PCA Notes

Walid Krichene

1 SVD and PCA

1.1 Power iteration

We would like to recover the largest singular value σ_1, and corresponding singular vectors u_1 and v_1. To simplify the discussion, assume that

$$\sigma_2 < \sigma_1$$

We consider the following optimization problem

$$\min_{p \in \mathbb{R}^n, q \in \mathbb{R}^m} \|A - pq^T\|_F^2 \quad \text{s.t.} \|p\|_2 = 1$$

Let $f(p, q) = \|A - pq^T\|_F^2$. The optimizers are shown to be $(p^*, q^*) = (u_1, \sigma_1 v_1)$ (up to a sign change).

This problem can be solved using a block coordinate descent algorithm. Assume p is fixed, and such that $\|p\|_2 = 1$, and consider the partial optimization over q. The optimal q satisfies stationarity

$$\nabla_q f(p, q) = 0$$

we have $f(p, q) = \sum_{i,j}(A_{ij} - p_i q_j)^2$, and the partial derivative of $f(p, .)$ with respect to q_j is $\sum_i(-2p_i(A_{ij} - p_i q_j)) = -2\sum_i A^T_{ji} p_i + 2\sum_i p_i^2 q_j = -2(A^T p)_j + 2q_j$. The stationarity condition can then be written $-2A^T p + 2q = 0$, i.e.

$$q = A^T p$$

Now let q be fixed, and consider the partial optimization over p, subject to $\|p\|_2 = 1$. The optimal p satisfies stationarity of the Lagrange function

$$\begin{aligned} L(p, \nu) &= \|A - pq^T\|_F^2 - \nu(p^T p - 1) \\ &= Tr\left[(A - pq^T)^T(A - pq^T) - \nu p^T p\right] + \nu \\ &= Tr\left[qp^T pq^T - 2pq^T A^T - \nu p^T p\right] + Tr(A^T A) + \nu \\ &= Tr\left[p^T p(q^T q - \nu) - 2p(Aq)^T\right] + Tr(A^T A) + \nu \end{aligned}$$

the gradient is given by

$$\nabla_p L(p, \nu) = 2p(\|q\|_2^2 - \nu) - 2Aq$$

and from stationarity and primal feasibility we have at optimum $p = Aq/(\|q\|_2^2 - \nu)$ and $\|p\| = 1$, thus

$$p = \frac{Aq}{\|Aq\|_2} = P(Aq)$$

where P is the euclidean projection on the unit sphere.

We can then define an iterative algorithm, starting from a random unit vector p_0, update as follows

$$\begin{aligned} q_{k+1} &= A^T p_k \\ p_{k+1} &= P(Aq_{k+1}) = P(AA^T p_k) \end{aligned}$$

the current objective value at step k is then $\|A - p_k q_{k+1}^T\|_F^2 = \|A - p_k p_k^T A^T\|_F^2$. Note that one can keep track only of p_k, which is simply given by

$$p_k = P((AA^T)^k p_0)$$

1.1.1 convergence

Using the SVD of A, we have $(AA^T)^k = (U\Sigma^2U^T)^k = U\Sigma^{2k}U^T = \sum_{i=1}^r \sigma_i^{2k} u_i u_i^T$. We can bound the distance to the optimum $\|p_k - u_1\|_2$. Let $p_0 = \sum_{i=1}^m x_i u_i$, and assume $x_1 > 0$. Then we have

$$\begin{aligned}
p_k &= P((AA^T)^k p_0) \\
&= P\left(\sum_{i=1}^r \sigma_i^{2k} u_i u_i^T p_0\right) \\
&= P\left(\sum_{i=1}^r x_i \sigma_i^{2k} u_i\right) \\
&= \sum_{i=1}^r x_i \sigma_i^{2k} u_i / \sqrt{\sum_{i=1}^r (x_i \sigma_i^{2k})^2}
\end{aligned}$$

