Lecture 14 — May 14

Lecturer: Lester Mackey *Scribe: Renjie You*

14.1 Recap

Recall that the goal of sparse / interpretable unsupervised learning is to obtain results that only depend on a small subset of the input features so that we can better interpret those results. For now, we will continue to specialize our focus to sparse PCA, where we want to obtain sparse loadings $u_j \in \mathbb{R}^p$. Last time we considered several modern approaches to tackling the sparse PCA problem, including combinatorial branch-and-bound, greedy variable selection approach, and SCoTLASS. Today, we will explore several alternative strategies.

14.2 SPCA

See ESL 14.5.5 for more reading. Like SCoTLASS, **SPCA** transforms the combinatorial cardinality-constrained sparse PCA problem into a continuous optimization problem by replacing the cardinality constraint with an ℓ_1 surrogate. However, SPCA works with a different formulation of the PCA optimization problem and employs a different algorithm. More precisely, SPCA solves the following optimization problem for u_1:

$$\begin{aligned} \min_{u_1, v_1} \quad & \sum_{i=1}^{n} \left\| x_i - v_1 u_1^T x_i \right\|_2^2 + \lambda \left\| u_1 \right\|_2^2 + \lambda_1 \left\| u_1 \right\|_1 \\ \text{s.t.} \quad & \left\| v_1 \right\|_2 = 1, \end{aligned}$$

which is an ℓ_1 penalized variant of the PCA reconstruction formulation (R2) from last lecture. If we removed the ℓ_1 penalty, this would just be PCA, but the ℓ_1 penalty encourages sparsity. Note that λ and λ_1 are both tuning parameters. As you vary λ_1 for fixed λ, you get different amounts of sparsity. The vector u_1^* recovered from an optimal solution (u_1^*, v_1^*) to this problem represents the first sparse loadings vector.

14.2.1 Multi-component version

This SPCA formulation only solves for a single component u_1. Suppose that you wanted to solve for multiple components at once. Then we could define $V = \begin{bmatrix} — & v_1 & — \\ — & v_2 & — \\ & \vdots & \\ — & v_k & — \end{bmatrix} \in$

$\mathbb{R}^{k\times p}$ and $U = \begin{bmatrix} — & u_1 & — \\ — & u_2 & — \\ & \vdots & \\ — & u_k & — \end{bmatrix} \in \mathbb{R}^{k\times p}$ and solve the multi-component optimization problem (recommended in ESL),

$$\begin{aligned} \min_{U,V} \quad & \sum_{i=1}^{n} \left\|x_i - VU^T x_i\right\|_2^2 + \lambda \sum_{j=1}^{k} \|u_j\|_2^2 + \sum_{j=1}^{k} \lambda_{1j} \|u_j\|_1 \\ \text{s.t.} \quad & V^T V = I. \end{aligned}$$

In the multi-component version, we have introduced a tuning parameter λ_{1j} for each j in order to allow for different sparsity levels in different loadings.

Note that this is a non-convex program subject to local minima for the same reason that SCoTLASS was. However, we can approximately find a local optimum fairly efficiently using **alternating minimization**:

- When V is held fixed, the problem of optimizing over U decouples into k independent **elastic net regression** problems, each one with variable u_j, that can be solved using standard elastic net implementations.
- When U is held fixed, optimization over V is a **reduced rank Procrustes problem**. If $A\Sigma B^T$ is the SVD of $X^T XU$, then the update for V is $V = AB^T$.
- The SPCA algorithm updates U and V until convergence.

14.3 Direct Sparse PCA (DSPCA)

See d'Aspremont et al. (2007) for more reading.

We will next consider a rather different approach to sparse PCA called **DSPCA**. Here we will replace the combinatorial cardinality-constrained maximum variance problem with a convex relaxation problem. Unlike the other surrogates formulations we have seen, this one is convex, and we can find the global optimum.

Let $A = \dfrac{X^T X}{n}$. Then, our intractable cardinality-constrained problem problem is:

$$\begin{aligned} \max_{u_1} \quad & u_1^T A u_1 \\ \text{s.t.} \quad & \|u_1\|_2 = 1 \\ & \text{card}(u_1) \le c_1 \end{aligned}$$

We are going to relax this into a convex optimization problem.

