Bounding tree-based Sparse PCA using perturbation theory

Daniel Ringwalt

January 5, 2023

1 Perturbing the partial solution

- Some nodes in the tree will commit an additional variable to the partial solution.
- The node can be represented by an indicator vector $z \in \{0, 1\}^n$ having lower and upper bounds.
- The nodes of interest will assign a value of 1 at a formerly unbounded location.
- Sparse PCA is given by the SVD of a completed data subset M_z, $z \in \{0, 1\}^n, ||z||_1 = k$.

2 Lower bound from the partial solution

- Use the SVD of the partially built data matrix $M_{\mathbf{1}}$.
- Specifically, we can compute the dominant left singular vector u.
- There is a different u after subsetting M according to the global optimum k-sparse PCA solution.
- Suppose that the correlation of u and the globally optimum u is very high.
- Select the remaining $k - i$ variables from M greedily according to their dot product with u.
- We may find a better u and objective value on this greedy subset by running SVD again, or assume that u is of high quality and take the sum of dot products as a lower bound on the objective.
- This deterministic (over some fixed $M_{\mathbf{1}}$) algorithm will be evaluated against stochastic sparse PCA.

3 Upper bound on completing the solution

- Completing $M_{\mathbf{1}}$ is a combinatoric problem to be addressed in the next section.
- Suppose that $M_{\mathbf{1}}$ is a good approximation of the i-sparse PCA problem, so the left-SVD of $M_{\mathbf{1}}$ is starting to approach the left-SVD of our solution M_z.
- Consider the Gram matrix $M_{\mathbf{1}}M_{\mathbf{1}}^T$ (where the entries of the matrix represent the observations): $\Sigma_{\mathbf{1}}$.
- The matrix $\Sigma_{\mathbf{1}}$ has an eigensystem where only the dominant eigenvalue is of interest, so we can approximate $\Sigma_{\mathbf{1}}$ as being rank-one.
- We can complete $\Sigma_{\mathbf{1}}$ by adding a rank-$k-i$ matrix Σ_{k-i}. Assume that this matrix is fixed, and consider the combinatorial problem in the next section.
- The operator norm, or definite matrix spectral radius, $||\Sigma_{\mathbf{1}} + \Sigma_{k-i}||$ is upper-bounded by $||\Sigma_{\mathbf{1}}|| + ||\Sigma_{k-i}||$.
- Project the data onto u: $||\Sigma_{k-i}||_2 = ||M_{k-i}^T u||_2^2 + ||\Sigma_{k-i}^{\text{res}}||_2$.
- The operator norm $||\Sigma_{k-i}^{\text{res}}||_2$ is given by some vector $v, ||v||_2 = 1$ which maximizes $||\Sigma_{k-i}^{\text{res}} v||_2$.
- In this case, we constructed $\Sigma_{k-i}^{\text{res}}$ to be nearly singular by subtracting the projection onto u, so u and this dominant v are almost orthogonal. The bound is not very tight.
- For perturbing one positive semidefinite eigensystem by adding another such matrix, we can start from the dominant eigenvalue/eigenvector of one particular summand, and use a matrix-vector multiplication for the other summand:

$$||\Sigma_{\mathbf{1}} + \Sigma_{k-i}|| \leq ||M_{\mathbf{1}}u||_2^2 + ||M_{k-i}u||_2^2 + ||\Sigma_{k-i}^{\text{res}} u||_2$$

4 Parameterizing a solution

Revisit the indicator vector $z \in \{0,1\}^{n-i}, ||z||_1 = k - i$ (the contribution of the original i variables to the objective function has already been made constant). The objective function should be heavily influenced by the rank-one projection, with a greater contribution than the residual's contribution: $\sum_i z_i (M_{*i}^T u)^2$.

4.1 Residual term

For the residual's contribution, we need to insert a sparsity-inducing term into the middle of the matrix-vector multiplication over the full covariance matrix: $||\Sigma_{n-i}^{\text{res}} v||_2$. We sparsify the rows of Σ and of u using a diagonal matrix on the left side, but the sparsified u should be unit-normed:

$$\frac{||\text{Diag}(z)\Sigma_{n-i}^{\text{res}}\text{Diag}(z)v||}{||\text{Diag}(z)v||}$$

4.2 Limitations of quadratic programming

Our objective function could start to look like a matrix-vector multiplication quadratic. Just like PCA (which is solved by the dominant eigenvector), the equation $\max v^T \Sigma v$ cannot be solved via an optimization program. Usually, convex optimization solvers would be applied to a positive definite matrix of costs/risks to be minimized, not maximized.