and the distance to the optimum is

$$\begin{aligned}
\|p_k - u_1\|^2 &= \|p_k\|^2 + \|u_1\|^2 - 2u_1^T p_k \\
&= 2 - 2\frac{(x_1\sigma_1^{2k})^2}{\sum_{i=1}^r (x_i\sigma_i^{2k})^2}
\end{aligned}$$

and using the fact that $\sigma_i \le \sigma_2 \ \forall i \ge 2$, we have

$$\begin{aligned}
\|p_k - u_1\|^2 &\le 2 - 2\frac{(x_1\sigma_1^{2k})^2}{\sigma_1^{4k}x_1^2 + \sigma_2^{4k}\sum_{i=2}^r x_i^2} \\
&= 2\left(1 - \frac{1}{1 + (\sigma_2/\sigma_1)^{4k}\sum_{i=2}^r (x_i/x_1)^2}\right) \\
&\sim 2(\sigma_2/\sigma_1)^{4k}\sum_{i=2}^r (x_i/x_1)^2
\end{aligned}$$

Therefore the convergence is geometric, with rate $(\sigma_2/\sigma_1)^4$. The complexity of one iteration is $O(\log\frac{1}{\epsilon})$, where the constant is proportional to $1/\log(\sigma_1/\sigma_2)$, since

$$\begin{aligned}
& (\sigma_2/\sigma_1)^{4k} \le \epsilon \\
\Leftrightarrow \quad & 4k\ln(\sigma_2/\sigma_1) \le \ln(\epsilon) \\
\Leftrightarrow \quad & 4k\ln(\sigma_1/\sigma_2) \ge \ln(1/\epsilon) \\
\Leftrightarrow \quad & k \ge \frac{\ln(1/\epsilon)}{4\ln(\sigma_1/\sigma_2)}
\end{aligned}$$

The coordinate descent algorithm converges faster when the second singular value is very small compared to the largest one. The complexity of each iteration is simply that of updating p, i.e. computing AA^Tp and then projecting it. This done in $O(nm)$, and the total complexity is therefore $O(nm\log\frac{1}{\epsilon})$.

1.1.2 faster convergence

Note that we can make the convergence faster by computing powers of $AA^T \in \mathbb{R}^{n\times n}$, which can be done in $O(n^3)$. The update procedure is then

$$\begin{aligned}
(AA^T)^{2^{k+1}} &= \left((AA^T)^{2^k}\right)^2 \\
p_{k+1} &= (AA^T)^{2^k} p_k
\end{aligned}$$

and the bound becomes

$$\|p_k - u_1\|^2 \leq 2(\sigma_2/\sigma_1)^{2^{k+1}} \sum_{i=1}^{r} (x_i/x_1)^2 + o((\sigma_2/\sigma_1)^{2^k})$$

and the convergence is exponential. The total complexity is $O(mn^2 + n^3 \log\log \frac{1}{\epsilon})$, we have to compute AA^T before the first iteration, which is $O(n^2 m)$, then each iteration requires computing the square of the $n \times n$ matrix AA^T, which is, in general $O(n^3)$, and the total number of iterations is $O(\log\log \frac{1}{\epsilon})$. Compared to the original power-iteration algorithm, which is $O(nm \log \frac{1}{\epsilon})$, this is only interesting if $m \log \frac{1}{\epsilon} > n^2 \log\log \frac{1}{\epsilon}$

1.2 Notes on PCA

Principal Component Analysis (PCA) is the problem of finding the best rank one approximation of a data matrix A (in the sense of the Frobenius norm). It is also equivalent to finding the direction that best explains the variance observed in the data points A_i. The objective function of the PCA problem can be written as

$$\begin{aligned} \|A - pq^T\|_F^2 &= Tr(A^T - qp^T)(A - pq^T) \\ &= Tr(A^T A) - 2Tr(p(Aq)^T) + Tr(qp^T pq^T) \\ &= Tr(A^T A) - 2(Aq)^T p + Tr(qq^T) && \text{since } \|p\|_2 = 1 \\ &= Tr(A^T A) - 2(Aq)^T p + \|q\|_2^2 \end{aligned}$$