First, we will use the trace trick to obtain the equivalent problem

$$\begin{array}{ll} \max\limits_{u_1} & \text{tr}(u_1 u_1^T A) \\ \text{s.t.} & \text{tr}(u_1 u_1^T) = 1 \\ & \text{card}(u_1 u_1^T) \leq c_1^2. \end{array}$$

Then, we will perform **semidefinite lifting** by introducing $M = u_1 u_1^T$ to obtain another equivalent formulation

$$\begin{array}{ll} \max\limits_{M \in \mathbb{R}^{p \times p}} & \text{tr}(MA) \\ \text{s.t.} & \text{tr}(M) = 1 \\ & \text{card}(M) \leq c_1^2 \\ & M \succeq 0 \\ & \text{rank}(M) = 1. \end{array}$$

The value of this new formulation is that we are maximizing a linear function of our optimization variable instead of maximizing a quadratic function (which is a difficult thing to do in general). However, we still have a non-convex cardinality constraint and a non-convex rank constraint to deal with.

We can relax these two constraints to form a convex semidefinite program in the following way. First let $\text{vec}(M)$ denote vectorized form of the matrix M (with the columns stacked into a single vector. Then note that for any vector h, $\|h\|_1 \leq \sqrt{\|h\|_0}\,\|h\|_2$.[1] Under our problem constraints we have $\text{card}(M) = \|\text{vec}(M)\|_0$ and $\|\text{vec}(M)\|_2 = \sqrt{\text{tr}(M^T M)} = \sqrt{(u_1^T u_1)^2} = 1$. Hence, $\|\text{vec}(M)\|_1 \leq \sqrt{\|\text{vec}(M)\|_0}\,\|\text{vec}(M)\|_2 = c_1$, so we can replace the cardinality constraint with the relaxation $\|\text{vec}(M)\|_1 \leq c_1$. Finally, we drop the rank constraint (the ultimate relaxation).

This gives us the **DSPCA relaxation**:

$$\begin{array}{ll} \max\limits_{M \in \mathbb{R}^{p \times p}} & \text{tr}(MA) \\ \text{s.t.} & \text{tr}(M) = 1 \\ & \|\text{vec}(M)\|_1 \leq c_1 \\ & M \succeq 0 \end{array}$$

This is a convex semidefinite program (SDP), which we know how to solve. We recover the first component loadings u_1 from M by finding a leading eigenvector. The authors suggest that if your problem is small enough, you can use generic interior point solvers. However, these have a high complexity, so for larger problems, there is a customized first-order optimization algorithm.

The paper shows that this approach typically outperforms SPCA in the sense that for a fixed level of sparsity, it explains more variance, and for a fixed level of variance explained, it achieves greater sparsity.

[1] This follows from the arithmetic mean-geometric mean (AM-GM) inequality.

14.4 Variational renormalization

See Moghaddam et al. (2006) for more reading.

Most sparse PCA techniques attempt to jointly solve the model selection problem (which loadings are nonzero?) and the model estimation problem (what are the values of the nonzero entries for the selected model?) and commonly yield suboptimal solutions for the estimation task (even if the model selection component is accurate). As a result, we can often improve a generic (single-component) sparse PCA solution by doing the following:

1. Fix the learned nonzero entry locations.
2. Maximize variance assuming the other entries are zero.

This **variational renormalization** is equivalent to finding an eigenvector of a submatrix of A. This is an inexpensive operation that typically leads to a better solution than the original.

14.5 Multiple components

So far we have primarily focused on extracting a single sparse PCA loadings vector. Some algorithms can be explicitly reformulated to extract many components (e.g., SPCA). Even those that cannot be reformulated in this way can often be used to extract single components successively by, for example,

1. Constraining additional loadings to be orthogonal (this may be overly restrictive),
2. Constraining additional component variables ($u_j^T x$) to be uncorrelated, or
3. Finding loadings that explain the greatest additional variance.

It is important to note that only the true eigenvectors of A (i.e., the true PCA loadings) satisfy both (1) and (2) simultaneously, so at least one must be sacrificed in the sparse PCA setting (unless the true eigenvectors are already sufficiently sparse).

To better understand option (3), we need to formally define a notion of the **variance explained** by an arbitrary set of loadings $U = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix}$. We have a definition for the standard PCA loadings, but in general our loadings may not be orthogonal and may not be actual eigenvectors of $\frac{X^T X}{n}$.

One common way to extend the definition is to define the variance explained by U as the variance explained by an orthonormal basis $Q = \begin{bmatrix} q_1 & \cdots & q_{k'} \end{bmatrix}$ for $\text{span}(\{u_1, \ldots, u_k\})$. For any orthonormal basis, this variance explained is our usual quantity, $\sum_{j=1}^{k'} q_j^T \frac{X^T X}{n} q_j$ = the sum of basis component variances.