Plug-ins can be created for some optimization libraries in the form of custom cones. For example, the set $(t \in \mathbb{R}, x \in \mathbb{R}^n) : t \geq ||x||_2$ is a quadratic cone constraint. The cone must be closed under addition (in $(t_1, x_1) + (t_2, x_2)$, the t grows very quickly and the bound clearly holds in this direction). Also, the cone must be closed under scaling by a real number (so we cannot bound the reciprocal ℓ_2 norm in the direction which we might want, for scaling our objective function post-hoc).

4.3 Semidefinite programming relaxation

Rewriting our objective function using a rank-one zero-one matrix zz^T helps us find $||z||_2^2$ using Tr (zz^T). A rank-one constraint would not be a conic constraint (the set of matrices is clearly not closed under addition), so the constraint is usually dropped. Dropping Diag(z), use a PSD matrix (expected to be low-rank) S (sparsity-inducing matrix).

$$||\text{Diag}(z)v||_2^2 = v^T \text{Diag}^2(z) v \approx v^T S v = \text{Tr } v^T S v = \text{Tr } S v v^T = \left\langle S, vv^T \right\rangle$$

The sparsity-inducing matrix S will be applied to our residual matrix here, so use a scaled matrix S_{res} with constraint (v defined ahead of time) $\left\langle S_{\text{res}}, vv^T \right\rangle = 1$. Later, we will scale S back up by some variable scale factor (which cannot be expressed ahead of time in a linear program).

For the (squared) numerator, we will need to generalize this inner product to encompass 3 PSD matrices. The inner product $\langle A, B \rangle$ can be expressed as $\sum_i \sum_j a_{ij} b_{ij} = \sum_i \sum_j (A \odot B)_{ij}$, and we will generalize this to applying 3 PSD matrices.

$$||\text{Diag}(z)\Sigma_{n-i}^{\text{res}}\text{Diag}(z)v||_2^2 \approx ||(S_{\text{res}} \odot \Sigma_{n-i}^{\text{res}})v||_2^2 = v^T (S_{\text{res}} \odot \Sigma_{n-i}^{\text{res}})^2 v$$

In general, introducing square convex terms into a maximization (not minimization) problem would lead to NP-hardness (a la quadratic programming minimization with an indefinite matrix). We cannot square the matrix, and will bound the action of the matrix's spectrum on this vector v. First, we have $vS_{\text{res}}v^T = 1$ as one of our constraints. Elementwise multiplication by the matrix $\Sigma^{\text{res}}_{n-i}$ produces another positive semidefinite matrix (Schur product theorem). This matrix can be loosely bounded using the largest absolute value element of Σ, which is found on its diagonal, multiplied by S_{res}. For any bound B on $vMv^T \leq B$ for positive semidefinite M, we expect $vM^2v^T \leq vMv^T * B$.

$$||\text{Diag}(z)\Sigma^{\text{res}}_{n-i}\text{Diag}(z)v||_2^2 < \left\langle S_{\text{res}}, \Sigma^{\text{res}}_{n-i} \odot vv^T \right\rangle * \max_i \left| \left(\Sigma^{\text{res}}_{n-i}\right)_{ii} \right|$$

Finally, this term in the objective requires a new scalar value which is the square root. The square root function is concave. The quadratic cone can allow for slack in the inequality $|a| \leq \sqrt{b}$, so conic programming is useful for a square root term which represents a valuable quantity, but not for representing the square root of a cost.

4.4 Rank-one projection term

The impact of approximations above is limited in scope, since the problem should have reasonable eigenvalue separation and the current rank-one principal component should be a guess at diagonalizing the problem. Therefore, we add large contributions to the problem (rank-one projections of each variable) which are linear in the diagonal entries of S. If the positive coefficients in the entries of S dominate the problem, then the problem reduces to selecting $S = zz^T, z \in \{0,1\}^n, ||z||_0 = k$.

4.5 Scaling S using the power cone

If S_{res} is rank-one, then we can recover the vector representing its range, applying some scale factor to be determined. We simply need a square root and arbitrary scaling of the diagonal entries of S_{res} (use positive square root, since all entries of S_{res} are constrained to be nonnegative). This can be done in conic programming using the power cone $\mathcal{P}_3^{1/2}$, which is the set of triples $(x_1, x_2, x_3) : \sqrt{x_1 x_2} \geq |x_3|$. We will have n conic constraints where x_2 is some scalar variable not otherwise used, x_1 is a diagonal entry of S_{res}, and x_3 is a problem variable taking the rank-one projection as a coefficient.

The free variable x_2 in the conic constraints will generally be greater than 1, to scale up the sparsity S entries to a unit value. The x_3 problem variables (z_i) are constrained by the unit box. The z_i variables are used to sum up the contribution of variance from the rank-one projection, and should be at unity.