Therefore the problem is

$$\begin{aligned} \min_{q, \|p\|_2 = 1} Tr(A^T A) - 2(Aq)^T p + \|q\|_2^2 &= Tr(A^T A) + \min_q \left(\|q\|_2^2 - 2 \max_{\|p\|_2 = 1} (Aq)^T p \right) \\ &= Tr(A^T A) + \min_q \left(\|q\|_2^2 - 2\|Aq\|_2 \right) \end{aligned}$$

since $\max_{\|u\|_2=1} v^T u = \|v\|_2$, and the optimal p is here $Aq/\|Aq\|_2$. The problem is therefore

$$\min_q Tr(A^T A) + \|q\|_2^2 - 2\|Aq\|_2$$

Note that we can optimize separately on the direction and the norm of q, since the problem can be reformulated as

$$\min_{\|q\|_2 = 1, \sigma \geq 0} Tr(A^T A) + \sigma^2 - 2\sigma\|Aq\|_2$$

Minimizing on q leads to the subproblem

$$\max_{\|q\|_2 = 1} \|Aq\|_2$$

whose optimizer is $q = v_1$, and the optimal σ is $\sigma = \|Aq\|_2 = \|Av_1\|_2 = \sigma_1$

The subproblem above is another standard formulation of PCA, where one seeks a direction that maximizes the variance: given the data matrix $A \in \mathbb{R}^{m \times n}$, the problem is

$$\max_{\|q\|_2 = 1} \|Aq\|_2$$

Note that $\|Aq\|_2 = q^T A^T A q = q^T S q$ is the variance along q, where $S = A^T A$ is the covariance matrix (assuming the data in A is centered)

Another interpretation of the PCA problem is finding an affine line $l = \mathbb{R}w + b$ that minimizes the sum of distances of points to l. The objective to minimize is

$$\sum$$

The problem can be reduced to linear hyperplanes of the form $H = \{x \in \mathbb{R}^n | w^T x = 0\}$, by centering the data. Indeed,

1.3 Robust PCA using ℓ-1 norm

[Credit: part of these notes is from Laurent El Ghaoui's notes]

Consider the problem of recovering a rank one matrix from noisy observations. One can solve the ℓ_1-norm problem

$$\min_{p\in\mathbb{R}^n, q\in\mathbb{R}^m} \|A - pq^T\|_1 \quad \text{s.t. } \|p\|_\infty = 1$$

where the ℓ_1-norm is that of the vector created by stacking all the matrix columns. One can use a block coordinate descent algorithm, similar to power iteration, by iteratively fixing one vector and optimizing on the other.

1.3.1 q-step

For a fixed $p \in \mathbb{R}^n$, the problem is

$$\min_{q\in\mathbb{R}^m} \sum_{j=1}^m \|A_j - q_j p\|_1 = \sum_{j=1}^m \min_{q_j} \|A_j - q_j p\|_1$$

where $A_j \in \mathbb{R}^n$ is the j-th column of A. Each subproblem is an ℓ_1-norm projection (projection of A_j onto span(p))

$$\min_t \|z - tp\|_1$$

This is a weighted median problem $\min_t \|z - tp\|_1 \Leftrightarrow \min_t \sum_{i\in\text{supp(p)}} |p_i| \left|\frac{z_i}{p_i} - t\right|$.

1.3.2 Median problem

The median problem is

$$\min_t \sum_{i=1}^n |z_i - t|$$

and an optimizer is $t = z_{[k]}$ where $z_{[i]}$ is the i-th element in the ordered sequence, k is the maximum index such that $k \leq n - k$ i.e. $\lfloor \frac{n}{2} \rfloor \leq k \leq \lceil \frac{n}{2} \rceil$. This can be solved by simply sorting the elements of z, which can be done in $O(n \log n)$.