If a new loadings vector u_{k+1} is extracted, we can extend the basis by calculating $\tilde{q}_{k'+1} = (I - QQ^T)u_{k+1}$, which is the projection onto the orthocomplement, and renormalizing to obtain $q_{k'+1} = \frac{\tilde{q}_{k'+1}}{\|\tilde{q}_{k'+1}\|_2}$. The **additional variance explained** by u_{k+1} is then

$$q_{k'+1}\frac{X^TX}{n}q_{k'+1} = \frac{u_{k+1}^T(I-QQ^T)\frac{X^TX}{n}(I-QQ^T)u_{k+1}}{\|(I-QQ^T)u_{k+1}\|_2^2}.$$

This suggests the following criterion for selecting new components:

$$\begin{array}{ll}\max\limits_{u_{k+1}} & u_{k+1}^T(I-QQ^T)\dfrac{X^TX}{n}(I-QQ^T)u_{k+1} \\ \text{s.t.} & \left\|(I-QQ^T)u_{k+1}\right\|_2 = 1 \\ & \text{card}(u_{k+1}) \le c_{k+1}\end{array}$$

Most single component sparse PCA techniques can be adapted to solve this. This is called **generalized deflation** (see Mackey 2008).

14.6 Sparse PCA theory

The theory and complexity of sparse PCA under various data generating models is an active research area.

14.7 Beyond PCA

We could use the ideas discussed so far to find sparse versions of other latent feature models like FA, CCA, ICA, etc. We can also develop sparse clustering algorithms. Let us detail a particular example.

14.7.1 Sparse clustering with pairwise dissimilarity

Suppose that our goal is to cluster a set of datapoints and that for each pair $(x_i, x_{i'})$ and each coordinate j, we have a feature-specific measure of dissimilarity $d_{ii'j}$ between x_{ij} and $x_{i'j}$, e.g., $d_{ii'j} = (x_{ij} - x_{i'j})^2$.

A typical non-sparse clustering algorithm (like k-medoids or hierarchical clustering) would cluster using the dissimilarity matrix with entries $d_{ii'} = \sum_{j=1}^{p} d_{ii'j}$, which depends on all features. Our sparse clustering goal is to obtain a clustering that uses only a relevant subset of the features. For instance, we could form a dissimilarity matrix based on the weighted

$\text{sum}d_{ii'} = \sum_{j=1}^{p} w_j d_{ii'j}$, where many weights w_j are zero. However, we are now faced with the question of how to choose w.

One option, proposed by Witten and Tibshirani (2010), suggests that we first form a feature dissimilarity matrix $D \in \mathbb{R}^{n^2 \times p}$, where each row represents a *pair* of data points. Think of this as $(d_{ii'j})_{(i,i'),j}$. Then, we run single-component sparse PCA to obtain sparse loadings w that explain most of the variance in D. Specifically, we solve

$$\begin{aligned} \max_{w} \quad & w^T D^T D w \\ \text{s.t.} \quad & \|w\|_2 \le 1, \|w\|_1 \le c, w_j \ge 0 \end{aligned}$$

for $c \ge 1$. Note that this can be rewritten as

$$\begin{aligned} \max_{u,w} \quad & u^T D w \\ \text{s.t.} \quad & \|w\|_2 \le 1, \|w\|_1 \le c, w_j \ge 0 \\ & \|u\|_2 \le 1. \end{aligned}$$

The benefit of this formulation is that it is biconvex. Thus, we can use alternating minimization to update u and w in succession until convergence. The closed form update for u given w is

$$u = \frac{Dw}{\|Dw\|_2}.$$

The update for w can be computed in several steps. First define $a = (D^T u)_+$ (where x_+ is the positive part of x) and $b = (a - \Delta)_+$ (this is soft thresholding for a value Δ to be specified). Finally, we set $w = \dfrac{b}{\|b\|_2}$ (defined to be 0 if $\|b\|_2 = 0$). We choose Δ such that $\|w\|_1 \le c$. That is, $\Delta = 0$ if $\frac{\|a\|_1}{\|a\|_2} \le c$, and otherwise Δ is chosen such that $\|w\|_1 = c$.

The result is that we can form a dissimilarity matrix with entries $d_{ii'} = \sum_{j=1}^{p} w_j d_{ii'j}$ that only depends on a subset of features. Hence, any clustering algorithm based on this dissimilarity achieves our goal of only being based on a subset of features. See the accompanying slides for an example of these ideas used in the hierarchical clustering setting.