat{\Omega}$ can be readily formed. We are going to derive the update steps in this exercise. Throughout, let σ_{ij} be the ijth element of Σ.

(a) Derive $\gamma_{ij} = \partial_{\omega_{ij}}||\Omega||_1$, the subgradient of the matrix ℓ_1 norm.

(b) Let Γ be the matrix with elements γ_{ij}. Write the graphical lasso subgradient in terms of Σ, S, Γ, and λ and set it equal to 0.

(c) Find the optimal σ_{ii}. (*Hint:* $\omega_{ii} > 0$ *for all* $i = 1, \ldots, d$.)

Now we know how to find the optimal diagonal terms of Σ. The remaining parts will guide you through the derivation of a coordinate descent algorithm for the terms σ_{ij}, $i \neq j$. We can actually think about updating an entire column of Σ all at once. Recalling that Σ is symmetric, we are really updating a row *and* column.

Assume we have a current estimate of our covariance matrix, which for simplicity of notation we also denote by Σ with corresponding inverse Ω. When considering the update to the ith row and column, we examine the partitioned matrix

$$\begin{bmatrix} \Sigma_{-i} & \sigma_{i\cdot} \\ \sigma_{i\cdot}^T & \sigma_{ii} \end{bmatrix}, \tag{1}$$

where Σ_{-i} denotes the $(d-1) \times (d-1)$ matrix of rows and columns of Σ except for the ith, and $\sigma_{i\cdot}$ denotes the ith column of Σ except for the element σ_{ii} itself. We also rearrange and partition S and Ω accordingly:

$$\begin{bmatrix} S_{-i} & s_{i\cdot} \\ s_{i\cdot}^T & s_{ii} \end{bmatrix}, \quad \begin{bmatrix} \Omega_{-i} & \omega_{i\cdot} \\ \omega_{i\cdot}^T & \omega_{ii} \end{bmatrix}.$$

Since we have already solved for σ_{ii}, we focus exclusively on the vector $\sigma_{i\cdot}$ of terms σ_{ij}, $j \neq i$.

(d) Recall that we showed in part (b) that

$$-\sigma_{i\cdot} + s_{i\cdot} + \lambda\gamma_{i\cdot} = 0. \tag{2}$$

In this part we will show that for some definition of $\beta \in \mathbb{R}^{(d-1)\times 1}$, we can rewrite this equation as

$$-\Sigma_{-i}\beta + s_{i\cdot} - \lambda\nu = 0, \tag{3}$$

where $\nu_j = \text{sign}(\beta_j)$ if $\beta_j \neq 0$ and $\nu_j \in [-1, 1]$ if $\beta_j = 0$.

(i) Determine $\sigma_{i\cdot}$ in terms of Σ_{-i} and elements of Ω.
(*Hint: Use the identity* $\Sigma\Omega = I$ *for the partitioned matrices, expanding the terms for the top righthand block.*)

(ii) Find β such that the first term of Eq. (2) matches that of Eq. (3).

(iii) Prove that ν serves the same function as $\gamma_{i\cdot}$.
(*Hint: Use the definition of* β *derived above.*)

(e) In our standard lasso setting, we aimed to minimize the following objective with respect to β:

$$\frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1.$$

(i) Derive the subgradient corresponding to this objective.

(ii) Letting $X = (\Sigma_{-i})^{1/2}$ and $y = (\Sigma_{-i})^{-1/2} s_{i\cdot}$, show that this lasso subgradient corresponds to the subgradient in Eq. (3).

Because we can frame the solution for β in terms of a lasso solution, we can use any number of lasso algorithms to iteratively solve for $\hat{\beta}$. Using the optimal $\hat{\beta}$, we can then solve for $\sigma_{i\cdot}$.

(f) Recover $\sigma_{i\cdot}$ from $\hat{\beta}$.

(g) Fill in the pseudocode in Algorithm 1 for graphical lasso.

Algorithm 1: Graphical Lasso

---

1. Initialize $\Sigma = S + \lambda I$.
2. Repeat for $i = 1, 2, \ldots, d, 1, 2, \ldots, d, \ldots$ until convergence:
   (a) Partition the matrix Σ as in Eq. (1).
   (b) Solve for $\hat{\beta}$ as the solution to FILL IN THE ESTIMATING EQUATIONS using any lasso solver for this modified lasso.
   (c) Update $\sigma_{i\cdot}$ in terms of $\hat{\beta}$ as FILL IN THE UPDATE EQUATION.
3. Recover Ω as FILL IN SOLUTION.

Problem 4.2

Collaborative Filtering and Clustering on the Netflix Dataset

**Matrix notation: We will use subscript(superscript) to denote the column(row) of a matrix, e.g. For any matrix X, X_j is the jth column, X^i is the ith row, and $X_{i,j}$ is the entry (i, j).*

This problem contains 2 parts. In the first part, you will learn to use GraphChi (`graphchi.org`) to perform matrix factorization on the Netflix Dataset. In the second part, you will use the latent factor matrix from part 1 to perform kmeans and spectral clustering.

The famous Netflix dataset contains 100M ratings from $m = 480K$ users to $n = 17770$ movies. Let X be the $n \times p$ rating matrix, and k be the number of latent factors, the goal is to find $U \in R^{n\times k}$ and $V \in R^{m\times k}$ such that

$$f(U, V) = \frac{1}{2} \sum_{i,j:X_{i,j}\neq 0} \Big((X_{i,j} - U^{i^T} V^j)^2 + \lambda \|U^i\|^2 + \lambda \|V^i\|^2 \Big) \tag{4}$$

is minimized.

One algorithm that solves the above optimization problem is the Alternating Least Squares method discussed in class. This algorithm is already implemented in GraphChi.

- Download "netflix_mm.gz" from the course website.
- Run "gunzip netflix_mm.gz" and you will get the rating matrix in sparse Matrix Market format.
- Follow the instruction in
  `bickson.blogspot.co.il/2013/02/setting-up-java-graphchi-development.html`
  to setup graphchi-java.

The latent dimension k trades off computation with accuracy.

(a) Run ALSMatrixFactorization with $nshards = 10$ and $k = 2, 4, 6, 8, 10$.

  - Plot the training RMSE against k.
  - Plot the engine runtime against k.

The resulting latent factor U and V are stored in "netflix_mm_U.mm" and "netflix_mm_V.mm" in sparse Matrix Market format.

In the next part, you will use the latent factor matrix V as movie features, and cluster the movies using Kmeans and Spectral Clustering (Normalized Cut). In you use Matlab, for the next 2 questions, you only need to implement spectral clustering (normalized cuts) in "Ncut.m" and the rest of the logics are handled by the starter code. For simplicity and better interpretability, we only cluster the first 3000 movies. (The first 3000 rows in matrix $netflix\_mm\_V$).

(b) Run "kmeans_run.m" and report 20 closest movies for each cluster. (No coding required, if using the starter code.)

(c) Implement the normalized cuts algorithm in "Ncut.m".

(d) Run "ncut_run.m" in the starter code and report 20 closest movies for each cluster. (No coding required, if using the starter code.)