Note that computing the median can be done in $O(n)$ if the elements are not sorted entirely: using the quick-select algorithm, a modified version of the quicksort algorithm. At each step, assume we have a list of k elements. We choose a random pivot, and in one pass compute the elements that are less than, respectively greater than, the pivot. From the length of each list, we know where to look for the median next. If the sizes of the sublists are sufficiently balanced, this will result in $O(n)$ algorithm. In the exactly balanced case, the complexity is proportional to $n + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{\log_2 n} = n\frac{1-(1/2)^{1+\log_2 n}}{1-1/2} \leq 2n$. In the case the size of each sublist is no less than α times the size of the parent list, the complexity is also linear, since it is proportional to $n + \alpha n + \alpha^2 n + \cdots + \alpha^{\log_{1/\alpha} n} n \leq \frac{1}{1-\alpha} n$. Although the worst-case complexity is $O(n^2)$, the algorithm is linear in practice. Assuming a uniform distribution of the permutation that will result in the sorted list, the expected complexity is $O(n)$.

1.3.3 Weighted median problem

The solution to the weighted median problem is slightly different

$$\min_t \sum_{i=1}^n \alpha_j |z_i - t|$$

where $\alpha_i > 0$. An optimizer is $t = z_{[k]}$ where k is the maximum index such that $\sum_{i=1}^k \alpha_i \leq \sum_{i=k+1}^n \alpha_i$. This can also be solved by sorting.

If we denote this subproblem by

$$\mathbf{wmed}(Z,u) = \arg\min_t \sum_j \|z_j - t_j u\|_1$$

where z_j is the j-th column of Z, then the q-step is simply given by

$$q \leftarrow \mathbf{wmed}(A,p)$$

and the complexity of the q-step is $O(mn\log n)$.

1.3.4 p-step

For a fixed $q \in \mathbb{R}^m$, the problem is given by

$$\min_{\|p\|_\infty=1} \sum_{i=1}^{n} \|A_i - p_i q\|_1$$

where A_i is the i-th row of A. We can denote this problem by

$$p \leftarrow \mathbf{pwmed}(A^T, q)$$

where $\mathbf{pwmed}(Z,u)$ is the solution of the projected weighted median problem

$$\min_{\|t\|_\infty=1} \sum_i \|z_i - t_i u\|_1 = \sum_{i=1}^{n} \min_{|t_i|\le 1} \|z_i - t_i u\|_1 \text{ subject to } \max_i |t_i| = 1$$

We can start by solving n projected weighted median problems $\min_{|t_i|\le 1} \|z_i - t_i u\|_1$. This is solved by computing the solution to the unconstrained weighted median $t_i = \arg\min_t \sum_{j\in\mathrm{supp}(u)} |u_j| \left|\frac{Z_{ij}}{u_j} - t_i\right|$ then projecting t_i on $[-1,1]$.

After solving the n subproblems, there are 2 cases. Either one of the t_i's satisfies $|t_i| = 1$, in which case the $\|t\|_\infty = 1$ constraint is satisfied and the problem is solved, or all the t_i's are such that $|t_i| < 1$. In this case, we choose which t_i to project on $[-1,1]$ by solving the problem

$$\begin{array}{ll} \text{minimize} & -\|Z_i - t_i u\|_1 + \|Z_i - \epsilon u\|_1 \\ \text{subject to} & i \in \{1,\ldots,n\} \\ & \epsilon \in \{-1,1\} \end{array} \tag{1}$$

this problem finds (i,ϵ) that minimize the increase of the objective value that is due to setting $t_i = \epsilon$. Then if (i_0,ϵ_0) is a minimizer, we set $t_{i_0} = \epsilon_0$. This additional projection step is $O(nm)$ since there are $2n$ feasible points to evaluate in problem (1).

1.3.5 Algorithm

The iterative algorithm is

$$q \leftarrow \mathbf{wmed}(A,p)$$
$$p \leftarrow \mathbf{pwmed}(A^T,q)$$

The complexity of each iteration is $O(nm\log(nm))$ if the weighted median is computed by sorting the vectors, or $O(nm)$ if the quick select algorithm is used instead.

1.3.6 convergence in the rank one case

Note on convergence when there is no noise. Assume A is rank one (no observation noise), and let $A = \sigma uv^T$ where $\sigma > 0$, u and v are unit vectors.

Then the p-step is given by

$$\begin{aligned} p_i &= \arg\min_{p_i} \|\sigma u_i v - p_i q\|_1 \\ &= \arg\min_{p_i} \sum_{j=1}^{n} |\sigma u_i v_j - p_i q_j| \\ &= \arg\min_{p_i} \sum_{j=1}^{n} |q_j| \left| \frac{\sigma u_i v_j}{q_j} - p_i \right| \\ &= \sigma u_i \tilde{v}_{[k]} \end{aligned}$$

where $\tilde{v}_i = \frac{v_i}{q_i}$, $\tilde{v}_{[k]}$ is the k-th element in the ordered sequence of $\tilde{v}$. Note that the sign of u_i does not affect the solution, even though it may seem to affect the order of the sequence: if $u_i < 0$, simply write the optimization problem as $\min_{p_i} \|\sigma u_i v - p_i q\|_1 = \min_{p_i} \|\sigma(-u_i)v - (-p_i)q\|_1$ where now $-u_i > 0$ and the variable is $-p_i$. The solution in the positive case yields $-p_i = \sigma(-u_i)\tilde{v}_{[k]}$, which is the same as $p_i = \sigma u_i \tilde{v}_{[k]}$.

Therefore we have that after the first iteration,

$$p = \tilde{v}_{[k]} u$$

since $\tilde{v}_{[k]}$ does not depend on i (the order of the sequence only depends on q), thus we immediately recover the direction of the left singular vector u. Similarly, q recovers the direction of the right singular vector v, and the algorithm converges after one iteration.

1.3.7 convergence in the general case

[so far, no results on convergence in the general case]

1.4 PCA as successive ℓ_p-projections on hyperplanes

PCA can be computed as successive projections on hyperplanes: at each step, the data points are projected on a hyperplane H that minimizes the sum of distances of the data points A_i to H, thus reducing the dimension of the subspace in which the data points lie. At each step, we look for a hyperplane H that solves the problem

$$\min_H \sum_{i=1}^{n} d(A_i, H)$$

1.5 ℓ_p-projection on a hyperplane

In the classical ℓ_2 PCA problem, the distance is the Euclidean distance. This can be generalized to any ℓ_p distance. [Need to justify this in the ℓ_p case]. In order to compute the best fit hyperplane in each step, we first need to characterize the projection of a vector on a hyperplane in the ℓ_p sense.

Consider the problem of projecting a vector $y \in \mathbb{R}^n$, with respect to a general norm $\|.\|$, on an affine hyperplane H given by

$$H = \{x \in \mathbb{R}^n | w^T(x - x_0) = 0\}$$

where x_0 is a given point, and w is a normal vector to the hyperplane.

The problem is

$$P_H(y) = \arg\min_{x \in H} \|y - x\| = \arg\min_{w^T(x-x_0)} \|y - x\|$$

the solution set $P_H(y)$ is given by

$$P_H(y) = y - \frac{w^T(y-x_0)}{\|w\|_*} \arg\max_{\|u\|=1} u^T w$$

in other words, the complement of the projection is

$$y - P_H(y) = \frac{w^T(y-x_0)}{\|w\|_*} \arg\max_{\|u\|=1} u^T w$$

We have that $|w^T(y-x_0)|/\|w\|_*$ is the distance of y to the hyperplane

$$d(y,H) = \frac{|w^T(y-x_0)|}{\|w\|_*}\|u\| = \frac{|w^T(y-x_0)|}{\|w\|_*}$$

and $\arg\max_{\|u\|=1} u^T w$ gives the direction of the projection. Note that this direction does not depend on y, but only on the direction w of the hyperplane. This is a familiar fact in the ℓ_2 projection case, since the projection direction is orthogonal to the hyperplane (thus aligned with w, and this is consistent with the above result $\arg\max_{\|u\|_2\le 1} u^T w = w/\|w\|_2$), and it is also the case for a general norm (but the direction of projection is not necessarily orthogonal to the hyperplane, and not necessarily unique).

To show this, we first check that $P_H(y) \subset H$ (note that $P_H(y)$ is a set, the projection of y on H is not necessarily unique). This is true since we have $\forall p \in P_H(y)$

$$\begin{aligned} w^T(p-x_0) &= w^T\left(y - \frac{w^T(y-x_0)}{\|w\|_*}u^* - x_0\right) && \text{where } u^* \in \arg\max_{\|u\|=1} u^T w \\ &= w^T(y-x_0) - \frac{w^T(y-x_0)}{\|w\|_*} w^T u^* \\ &= 0 && \text{since } \max_{\|u\|=1} u^T w = \|w\|_* \end{aligned}$$

then we check that for any $p \in P_H(y) \subset H$, and any other point $x \in H$, the distance $\|x-y\| \ge \|p-y\|$. One way to do this is to use the Lagrangian $L(x,\nu) = \|x-y\| - \nu w^T(x-x_0)$ and show that for some ν, $L(p,\nu) \le L(x,\nu)$ for all x. Then we would have in particular $\forall x \in H$, $\|p-x\| \le \|x-y\|$. Taking $\nu = \frac{\operatorname{sign}(w^T(y-x_0))}{\|w\|_*}$, we have

$$\begin{aligned} L(p,\nu) &= \|p-y\| - \nu w^T(x-x_0) \\ &= \frac{|w^T(y-x_0)|}{\|w\|_*} - \nu w^T(x-x_0) \\ &= \nu w^T(y-x_0) - \nu w^T(x-x_0) \\ &= \nu w^T(y-x) \\ &\le |\nu||w^T(y-x)| \\ &\le \frac{1}{\|w\|_*}\|w\|_*\|y-x\| \\ &= L(x,\nu) \end{aligned}$$

where the last inequality is obtained using the generalized Cauchy Schwartz inequality $|u^T v| \le \|u\|\|v\|_*$.

Alternative method Another way to show this result is to use the fact that $\|w\|_* = \max_{\|u\|=1} w^T u$ (also called the generalized Cauchy-Schwartz inequality $u^T w \le \|u\|\|w\|_*$). We have for any $x \in H$

$$\begin{aligned}
\|y-x\| &\geq \frac{w^T(y-x)}{\|w\|_*} \\
&= \frac{w^T(y-x)+w^T(x-x_0)}{\|w\|_*} && \text{since } x \in H \\
&= \frac{w^T(y-x_0)}{\|w\|_*}
\end{aligned}$$

with equality if and only if $\frac{y-x}{\|y-x\|} = u \in \arg\max_{\|u\|=1} w^T u$. The projection $P_H(y)$ of y on the hyperplane H is then the set of points p that satisfy

$$y - p \in \mathbb{R} \arg\max_{\|u\|=1} w^T u$$
$$\|y-p\| = \frac{w^T(y-x_0)}{\|w\|_*}$$

Thus

$$P_H(y) = y - \frac{w^T y}{\|w\|_*} \arg\max_{\|u\|=1} w^T u$$

1.5.1 Reducing the problem to linear Hyperplanes

If we can project a vector y on a linear hyperplane H, we can extend this to projecting on affine hyperplanes by translating the coordinate system: let $H = \{x \in \mathbb{R}^n | w^T(x-x_0) = 0\}$ be the affine hyperplane we want to project on, where $x_0 \in H$. Let $\tilde{H} = \{x \in \mathbb{R}^n | w^T x = 0\}$ be the linear hyperplane with same normal vector. Then projecting y on H is equivalent to projecting $y - x_0$ on $\tilde{H}$: the distance $d(y,H) = \min_{x \in H} \|y-x\| = \min_{x \in H} \|y - x_0 - (x-x_0)\| = \min_{x' \in \tilde{H}} \|y - x_0 - x'\|$ since $x \in H \Leftrightarrow x - x_0 \in \tilde{H}$.

1.5.2 Optimal ℓ_p-projection hyperplane

Given a set of points A_i, the optimal projection hyperplane $H = \{x | w^T(x-x_0) = 0\}$ is the minimizer of the problem

$$\min_H \sum_{i=1}^n d(A_i, H) = \min_{\|w\|_*=1, x_0} \sum_{i=1}^n |w^T(A_i - x_0)|$$

where $|w^T(A_i - x_0)|$ is the distance between A_i and the hyperplane. For a fixed w, optimizing on x_0 yields

1.5.3 Exact solution of the ℓ_1 rank one PCA using iterative projections

1.6 ℓ_1 PCA heuristics for higher ranks

One can generalize the ℓ_1 PCA block coordinate descent algorithms to cases where we want to approximate A with a rank r matrix, in two different ways depending on what we want to achieve, and on the performance requirements.

1.6.1 Analysis ℓ_1 PCA

In the analysis view of PCA, one seeks to *sequentially* find directions that best explain the data. In the ℓ_2 case, at each step, we look for a direction p that maximizes the variance along p, i.e. $\max_{\|p\|_2=1} p^T A A^T p = \max_{\|p\|_2=1} \|A^T p\|_2^2$. Equivalently, we look for p that minimizes the distances of the data points A_i to $\texttt{span}(p)$

(A_i is the i-th column of A). Indeed, assuming $\|p\|_2 = 1$, the projection of data point M_i on p is given by $(A_i^T p)p$, and the distance of A_i to $\mathtt{span}(p)$ is simply $\|A_i - (A_i^T p)p\|_2^2 = \|A_i\|_2^2 - (A_i^T p)^2$. Thus minimizing the sum of the squared distances is equivalent to

$$\min_{\|p\|_2=1} -\sum_i (A_i^T p)^2 = -\max_{\|p\|_2=1} \|A^T p\|_2^2$$

For a general norm $\|.\|$, minimizing the distance to $\mathbf{span}(p)$ can be written

$$\min_{q,\|p\|_*=1} \|A^T - pq^T\|$$

where p is normalized in the dual sense, and $q_i p$ is the projection of A_i on p.

The analysis view of ℓ_1 PCA leads to a sequential formulation where at each step, a one-dimensional projection of the data is computed

$$\min_{q_k,\|p_k\|_\infty=1} \|A^{(k)} - p_k q_k^T\|_1$$

then the data matrix is updated (the covariance matrix is deflated)

$$A^{(k+1)} = A^{(k)} - p_k q_k^T$$

One way to solve this problem is to use block coordinate descent *separately for each pair* (p_k, q_k) (i.e. one block coordinate descent for each step k).

1.6.2 Synthesis ℓ_1 PCA

In the synthesis view of PCA, one seeks to find, *simultaneously*, a small set of directions (of size r), or dictionary elements, such that each data point A_i admits a decomposition on that set with low error. This leads to the batch problem

$$\min_{\|p_k\|_\infty=1,q_k} \|A - \sum_{k=1}^{r} p_k q_k^T\|_1$$

that can be solved using block coordinate descent for all vectors p_k, q_k.

Note that in the case of ℓ_2 PCA, both views are equivalent and lead to the same solution (p_k is the left singular vector corresponding to the k-th largest singular value σ_k, and q_k/σ_k is the corresponding right singular vector). However in the general case, these formulations are not equivalent. The synthesis view has slower convergence, since the optimization is done simultaneously on all vectors. The analysis view has faster convergence, but performs poorly compared to the analysis view if the criterion is the reconstruction error $\|A - \sum_{k=1}^r p_k q_k^T\|_1$.